Journal of Computational Physics 348 (2017) 567–590

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A parallel interaction potential approach coupled with the immersed boundary method for fully resolved simulations of deformable interfaces and membranes

Vamsi Spandan [a,*], Valentina Meschini [b], Rodolfo Ostilla-Mónico [c], Detlef Lohse [a,d], Giorgio Querzoli [e], Marco D. de Tullio [f], Roberto Verzicco [a,g,*]

[a] Physics of Fluids, University of Twente, Enschede, PO Box 217, 7500 AE, Netherlands
[b] Gran Sasso Science Institute, Viale Francesco Crispi, 7, L'Aquila, Italy
[c] Harvard John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA
[d] Max Planck Institute for Dynamics and Self-Organization, 37077, Göttingen, Germany
[e] DICAAR – University of Cagliari, via Marengo 2, 09123 Cagliari, Italy
[f] Politecnico di Bari, Via Re David 200, 70125, Bari, Italy
[g] University of Rome 'Tor Vergata', Via del Politecnico, Rome 00133, Italy

ARTICLE INFO

Article history:
Received 11 December 2016
Received in revised form 24 May 2017
Accepted 19 July 2017
Available online 26 July 2017

Keywords:
Fluid–structure interaction
Spring networks
Liquid interfaces and membranes

ABSTRACT

In this paper we show and discuss how the deformation dynamics of closed liquid–liquid interfaces (for example drops and bubbles) can be replicated with use of a phenomenological interaction potential model. This new approach to simulate liquid–liquid interfaces is based on the fundamental principle of minimum potential energy where the total potential energy depends on the extent of deformation of a spring network distributed on the surface of the immersed drop or bubble. Simulating liquid–liquid interfaces using this model require computing ad-hoc elastic constants which is done through a reverse-engineered approach. The results from our simulations agree very well with previous studies on the deformation of drops in standard flow configurations such as a deforming drop in a shear flow or cross flow. The interaction potential model is highly versatile, computationally efficient and can be easily incorporated into generic single phase fluid solvers to also simulate complex fluid–structure interaction problems. This is shown by simulating flow in the left ventricle of the heart with mechanical and natural mitral valves where the imposed flow, motion of ventricle and valves dynamically govern the behaviour of each other. Results from these simulations are compared with ad-hoc in-house experimental measurements. Finally, we present a simple and easy to implement parallelisation scheme, as high performance computing is unavoidable when studying large scale problems involving several thousands of simultaneously deforming bodies in highly turbulent flows.

* Corresponding authors.
E-mail addresses: vamsispandan@gmail.com (V. Spandan), verzicco@uniroma2.it (R. Verzicco).

http://dx.doi.org/10.1016/j.jcp.2017.07.036
0021-9991/

1. Introduction

The interaction between fluid flow and an immersed elastic body (fluid or solid) has been studied extensively over the last few decades due to its wide range of applications, for example, bubbles and drops dispersed in a turbulent flow [1,2], red blood cells flowing through blood vessels [3], pumping motion of ventricles and valves in the heart [4,5], oscillation of large structures such as aircraft wings and high-rise buildings etc. [6]. While the source of elasticity of the immersed body in each of these phenomena is different, the interplay between a deformable body and a surrounding inhomogeneous time dependent flow can result in a complex non-linear system where they determine each other's behaviour in a coupled manner. Additionally, the presence of multiple bodies with different static and dynamic properties interacting with each other gives rise to a wide range of control parameters which makes these systems extremely challenging to study. Over the last few decades, tremendous amount of effort has been devoted to the modelling and simulation of such systems which are often classified in literature as Fluid–Structure Interaction (FSI) problems. Among a variety of techniques developed to tackle FSI problems, the immersed boundary method (IBM) [7–9] has gained immense popularity and has been instrumental in making efficient and accurate simulations of several complex flow systems possible; for example cardiac and vascular hemodynamics [4,5], suspensions of rigid spheres [10–13], deformable bubbles or drops [14], vehicle aerodynamics [15,16] etc.

One of the biggest advantages of IBM is that it relies on the use of a single underlying mesh for the fluid flow (hereafter referred to as Eulerian mesh) which does not have to conform/adapt with the moving/deforming immersed body [7–9]. This eliminates the complex and computationally expensive procedure of Eulerian mesh regeneration every time step as the immersed body moves or deforms, resulting in the decoupling of the mesh required for the flow solver from the position and morphology of the immersed body. The surface of the immersed body is discretised independently of the Eulerian mesh and is often called a Lagrangian or a structural mesh. The influence of the immersed body on the flow can be achieved through a volume averaged body force in the fluid governing equations after a careful transfer of information between the Eulerian and Lagrangian meshes. Additionally, the time invariant nature of the Eulerian mesh makes IBM promising for parallelisation on multiple distributed memory computing processors and this has led to breakthroughs in simulations of highly turbulent flows around complex geometries. However, the inclusion of deformability into the immersed boundary framework to study the motion of liquid–liquid interfaces or elastic membranes is not straightforward and this is the focus of this paper.

In the field of multiphase flows, several techniques have been developed to understand the motion and influence of particles, drops or bubbles in a turbulent flow (e.g. point-particle, Volume of Fluid, level-set, front tracking, Physalis etc.) [17]. When the inherent surface tension forces in the bubbles or drops are not strong enough, they deform according to local flow conditions and also alter them simultaneously. The challenge in direct numerical simulations of such flows arises from the wide range of length, time scales and regimes involved, see [2,18,19] for detailed reviews. Numerically handling a sharp boundary between different phases and the singularity of the surface tension term in the governing equations is non-trivial [20]. Additionally, since the density is not uniform in the flow domain, the pressure cannot be computed using fast Poisson solvers, but through relatively slower iterative methods. The complex algorithms and procedures put in place to tackle the above mentioned issues have restricted the scale of multiphase flows that can be studied; for example, state-of-art parallel simulations can only reach up to $O(10^2)$ deformable drops/bubbles in a reasonably turbulent flow [2]. In order to scale up numerical simulations of dispersed multiphase flows, there is a need for development of alternative methods which are relatively computationally inexpensive and still account for the various active physical mechanisms in the flow.

In the first part of this paper, we show the validity and use of a new phenomenological approach to replicate the deformation of closed liquid–liquid interfaces under given flow conditions. One of the big advantages of this approach is the computationally inexpensive nature of the deformation algorithm which allows for simulations of large scale dispersed multiphase flows. The deformability of any immersed drop or bubble is replicated by solving for the dynamics of a three-dimensional spring network spread over its surface. Hereafter we refer to this technique as the interaction potential (IP) model. The Navier–Stokes equations which govern the fluid motion inside and around the immersed body are solved using direct numerical simulations (DNS) while IBM is used to enforce the interfacial boundary conditions (for example no-slip or free-slip). Thus the fluid motion and the effect of an immersed interface on the flow is fully resolved, while the deformability is captured through a stable, versatile, easy to implement and computationally inexpensive IP model. In later sections we will further discuss why the IP model is able to sufficiently replicate the dynamics of deforming liquid–liquid interfaces.

As mentioned earlier, the IP model is extremely versatile and has been used previously by de Tullio and Pascazio [21] within the immersed boundary framework to simulate elastic bodies with arbitrary thickness such as flapping flags, leaflets of heart valves, thin elastic sheets etc. In the second part of the paper, we show that it is possible to extend the approach to more complex FSI problems by performing a three dimensional simulation of the flow in the left heart ventricle with mechanical and biological valves. In particular, we focus on the simulation of the left ventricle of the human heart along with a physical mitral valve in both pathological and physiological conditions. IBM has been a front runner in the field of cardio-vascular hemodynamics as has been evidenced in several previous studies [22–29]. A main impulse in developing this area of study from a computational fluid dynamics point of view is the increasing demand from the medical community for scientifically rigorous and quantitative investigations of cardiovascular diseases. Again, the major bottleneck in conducting fully resolved simulations of the complete human heart is created by the complex deformation dynamics of the various ventricles and valves which interact with the pulsatile blood flow (see [5] for a recent review).

Various approaches have been employed over the years to achieve realistic and reliable cardiac hemodynamic simulations. One approach is to use available models of the heart functionality along with the biophysical component of cardiac electromechanics. This has been used in the 'Living Heart Project' [30], where a fully coupled electro-mechanic and hemodynamic simulation is realised, and more recently by Choi et al. [22], who coupled a multi-scale model for electromechanics with the Navier–Stokes equations for the flow dynamics. The work of Zheng et al. [26] and Seo and Mittal [27] focused on intra-ventricular flow and the accompanying pathologies under the effect of a diastolic flow pattern while Seo et al. [28] studied the effect of the mitral valve on the flow dynamics. In all these simulations, the deformation cum motion of the left ventricle and the valves are imposed either through kinematic models or derived through imaging but not through a fully coupled fluid–structure interaction simulation.

The motivation in employing kinematic models to describe the motion of ventricles and valves instead of a fully coupled FSI simulation is to eliminate the massive computational cost in solving the three dimensional Cauchy–Navier equations for the immersed elastic body along with the Navier–Stokes equations. Although numerical simulations with pre-defined ventricle/valve motion is a challenging task in itself, in reality the motion of the ventricle, valves, and the fluid are coupled to each other and can govern each other's motion.

The FSI simulation of the left ventricle with a physical valve which is described in detail later is performed within the equivalent of 48 CPU hours on a single processor with an Eulerian grid of $150 \times 150 \times 150$ and a Lagrangian mesh of approximately 50000 elements on the ventricle which shows the computationally inexpensive nature of the IP approach. Additionally, these simulations are compared and validated with in-house ad-hoc laboratory experiments to ensure the reliability of the results. While we do not solve for computationally challenging coupled three dimensional Navier–Stokes Cauchy–Navier equations, we take a step further from employing kinematic models towards having full FSI between the flow, ventricle and the mitral valves. In order to further throw light on the computational time gained by the IP approach, a relatively simpler problem of flow across an aortic valve within a deformable aortic root which used 10^6 nodes for the fluid solver and 10^3 triangular elements for the structure [25] requires the same computational time as that of the IP approach to model the full left ventricle with physical mitral valves. In comparison to the FEM solver, the IP approach needs only 3% of the CPU time per time step for the structural solver.

The IBM described in this paper makes use of the moving-least-squares (MLS) approximation which is crucial when the system involves moving and deforming boundaries. However, the computational cost of MLS increases rapidly with increase in the resolution of the immersed bodies. New algorithms or a parallel implementation of the computation on distributed memory processors thus become an invaluable tool in scaling up fully resolved flows interacting with several moving/deforming immersed bodies. In particular, parallelisation is an attractive prospect given the increasing availability of cost-efficient high performance computing facilities. A parallel implementation of a flow solver involving multiple deforming bodies is not a straightforward task due to many algorithmic complexities. The challenge lies underneath the fact that two different meshes (Eulerian and Lagrangian) are required for the complete solution and different parallelisation strategies would be required to make use of multiple processors effectively.

Keeping this in mind, in the last part of the paper we describe a parallelisation scheme designed to track the time evolution of several deformable bodies (e.g. vesicles, drops, biological tissues etc.) immersed in turbulent flows. This strategy is built upon the already underlying parallelisation scheme implemented for the fluid solver, thus reducing the downtime of overall code development. In particular, the benefits of the parallelisation is oriented towards simulation of dispersed phase systems with several thousand deforming drops, bubbles, vesicles or bodies moving in a highly turbulent carrier fluid phase.

The novelty of the current work is three fold. We first discuss the extension of the algorithm to liquid–liquid interfaces which is not straight-forward. From a theoretical point of view, we take advantage of the fact that the deformation of any immersed interface or membrane are both based on the fundamental principle of minimum potential energy. This has not been taken advantage of in its full extent for liquid–liquid interfaces in previous works and given its computational efficiency and ease of implementation it can be extremely beneficial for studies involving large scale dispersed multiphase flows where traditional techniques become too expensive. We also demonstrate that the method is extremely versatile and that the approach can not only handle simple elastic structures but also tackle complex FSI problems; for example flow in the left heart ventricle with mitral valves. This is a step forward in the field of cardio-vascular simulations where the conventional approach is to derive the motion of the ventricle from kinematic models or imaging techniques. Although the discussed approach is computationally inexpensive, parallelisation is inevitable for handling thousands of immersed bodies in a turbulent flow which is discussed in the last part of the paper.

In the next section we give an overview of the governing equations for the solution of the fluid phase, implementation of the immersed boundary method using MLS and the interaction potential approach for computing the deformation of elastic bodies. In section 3, we show how the interaction potential approach can be used to study deformation of drops/bubbles where the flow is dynamically coupled with the interface morphology. These results are validated with analytical solutions and experimental measurements taken from literature. In section 4, we describe the simulation of the full left ventricle with both mechanical and natural mitral valves in addition to comparing the results from our simulations with ad-hoc in-house experiments. In section 5, we discuss the data structures required and also the parallelisation strategy to scale up the problem to study several thousand deforming immersed bodies. Finally, we provide a summary and outlook in section 6.

2. Governing equations and numerical scheme

2.1. Fluid phase

For the fluid phase we solve the Navier–Stokes equations governing incompressible flow in a Cartesian box as given in equations (1), (2). In the following text, velocity and force fields on the Lagrangian and Eulerian meshes are represented by upper case and lower case symbols, respectively.

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

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

Reynolds number of the flow is defined based on a characteristic length scale l and velocity scale u as $Re = ul/\nu$; ν is the kinematic viscosity of the fluid. $\mathbf{u}$, p are the velocities and pressure in the flow while $\mathbf{f}_b$ is the volume averaged force arising from the IBM and is included to enforce the interfacial boundary condition. A conservative second-order centred finite-difference scheme with velocities on a staggered grid is used for spatial discretisation; explicit Adams–Bashforth scheme is used to discretise the non-linear terms while an implicit Crank–Nicholson scheme is used for the viscous terms. Treating all the viscous terms implicitly results in a large sparse matrix which is avoided by an approximate factorisation of the sparse matrix into three tridiagonal matrices (one for each direction) which are solved using Thomas' algorithm. Time integration is performed via a self starting fractional step third-order Runge–Kutta (RK3) scheme. The pressure required to enforce mass conservation is computed by solving a Poisson equation for a pressure correction. The code for single phase flows has already been tested extensively in previous studies for a variety of flow configurations and additional details of the numerical scheme can be found in [31–33].

2.2. Dispersed phase: immersed boundary method

We now describe the procedure of constructing the Lagrangian mesh and the schemes used to transfer flow quantities between the Lagrangian and Eulerian mesh which is a crucial ingredient in IBM. The details of the methodology are reported in de Tullio et al. [21] and are included here for convenience. In Fig. 1, we show a schematic of the various Lagrangian meshes used in this study. Any given surface (closed or open) is discretised into triangular elements where each element is composed of three vertices (v_1, v_2, v_3) which are connected by edges (e_1, e_2, e_3). Under the condition that the mass is uniformly distributed on the triangular element, the position of the centroid (c_1) of each triangular element is computed based on the co-ordinates of the vertices. Fig. 1(a) shows a sphere discretised into triangular elements along with a schematic showing the composition of each triangle. In Fig. 1(b), we show the discretised geometry of the left ventricle and Fig. 1(c) shows the remaining auxiliary components.

Following the idea of Uhlmann [34] the force required to enforce the interfacial boundary condition is first computed on markers laid out on the Lagrangian mesh and then transferred to the Eulerian mesh. Here, we consider the triangle centroids to be the Lagrangian markers and are responsible for enforcing the interfacial boundary condition. The vertices and edges of the discretised triangular elements play a role in the deformation dynamics and will be explained later.

The next step is to build a transfer function around each Lagrangian marker (here the centroid c_i) which would be used to exchange information between the Eulerian and Lagrangian mesh. As noted previously, we adopt the moving-least-squares (MLS) [35,36] approach which is part of the class of meshless approximations and has been used previously in several fields such as element free Galerkin methods [37–41], computer graphics [42–46], and also recently for IBM [21,47]. In order to compute this transfer function we first need to build a support domain centred around each Lagrangian marker which consists of all Eulerian grid nodes closer than a threshold value in each direction. By taking a threshold value of $r_i = 1.5\Delta x_i$ in each direction, a three-dimensional support domain consisting of $N_e = 27$ ($3\times3\times3$) Eulerian nodes is built around each Lagrangian marker. In Fig. 2(a) we show a schematic of a two dimensional support domain consisting of nine Eulerian cells around the centroid c1. The next step is to use these Eulerian cells and build a transfer function through which any quantity q_i defined on the Eulerian nodes can be interpolated on the Lagrangian marker (i.e. centroid of each triangular element). The same transfer function can be used to extrapolate the force computed on the Lagrangian markers (F_i) to the Eulerian mesh (f_i).

The MLS interpolation of q_i at a Lagrangian marker (c_i) is defined as follows.

$$Q_i(\boldsymbol{x}^l) = \boldsymbol{p}^T(\boldsymbol{x}^l)\boldsymbol{a}(\boldsymbol{x}^l) = \sum_{j=1}^{m} p_j(\boldsymbol{x}^l)a_j(\boldsymbol{x}^l) \tag{3}$$

In equation (3), Q_i is the quantity interpolated on the Lagrangian marker while $\boldsymbol{p}^T(\boldsymbol{x}^l)$ is a basis function vector with dimension m. $\boldsymbol{x}$ is the position vector of the Lagrangian marker. In this work we consider a linear basis function with $\boldsymbol{p}^T(\boldsymbol{x}^l) = [1, x, y, z]$, i.e. $m = 4$, which is cost-efficient and also able to represent the gradients in the Eulerian field with second order accuracy. $\boldsymbol{a}(\boldsymbol{x}^l)$ is the vector of coefficients obtained by minimising the weighted L2 norm J as follows:

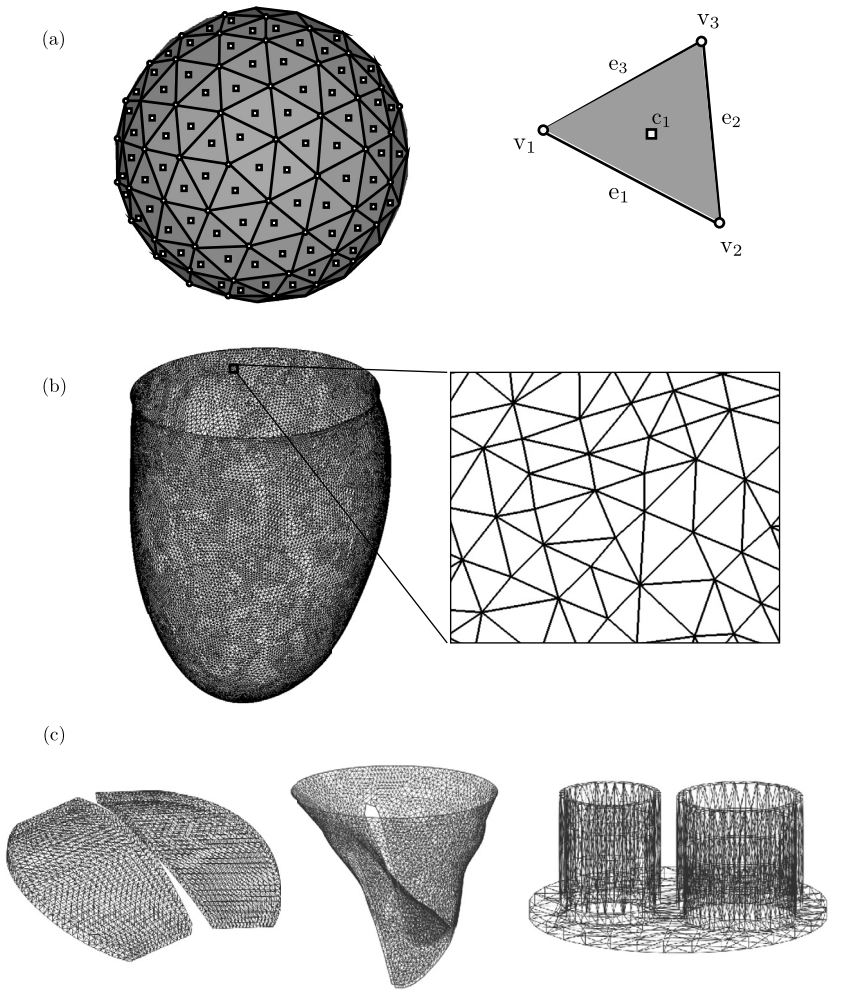

Fig. 1. Schematic of the Lagrangian mesh. (a) A sphere discretised using triangular elements. On the right a single triangular element is decomposed into three vertices v_1,v_2,v_3 (circles), three edges e_1,e_2,e_3 and one centroid c_1 (square). (b) Full structure of the left ventricle with a zoomed-in area showing the triangulated network. (c) Rest of the components of the full left ventricle structure. The left panel shows the leaflets of prosthetic mechanical mitral valve, the middle panel shows the natural mitral valve and the right panel the channels for mitral and aortic valves.

$$J = \sum_{k=1}^{N_e} W(\boldsymbol{x}^l - \boldsymbol{x}^k)[\boldsymbol{p}^T(\boldsymbol{x}^k)\boldsymbol{a}(\boldsymbol{x}^l) - q_i^k]^2 \tag{4}$$

Here $W(\boldsymbol{x}^l - \boldsymbol{x}^k)$ is a weight function; we use the exponential weight function which is given as follows.

$$W(\boldsymbol{x}^l - \boldsymbol{x}^k) = \begin{cases} e^{-(r_k/\alpha)^2}, & r_k \leq 1 \\ 0, & r_k > 1 \end{cases} \tag{5}$$

where α is a constant of shape parameter and r_k is given by

$$r_k = \frac{|\boldsymbol{x}^L - \boldsymbol{x}^k|}{r_i} \tag{6}$$

r_i is the size of the support domain in the ith direction as defined previously. Other commonly used shape functions are the cubic spline and quadratic spline functions and a spline function with any order of continuity can be constructed using the steps detailed in Liu and Gao [36]. Minimising J in equation (4) leads to $\boldsymbol{A}(\boldsymbol{x}^l)\boldsymbol{a}(\boldsymbol{x}^l) = \boldsymbol{B}(\boldsymbol{x}^l)\boldsymbol{q}_i^k$ where

$$\boldsymbol{A}(\boldsymbol{x}^l) = \sum_{k=1}^{N_e} W(\boldsymbol{x}^l - \boldsymbol{x}^k)\boldsymbol{p}(\boldsymbol{x}^k)\boldsymbol{p}^T(\boldsymbol{x}^k) \tag{7}$$

$$\boldsymbol{B}(\boldsymbol{x}^l) = [W(\boldsymbol{x}^l - \boldsymbol{x}^1)\boldsymbol{p}^T(\boldsymbol{x}^1)...W(\boldsymbol{x}^l - \boldsymbol{x}^{N_e})\boldsymbol{p}^T(\boldsymbol{x}^{N_e})] \tag{8}$$

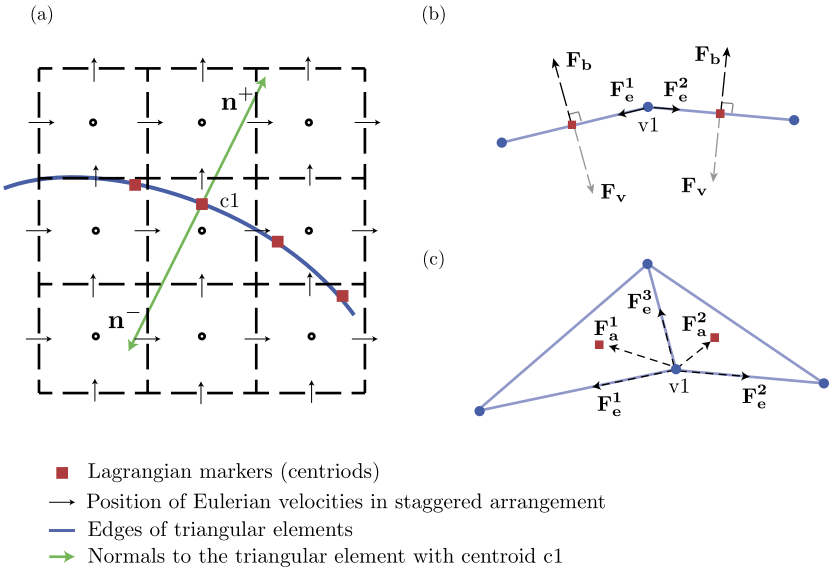

Fig. 2. (a) Schematic of a two-dimensional support domain around the centroid c1 (red squares) consisting of nine Eulerian cells; arrows show the position of Eulerian velocities in a staggered formation. (b), (c) Schematic of the direction of the various elastic forces acting on the triangular elements. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\boldsymbol{q}_i = [q_i^1 \ldots q_i^{N_e}]^T \tag{9}$$

Combining all the above equations, the interpolated quantity Q_i can be expressed as follows.

$$Q_i(\boldsymbol{x}^l) = \boldsymbol{\phi}^T(\boldsymbol{x}^l)\boldsymbol{q}_i = \sum_{k=1}^{N_e} \phi_k^l(\boldsymbol{x}^l) q_i^k \tag{10}$$

$\boldsymbol{\phi}^T(\boldsymbol{x}^l) = \boldsymbol{p}^T(\boldsymbol{x}^l)\boldsymbol{A}^{-1}(\boldsymbol{x}^l)B(\boldsymbol{x}^l)$ is the transfer function containing the shape function coefficients for each Lagrangian marker. This shape function is used to interpolate the value of the intermediate Eulerian velocity $\hat{u}_i$ at the exact location of all Lagrangian markers and the volume force in each direction is calculated as $F_i = (V_i^b - U_i)/\Delta t$, where V_i^b is the desired velocity boundary condition on the Lagrangian marker (c_i) and U_i is the Eulerian flow velocity interpolated on the Lagrangian marker using MLS. Under the assumption of a no-slip boundary condition on the interface, the desired velocity V_i^b is equal to the velocity of the corresponding centroid. This force needs to be transferred back to the Eulerian mesh using the same transfer function built for interpolation in equation (10) under the constraint that the total force is conserved during the extrapolation. The force to be included in the Eulerian mesh is written as $f_{b,i}^k = \sum_{k=1}^{N_L} c_l \phi_k^L F_i^L$, where N_L is the number of Lagrangian markers associated with a Eulerian point k. c_l is a scaling factor obtained by imposing the condition that there is no net-gain/loss in the IBM force while transferring flow information from the Lagrangian mesh to the Eulerian mesh which results in the following.

$$c_l = \frac{\Delta V^l}{\sum_{k=1}^{N_e} \phi_k^l \Delta V^K} \tag{11}$$

ΔV^L is the forcing volume associated with each Lagrangian marker and is computed as $\Delta V^l = A_l h_l$. A_l is the area of the triangular element associated with the Lagrangian marker (area composed by v_1, v_2, v_3 in Fig. 1) and $h_l = 1/3 \sum_{k=1}^{N_e} \phi_k^L(\Delta x^k + \Delta y^k + \Delta z^k)$. ΔV^k is the volume of the Eulerian cell k involved in the support domain. Here it is important to note that the transfer functions built using this approach conserves momentum on both uniform and stretched grids while reasonable accuracy is retained for torque equivalence on slightly stretched grids [21,47]. For example, Vanella and Balaras [47] report that with 10% grid stretching, the net loss/gain in torque conservation is less than 0.5%.

The calculation of hydrodynamic forces (pressure and viscous stresses) acting on the surface of any dispersed body in an IBM simulation is not straightforward as the Lagrangian and the Eulerian meshes do not necessarily align with each other at a given time instant. Since the surface of the dispersed bodies are discretised using triangular elements, the local pressure and viscous forces are first computed on the Lagrangian markers (centroids of triangular elements); the total external force on a triangular element with area A_l and surface normal $\boldsymbol{n}_l$ is calculated as $\boldsymbol{F}_{\text{ext}}^l = (-p_l \boldsymbol{n}_l + \boldsymbol{\tau}_l \cdot \boldsymbol{n}_l)A_l$. To evaluate p_l and τ_l, which are the pressure and viscous forces acting on a triangular element l, respectively a probe is sent along the normal of each triangular element with its centroid as the origin. The length of the probe h_l is equal to the mean local grid size and the MLS interpolation described above is used to interpolate both pressure and velocity gradients at the end point of

the probe. The velocity gradients can also be computed from the derivatives of the shape functions [36]. The value of the pressure gradient along this normal is computed from the momentum equation normal to the triangular element which gives $\frac{dp}{dn} = -\frac{D\boldsymbol{U}_L}{dt} \cdot \boldsymbol{n}_L$. The pressure on the Lagrangian marker (centroid) is then computed as follows:

$$p_l = p_l^* + h_l \frac{D\boldsymbol{U}_l}{Dt} \cdot \boldsymbol{n}_l \tag{12}$$

p_l^* is the pressure at the probe endpoint and $\frac{D\boldsymbol{U}_l}{Dt}$ is the acceleration of the Lagrangian marker [21,47,48]. The shear stress $\boldsymbol{\tau}_l$ on the Lagrangian marker is computed based on the velocity gradients interpolated at the probe endpoint. This holds true under the assumption that the velocity of the fluid near the surface of the body varies linearly. An important note here is that when the nature of the immersed body is such that fluid loads on either side of the interface are relevant, the pressure and viscous forces need to computed on both sides of the surface thus requiring two probes sent along the normal to every triangular element, one each along the positive and negative normal, respectively i.e. $\boldsymbol{F}^l_{\text{ext}} = [(-(p_l^+ - p_l^-)\boldsymbol{n}_l^+ + (\boldsymbol{\tau}_l^+ - \boldsymbol{\tau}_l^-) \cdot \boldsymbol{n}_l^+)]A_l$; the superscripts $^+$ and $^-$ represent quantities evaluated on the end points of the probe on either side of the surface [21,47,48].

2.3. Interaction potential approach for deformation

As mentioned in section 1, the dynamics of deformation is computed based on a minimum energy concept which we describe here briefly. The surface of any immersed body is first discretised using triangular elements (cf. Fig. 1) the edges of which are composed of hypothetical linear/non-linear springs thus resulting in a two-dimensional network of springs. Under the influence of external forces such as pressure fluctuations or viscous stresses, the spring network undergoes deformation thus storing potential energy into the system. The potential is converted to nodal forces acting on individual triangular vertices by differentiating the potentials with respect to its corresponding displacement. The force acting on each triangular vertex is converted into an acceleration and integrated based on Newton's second law of motion. The position of each individual vertex is then accordingly.

The first form of potential is the in-plane elastic potential (W_e) which comes from the work done by an external force parallel to the plane of a triangular face and is converted into elastic energy stored into every spring connecting the triangle. We also consider an out-of-plane deformation (W_b) for which the total potential is computed based on a bending spring connecting the centroids of two adjacent triangular faces. This out-of-plane bending potential is stored in a pair of two faces sharing an edge and is a function of the contact angle between them. Additional potentials can be included which constrain the geometrical properties of the overall immersed body. For example, we can include a volume or area potential (W_v or W_a) which is a function of the change in volume/area of a single element with respect to an initial reference state. All the individual potentials are formed as given in the equations below.

$$W_e = \frac{1}{2} k_e \boldsymbol{x}^2 \tag{13}$$

$$W_b = k_b(1 - cos\theta) \tag{14}$$

$$W_v = \frac{1}{2} k_v \left(\frac{V - V_0}{V_0} \right)^2 V_0 \tag{15}$$

$$W_a = \frac{1}{2} k_a \left(\frac{A - A_0}{A_0} \right)^2 A_0 \tag{16}$$

In the above equations, k_e, k_b, k_v, and k_a are the elastic constants for in-plane deformation, out-of-plane deformation, volume constraint and area constraint potentials, respectively. $\boldsymbol{x}$ is the change in length of a single edge; θ is the angle between the normals of two triangular faces sharing an edge; V_0, V and A_0, A are the corresponding initial (reference) and deformed volumes and areas of each triangular element, respectively. While the equations (13)–(16) can be used to simulate homogeneous isotropic materials, the interaction potential approach can also be used for inhomogeneous anisotropic materials by changing the functional form of the elastic potentials [21].

In Fig. 2(b), (c) we show a 2D and 3D schematic, respectively of the forces originating from these potentials on the vertices of the triangular elements. $\boldsymbol{F}_e$ is the in-plane elastic force and it acts along the edges connecting two vertices; $\boldsymbol{F}_b$ and $\boldsymbol{F}_v$ are the out of plane bending and volume constraint forces which act along the normal at the centroids of each triangular element. $\boldsymbol{F}_a$ is the force originating from the area constraint potential and is directed towards the centroid of the triangular element. Once the forces on each of the triangular vertices are known, individual nodes are moved based on the equation $m\ddot{\boldsymbol{x}}^{\mathrm{v}_i} = F^{\mathrm{v}_i}_{\text{ext}} + F^{\mathrm{v}_i}_{\text{int}}$; $F^{\mathrm{v}_i}_{\text{ext}}$ and $F^{\mathrm{v}_i}_{\text{int}}$ are the external and internal forces acting on the triangular node v_i, $\ddot{\boldsymbol{x}}$ and m are the acceleration and mass of individual nodes. In the previous section F_{ext} was calculated on the centroid of each triangle. This force is transferred to an individual triangular vertex as $F^{\mathrm{v}_i}_{\text{ext}} = \sum_{j=1}^{\mathrm{n}_{fi}} (1/3) F^{\mathrm{c}_j}_{\text{ext}}$; where n_{fi} is the number of faces each vertex is connected to and $F^{\mathrm{c}_j}_{\text{ext}}$ is the external force computed on the triangle centroid (Lagrangian marker) c_j. The calculation of m, which is the mass of individual triangular node is straightforward for surfaces made of materials where

the density and thickness is known a priori. In cases where the immersed bodies are drops or bubbles, calculating m of the triangular nodes becomes tricky as there is no physical definition of the density and thickness of a liquid–liquid interface. In such a case m of the triangular nodes becomes a free parameter and to overcome this we fix the value of $m = 1$ and then correspondingly tune the elastic constants. This is detailed more in the next section.

Computing the individual potentials according to equations (13)–(16) in the interaction potential approach requires selection of several parameters (k_e, k_b, k_v, k_a). Once again, this step is straightforward for membranes where the elastic moduli are already known [21]. In a later section where we show simulations of flow in the heart ventricle we will further show how we compute the elastic constants from the physical properties of the material that is used. Additionally, it is important to note that when the surface is discretised with non-uniform triangles such that the lengths of the edges of triangle vary, k_e should be computed based on the model proposed by van Gelder [49]. This is to ensure that at the rest state of the elastic structure the properties of all the springs connecting the vertices are equivalent. On the other hand, simulating liquid–liquid interfaces using the interaction potential approach would require the use of ad-hoc elastic constants as again there is no direct physical correlation between the elastic constants and the surface properties of a liquid–liquid interface. The procedure of estimating these ad-hoc elastic constants will be described in detail in the next section.

Here it is important to note that modelling an elastic membrane or an interface using the interaction potential approach is a discrete formulation of the elasticity governing equations and thus a simplification of the existing continuum models. It has been shown in previous studies that through a careful design of the spring network and the selection of appropriate elastic constants the mechanics of several elastic membranes can be exactly reproduced [21,50,51]. While such a formulation is useful and necessary for complex simulations of several immersed deformable bodies owing to its simplicity and lower computational cost, it has its limitations. In particular for liquid–liquid interfaces, when the drops/bubbles deform to such an extent that they approach the critical Capillary/Weber number for breakup, the Lagrangian resolutions become terminally high. Such scenarios are better handled with techniques such as VOF or level-set. When the Capillary/Weber numbers for the drops/bubbles are well below their breakup limit, the IP model provides a viable alternative to simulate large scale dispersed multiphase flows with realistic computational expenses. Here we would like to note that the derivation of the interaction potential approach is not unique to this work and variants of this method have already been used previously to predominantly study red blood cells [50–53]. In this work we show how this approach can also be extended for more complex cases such as large scale flows with dispersed deforming drops/bubbles and also biological membranes with full fluid–structure interaction, for example flow in heart ventricles with valves. In the case of biological tissues, the deformable membranes can be hyperelastic and orthotropic. The versatility of the IP model easily allows us to extend the method to such materials by changing the formulation of the potentials and deriving the corresponding elastic constants. Additionally, electrophysiology can also be implemented into the IP model (this is currently in progress) by solving a constitutive equation for an electric potential along the surface of the membrane. In this work, the flow in the heart ventricle is driven using an inflow–outflow boundary condition to facilitate comparison with in-house experiments. Thick boundaries (i.e. with thickness larger than the local Eulerian resolution) or volumetric structural elements cannot be handled with the present approach, given the membrane/shell-like formulation of the model.

Another important issue in the simulations involving FSI problems is the type of coupling used i.e. loose (explicit) versus strong (implicit). In the loosely coupled (explicit) case, the fluid and the immersed body governing equations are solved separately one after the another with a transfer of information between them every time step. On the other hand, in the strongly coupled (implicit) case the governing equations are solved in an iterative manner for each time step using a predictor corrector scheme until sufficient convergence is achieved. A detailed solution procedure for a strongly coupled IBM-FSI Navier Stokes solver with the provision for the interaction potential approach is given in de Tullio and Pascazio [21] where the governing equations are solved using a Hammingś fourth order predictor–corrector scheme. In our code, we have provisions for both a strong (implicit) and weak (explicit) coupling between the fluid and the immersed body. For the simulations shown in the following sections, loose coupling is used given its computationally inexpensive nature. Also it has to be remembered that strong coupling is only needed when added mass effects from the immersed body become important and the recent work of Schwarz et al. [54] gives insights into how this can be tackled smartly while there is still loose coupling between the fluid and the immersed body. We now move on to combining DNS of the fluid governing equations along with a moving least squares IBM coupled with the interaction potential approach to simulate deformable drops/bubbles and heart ventricles/valves.

3. Liquid–liquid interface dynamics using the potential approach

In order to replicate the deformation dynamics of drops or bubbles using the IP model, we first need to devise a method to compute the elastic constants of a given spring network which can represent a liquid–liquid interface with a given surface tension. As mentioned earlier, this is not straight-forward since there is no direct physical correlation between the elastic constants and the surface tension of a liquid–liquid interface. Here we use a reverse-engineered approach and perform a single simulation with a set of intuitively chosen elastic constants and estimate the surface tension of the immersed drop by comparing its morphology with previously known analytical solutions. By using the same set of elastic constants but for different flow conditions we also show that such an approach is self-consistent and reliable. Our goal here is to show that the IP model for deformation can be used to replicate the deformation dynamics of liquid–liquid interfaces under given flow conditions.

Before we analyse the results, it important to understand from a theoretical point of view why the IP model can be expected to mimic the deformation behaviour of drops/bubbles. When an initially spherical drop is immersed in a fluctuating flow field, the viscous and pressure stresses tend to deform the drop while the surface tension forces tend to resist the deformation and bring it back to its original spherical shape. During deformation, the total potential energy of the deformation stored in the drops surface is given by $W = \int[(p_i - p_o) - 0.5\gamma(1/R_1 + 1/R_2)]d\boldsymbol{n} \cdot d\boldsymbol{A}$, here p_i, p_o are the pressures inside and outside the drop, γ is the surface tension while R_1, R_2 are the principal radii of curvature [55]. Under the action of external forces, the shape of the drop adjusts itself such that the total displacement potential energy W tends to a minimum. The shape of the deformed drop can be computed by parametrising the surface in such a manner that it satisfies the above condition under certain constraints (for example an ellipsoidal shape or more complex shapes using spherical harmonics) [14,56]. The IP model is mathematically analogous in a way that the forces on each triangular vertex (cf. Fig. 2(b), (c)) act such that the total potential energy of deformation $\Pi = W_e + W_b + W_v + W_a$ tends to a minimum. Although the IP model for deforming drops is not an exact representation of the surface tension phenomena it can be considered as a phenomenological approach which can mimic the drop deformation characteristics given the fact that both the exact representation and the IP model rely on the fundamental principle of minimum potential energy.

3.1. Deformation of a neutrally buoyant drop in shear flow

For the first test case we choose a neutrally buoyant drop deforming in a laminar shear flow which has a simple configuration and a limited set of control parameters. Variants of this problem have been studied for a long time and several analytical and phenomenological models already exist in literature which can accurately predict the deformation dynamics of the immersed drop [57–59]. For this simulation, we use a Cartesian box which is wall-bounded in the vertical direction ($\hat{e}_z$) and fully periodic in the horizontal directions. The top and bottom walls move in the opposite direction parallel to each other with the same velocity to generate a laminar shear in the domain. A triangulated sphere as shown in Fig. 1(a) is positioned in the flow at a distance $0.5L_z$ from the walls (L_z is the gap between the walls).

The degree of deformation and orientation of a viscous drop in the presence of a velocity gradient depends on the Capillary number $Ca = \mu_f R\dot{\gamma}/\sigma$, where R, $\dot{\gamma}$, σ and μ_f are the drop radius, local strain rate, surface tension and fluid viscosity, respectively. For these simulations the viscosity ratio of the droplet and the carrier phase is set to 1, i.e. $\hat{\mu} = \mu_d/\mu_c = 1$. Since this simulation will be used to 'tune' the elastic constants to replicate a liquid–liquid interface, we assume that the immersed drop is very small in comparison to the distance between the walls and the immersed boundary forcing $\boldsymbol{f}$ in equation 1 is set to zero. This is done so that the immersed sphere only experiences the forces generated due to the laminar shear from the moving walls and not due to any wall effects. It also facilitates in quickly tuning the elastic constants due to the ease in setting up such a simulation. Here it is important to point out that while we choose the system of a neutrally buoyant deforming drop any other flow with known solutions can be used.

To estimate the ad-hoc surface tension value for any given spring network the following steps are undertaken. We first fix the Lagrangian resolution i.e. the number of vertices on the surface of a sphere and initialize a spherical drop under a given shear rate $\dot{\gamma}$ with a set of elastic constants. For the first set of elastic constants, k_e and k_a are fixed to large values in comparison to k_b thus resulting in an extremely stiff drop. k_v is chosen to be much larger than the rest of all constants as this ensures incompressibility of the immersed drop. Once the first set of elastic constants are chosen the drop is allowed to deform under the action of the velocity gradient $\dot{\gamma}$ according to the potential approach described in the previous section. If the final state of the drop is close to spherical, the elastic constant k_e and area constant k_a are reduced simultaneously which reduces the overall stiffness resulting in deformation of the spring network. Here it is important to note that if both k_v and k_a are fixed to a large value, which would imply conservation of both the volume and total area, the triangulated sphere would represent a vesicle. To represent a drop, both k_e and k_a are reduced to a low enough value such that the drop deforms approximately into an ellipsoid as shown in Figs. 3(a), (b). The initial and final states of the triangulated sphere shown in Fig. 3(a),(b) are for two different Lagrangian resolutions i.e. the spheres are discretised using 320 and 1280 triangular elements (faces), respectively.

We now give an estimate of the elastic constants used for the following analysis. For any given spring-network with a prescribed elastic modulus and mass distribution, the ratio $k_e l_e^2$ (l_e is the mean edge length on the sphere) should be independent of the number of triangular elements [49]. For the sphere shown in Fig. 3(a), $k_e l_e^2 = 3 \times 10^{-3}$ and $l_e/L = 1.5 \times 10^{-2}$. The bending constant is set as $k_b = k_e/10.0$ while the volume constant is set to a very large value; in this case $k_v = \alpha_v k_e$ where typically $\alpha_v \sim 10^4 - 10^5$ is sufficient to ensure that incompressibility of the sphere is ensured. The area constant k_a is set equal to that of the elastic constant. As discussed previously, when simulating other systems (for example vesicles) which also require surface area conservation along with volume conservation the area constant k_a is also set to relatively large value similar to the volume constant k_v. On changing the Lagrangian resolution the elastic constants should also be scaled in a way that the product $k_e l_e^2$ remains unchanged. This ensures that the physical properties of the spring network remains similar and is independent of the resolution used. The time step for the following simulation is set to $\Delta t = 10^{-4}/\dot{\gamma}$ ($\dot{\gamma}$ is the imposed strain rate) and the convergence criteria is such that the change in the total surface area of the sphere in successive time steps is less than 10^{-6}.

Once the deformation of the sphere has reached a steady state under a laminar shear flow, we compute its semi-major axis (L) and semi-minor axis (B). Next, we use a phenomenological model proposed by Maffettone and Minale [59] (hereafter called 'MM' model) to estimate the Capillary number Ca for which a neutrally buoyant immersed drop would have the

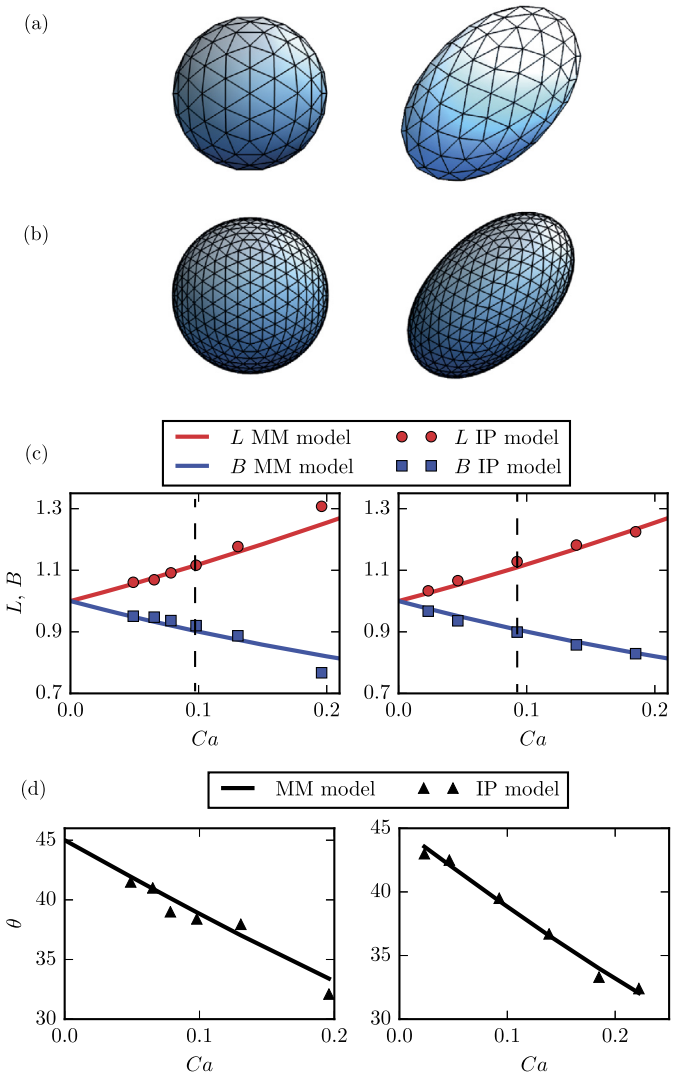

Fig. 3. Deformation of a neutrally buoyant drop in a laminar shear flow using the IP model. Lagrangian resolution of (a) $N_{\text{faces}} = 320$, (b) $N_{\text{faces}} = 1280$. In both cases, the viscosity ratio is set to $\hat{\mu} = 1$. (c) Comparison of the semi-axes lengths versus the Capillary number; (b) Comparison of the angle formed by the major-axis in the shear plane with the velocity direction.

same final state under similar flow conditions. The model proposed by MM [59] predicts the deformation of a drop in an arbitrary velocity field under the assumption that the drop is ellipsoidal in shape. For a simple flow field such as a laminar shear flow the model can be analytically solved to give the steady state values of the semi-major (L) and semi-minor axis (B) of the deformed drop as given below.

$$L^2 = \frac{f_1^2 + Ca^2 + f_2 Ca\sqrt{f_1^2 + Ca^2}}{(f_1^2 + Ca^2)^{1/3}(f_1^2 + Ca^2 - f_2^2 Ca^2)^{2/3}} \tag{17}$$

$$B^2 = \frac{f_1^2 + Ca^2 - f_2 Ca\sqrt{f_1^2 + Ca^2}}{(f_1^2 + Ca^2)^{1/3}(f_1^2 + Ca^2 - f_2^2 Ca^2)^{2/3}} \tag{18}$$

In equations (17) and (18), f_1 and f_2 are constants which depend on the viscosity ratio ($\hat{\mu}$) and Ca is the Capillary number.

$$f_1 = \frac{40(\hat{\mu} + 1)}{(2\hat{\mu} + 3)(19\hat{\mu} + 16)} \qquad f_2 = \frac{5}{2\hat{\mu} + 3} \tag{19}$$

This model has already been used in other studies; for example to predict hemolysis of red blood cells [60] and also deformation and orientation statistics of drops in turbulent flows [61,62]. Additionally, experimental studies have shown that under moderate deformations the steady-state droplet shape can be very well described by an ellipsoid [63,64].

In Fig. 3(c), (d) we plot the analytical solutions (MM model – solid lines) in the form of the lengths of the semi-axes and the orientation angle of the major axis (corresponding to the axes with length L) with the stream-wise direction versus the Capillary number. Using this as a reference, we check the position of overlap of the semi-axes lengths computed through the IP model with the MM model to estimate the corresponding Capillary number. This match is shown through a vertical dotted line in Fig. 3(c) and since the flow configuration such as drop radius, viscosity and shear rate are already fixed in the simulation, this Capillary number can be directly used to estimate the ad-hoc surface tension value for the chosen elastic constants. The left and right panels in Fig. 3 correspond to different Lagrangian resolutions. The important point to observe here is the reasonably good match between the semi-axes lengths computed from the IP model and the MM model. Small differences in the semi-axes lengths could arise due to multiple reasons, (i) lack of sufficient Lagrangian resolution since in the IP model the surface of the sphere is discretised using markers (ii) MM model assumes a perfectly shaped sphere which deforms into an ellipsoid while the IP model has no constraint of deforming into an ellipsoid (iii) the elastic constants would need further tuning.

Next, we keep the elastic constants the same and change the Capillary number which can be done by either changing the shear rate or the viscosity of the fluid. As shown in Fig. 3(c), (d) again the semi-axes lengths computed from the IP model agree reasonably well with the analytical solutions from the MM model. This shows that the ad-hoc surface tension computed by fitting the results from a single simulation using the IP model with MM model is reliable to extend the approach to other flow conditions. A good agreement with the MM model is found also for the orientation angle of the semi-major axes as shown in Fig. 3(d). At higher Capillary numbers ($Ca = 0.2$) there is some difference found in the lengths computed from the IP model as compared to MM model (left panel of Fig. 3(c)). However, this is just an effect of the Lagrangian resolution and can be corrected by increasing the number of vertices on the surface of the sphere as is seen in the right panel of Fig. 3(d).

3.2. Dynamics of a liquid–liquid interface deforming in cross-flow

In the previous subsection we demonstrated that by tuning the elastic constants for a single flow configuration to compute an ad-hoc surface tension, the IP model can be used to reliably simulate a neutrally buoyant drop deforming in a laminar shear flow. We now move on to simulating a more dynamic problem where the interface is strongly linked to the local flow conditions. In order to do this we take the same test case as done by Schwarz et al. [14] and have a drop immersed in a cross flow and compute the mean shape arising from the resulting flow conditions. For such a flow, the aspect ratio of the deforming drop depends strongly on the Weber number $We = \rho_f U_{\text{ref}}^2 d_{\text{eq}}/\sigma$, which is the ratio of inertia forces acting on the drop in comparison the surface tension forces (for more details see the review by Loth [65]). The cross-stream set up in the domain is influenced by the interface of the spherical drop leading to the development of a boundary layer on the drop surface and a corresponding wake.

The computational domain is taken to be of size $L = (10, 5, 5)d_{\text{eq}}$, d_{eq} is the diameter of the drop in its initial spherical shape. The spherical drop is triangulated with $N_v = 2562$ nodes and is placed at $\boldsymbol{x} = (0.5, 0.5, 0.5)L_z$. The vertical direction ($\hat{e}_z$) is wall bounded with stationary free-slip walls; $\hat{e}_y$ direction is periodic in nature and a uniform flow of $\boldsymbol{U} = U\hat{e}_x$ is imposed in the $\hat{e}_x$ direction.

The control parameters for such a problem are the Reynolds number, $Re = Ud_{\text{eq}}/\nu_f$ and the Weber number, $We = \rho_f U_{\text{ref}}^2 d_{\text{eq}}/\sigma$. The response of the system can be measured through the quantification of the wake of the drop and also through the morphology of the drop. The combined action of the dynamic pressure acting on the faces of the drop and the shear stresses generated from the boundary layer development on the surface of the drop leads to its deformation. In Fig. 4 we show the wall-normal component of the velocity field (u_z) and the corresponding deformed drop represented through the triangulated spring network. The two snapshots shown in Fig. 4 are at two different instants showing the starting up phase and the deforming phase. To quantify the shape of the immersed drop we compute the mean aspect ratio of the bubble measured as the ratio of the lengths of the drop bounding box in the wall-normal and stream-wise directions i.e. $X = l_z/l_x$, where X is the aspect ratio and l_z, l_x are the lengths of the box surrounding the deformed drop in the $\hat{e}_z, \hat{e}_x$ directions, respectively. In Fig. 5 we plot the inverse of the measured aspect ratios of the deformed drop versus the corresponding Weber number and compare it with experimental data from multiple measurements [65]. For these simulations the Reynolds number is fixed to $Re = 150$ and the Weber number is changed by modifying the elastic constants for each simulation. A very good match is found between the aspect ratios computed from the IP model and the several experimental measurements of drop shapes found in literature. These simulations further show that the IP model can be reliably used to simulate deformation in liquid–liquid interfaces under given flow conditions.

4. Dynamics of the left heart ventricle

We now move on to simulating the flow inside the left ventricle of the heart where the motion of the ventricle and the valves are fully coupled to the flow dynamics. The results from the numerical simulations are compared against ad-hoc experiments where the ventricle is made up of silicone rubber.

The various structures used for this simulation are shown in Fig. 1 and it is important to note that each structure is made up of a different material i.e. each material has a different elastic property. The left ventricle and the natural mitral valve can move and deform based on the local flow; the leaflets of the mechanical mitral valve, while rigid in shape, can

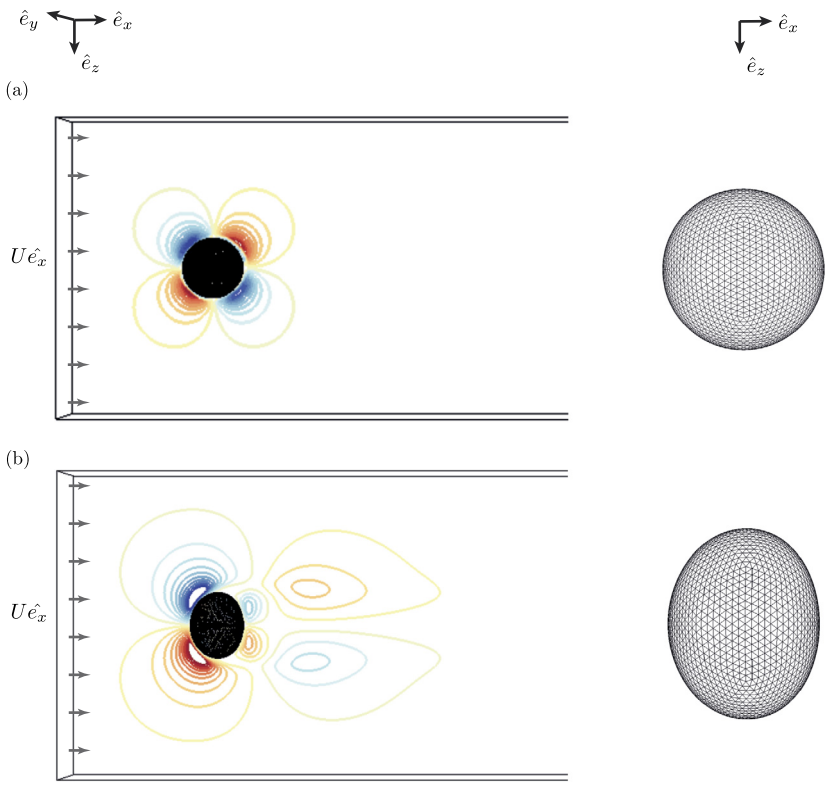

Fig. 4. Left panels show the contours of velocity in $\hat{e}_z$ direction along with the deforming drop at two different time instants. Right panels show the corresponding drop in the form of the deformed triangulated spring network. The Reynolds number of the flow based on the initial drop diameter is set to $Re = 150$, while the elastic constants chosen correspond to a Weber number $We = 2$.

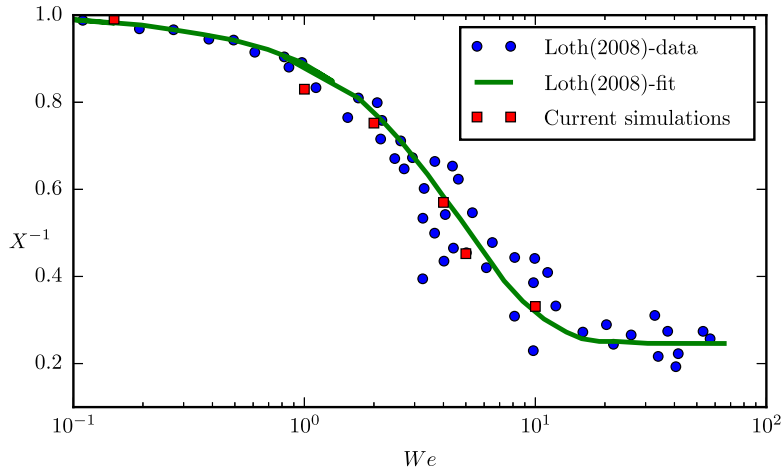

Fig. 5. Comparison of the inverse aspect ratio (X^{-1}) of the deformed drop versus Weber number at $Re = 150$ with data from Loth [65] for contaminated drops or bubbles.

move depending on the forces acting on their faces and more specifically on the moments of the pressure and viscous forces about the hinges of the leaflets; the channels for the aortic and mitral valves are completely rigid, fixed in space and provide a passage for the influx and outflux of the flow. The aortic valve is not simulated explicitly in these simulations but only through an opening/closing mechanism that is imposed by the immersed boundary depending on the phase of the cycle. While this has been done to limit the computational effort, it has no major consequences on the results because we are only interested in the ventricular flow and the aortic valve influences flow mainly in the ascending aorta. The dynamics of the aorta could affect the ventricular flow because of the timing of the opening and closure of the aperture, but it is

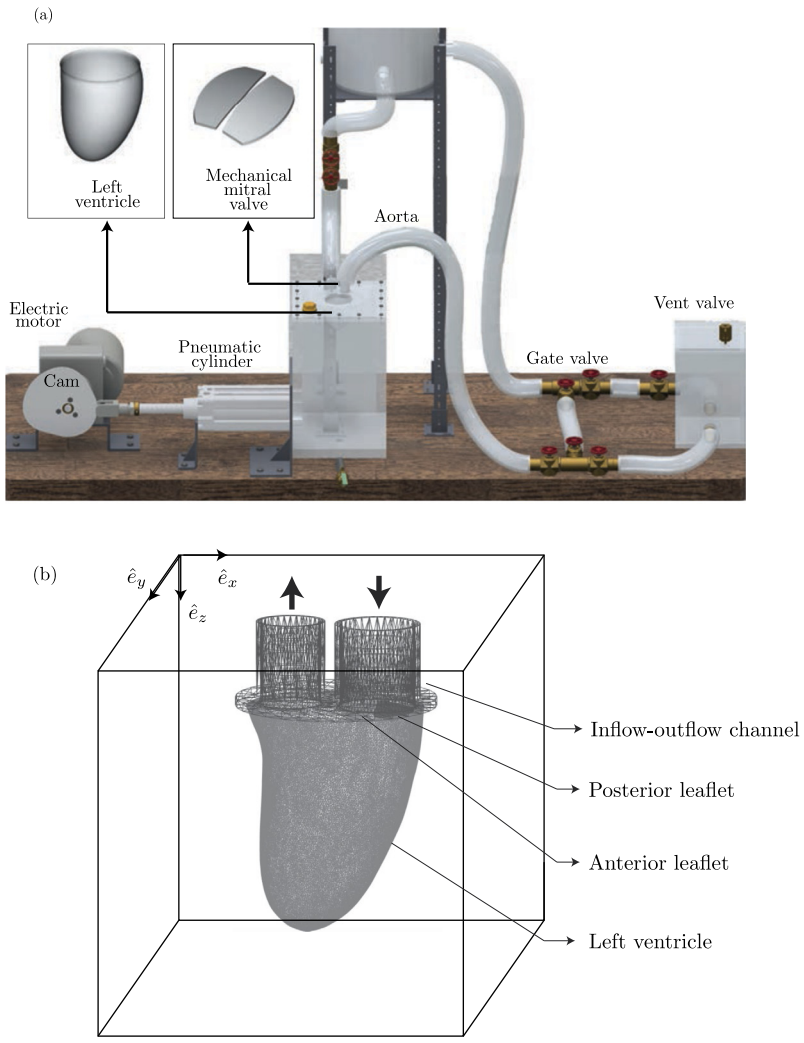

Fig. 6. (a) CAD rendering of the experimental setup built for validating the numerical approach. (b) Cartesian computational box with the inflow–outflow channel, mitral valve and the left ventricle. Individual components are shown in Fig. 1.

driven by the impedance of the circulatory system downstream and its simulation is much more complicated and out of scope of this paper.

In reality, the configuration of the left ventricle is determined by the dynamics of the myocardium contraction and relaxation along with the deformation of the valves and vessel walls. The complete structure adjusts to the forces induced by the hydrodynamic loads (pressure and shear stresses), body forces, internal damping and the internal elastic forces. In our simulations, the flow into the ventricle is governed through an inflow–outflow channel rather than a myocardium contraction to facilitate comparison with experiments. Similar to the experiments, the ventricle is assumed to be made of a homogeneous material i.e. silicone rubber. With minor modifications the IP approach works equally well for hyper elastic or inhomogeneous (orthotropic) materials as discussed in section 2.

4.1. Experimental and numerical setup

In Fig. 6(a) we show a CAD rendering of the experimental apparatus used to replicate the dynamics of the left ventricle with a mechanical mitral valve and results from this will be used to validate the numerical model. An electric motor is used to drive a cam which imposes a prescribed displacement in time of the pneumatic piston/cylinder. The cylinder is directly linked to a Plexiglass box which is transparent and allows for the observation of the evolution of the left ventricle model inside. The time law imposed by the pneumatic cylinder is replicated by the fluid in the tank in which the left ventricle is immersed and is the only deformable element. Additionally, the evolution of the flow rate imposed by the motion of the cylinder is captured versus time and this is used as a boundary condition in the numerical simulations (Fig. 6). This is shown in Fig. 7 where we plot the flow rate versus time. As can be seen in Fig. 7, the first part of the cycle has one

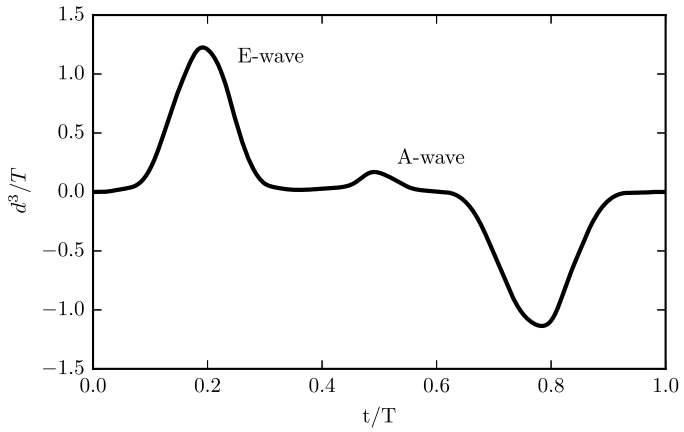

Fig. 7. Flow rate (scaled by the diameter d of the opening to the mitral valve) versus time (normalised using time T required for one cardiac cycle). The flow rate regulates the expansion (positive flow-rate or diastolic phase) and relaxation (negative flow-rate or systolic phase) of the ventricle.

strong peak (E-wave) and a secondary weak peak (A-wave) which is the result of the shape of the cam. The shape of the cam can be modified to achieve any desired flow rate profile. In the case shown here the ratio of amplitudes of E-wave to the A-wave is approximately 20:3. The profile of the cam is chosen in such a way that the flow rate resembles that of an inefficient and failing left ventricle and is generally observed in old people or heart patients. In a healthy condition, the time evolution of the flow rate versus time is similar to that shown in Fig. 7 but with an amplitude ratio of A-wave to E-wave of approximately 0.5. The efficiency/healthiness of the ventricle can also be quantified using ejection fraction (EF) which quantifies the pumping efficiency of the ventricle and is calculated as $EF = 100(V_{\max} - V_{\min})/V_{\max}$; $V_{\min}$ and $V_{\max}$ being the minimum and maximum values of the volume of the left ventricle, respectively during the cycle.

The left ventricle is transparent and made up of silicone rubber, fixed to the upper surface of the box by a rigid plate and consists of a mechanical mitral and aortic valves. The fluid (deionized water here) inside the left ventricle is pumped into the aorta which then flows into the hydraulic circuit composed of two branches. In one, the windkessel, there is a box connected in series to simulate the vascular capacitance while there are gate valves to regulate the impedance of the systemic circulation or to exclude one branch or another. The fluid after passing through the hydraulic circuit returns into the ventricle through the duct and a new cardiac cycle starts. In order to compare experimental measurements and numerical simulations we make use of Particle Image Velocimetry (PIV) measurements [66] where the fluid is seeded with tracer particles (10 μm diameter pine pollen) and illuminated by a laser sheet. The motion of the particles is captured using a high-speed camera and a robust algorithm is used to compare image windows in subsequent frames and estimate the velocity field in the flow on a regular grid.

In Fig. 6(b) we show the computational domain and the setup of the complete left ventricle along with the mechanical mitral valves and the channels for the aortic and mitral valves. The geometry of the structures, the material properties and the boundary conditions have been chosen to replicate the experimental conditions as close as possible. The channels connected to the mitral and aortic valve perform the function for allowing the influx/outflux of the fluid into/from the ventricle. Since we use the IBM formulation for representing any immersed body, the whole domain is filled with a single fluid. The domain is periodic in all the directions $\hat{e}_x$, $\hat{e}_y$ and while it is confined in the $\hat{e}_z$ direction it allows for inflow–outflow boundary conditions on selected regions. The flow rate evolution shown in Fig. 7 is used as the boundary condition on the inflow/outflow channels and is linked to an amplification factor that regulates its amplitude, i.e. the higher the amplitude the higher the ejection fraction of the left ventricle. In the numerical simulations we set the value of EF to 30% which is what is imposed in the experiments in order to study the flow in a severe failing left ventricle. After performing grid independence tests, a resolution of $150 \times 150 \times 150$ was chosen for the Eulerian field. The surface of the ventricle is discretised with 51142 triangular elements; mechanical valves with 2578 elements and the natural valves with 3794 elements each. Both experiments and the simulations are performed in dynamic similitude with a real left ventricle i.e. since the dimensions in the experiments and simulations are set to a 1:1 ratio in comparison with a real left ventricle and water is four times less viscous than blood, the total system is pulsated four times slower to maintain the same Reynolds number. The characteristic Reynolds number in the flow based on the mitral orifice diameter and maximum inflow velocity is around 5000. For a nearly isotropic membrane, the elastic constants used in the spring network can be computed using the model by van Gelder [49] as $k_e = Eh(A_1 + A_2)/l_o^2$ and $k_b = 2B/\sqrt{3}$; E and B are the Young's modulus and bending stiffness of the membrane, respectively while h is the local membrane thickness and A_1, A_2 are the areas of the triangles sharing an edge with initial length l_o. The experiments discussed in the paper use rubber silicon as the membrane material and the elastic constants for the simulations are directly computed from the physical properties of rubber silicon.

We first look at the large scale flow structures created inside the ventricle. In Fig. 8 we plot the instantaneous snapshots of the flow velocity vectors in the mid-Y plane at certain time instants. All the left panels correspond to numerical simulations while the right panels show the measurements from the PIV experiments. It can be seen that the large scale flow

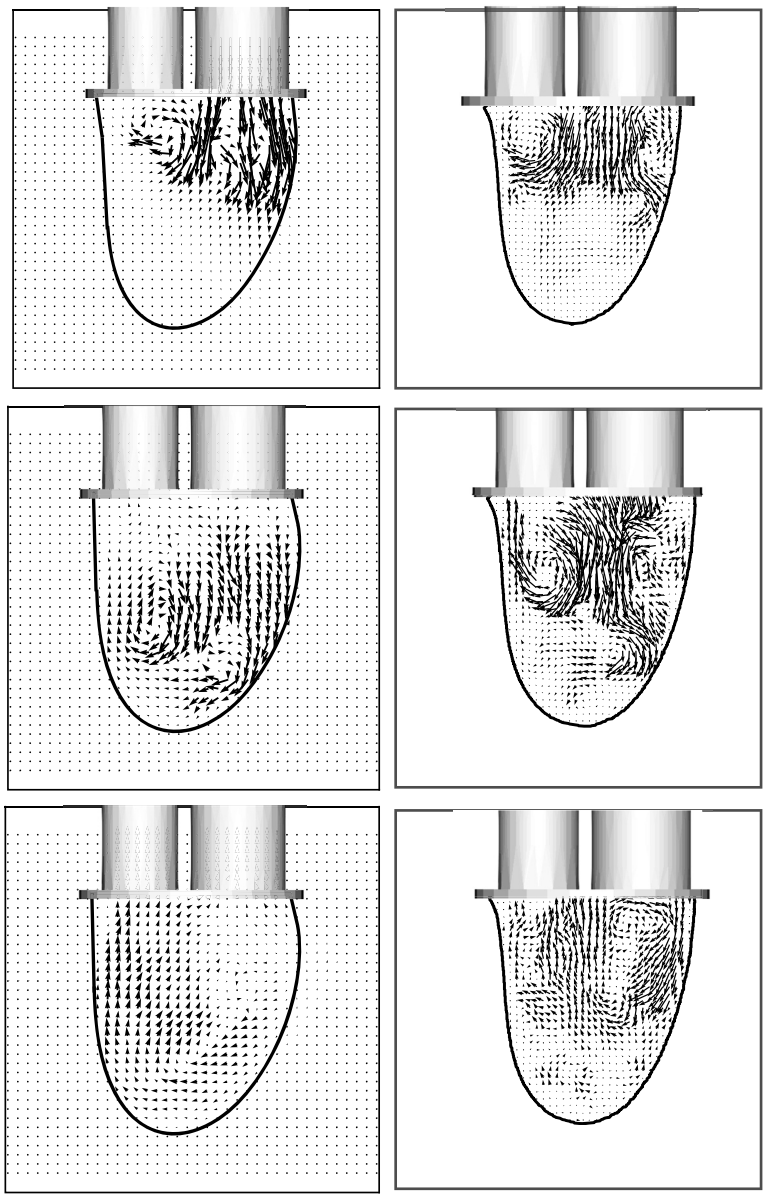

Fig. 8. Snapshots of the flow inside the ventricle during the diastolic phase in the cardiac cycle. (Left panels) Numerical simulations; (right panels) Experimental measurements.

dynamics can be reliably captured in the numerical simulations as compared to the experiments. During the initial part of the cardiac cycle i.e. the diastole, the jet from the mitral valve passes through the prosthetic mechanical leaflets which starts to open. The flow over the two leaflets results in the propagation of two vortices into the ventricle, one close to the left wall and the other in the centre. The two vortices are directed towards the apex of the ventricle, but since in both the simulations and experiments we reproduce the dynamics of a failing left ventricle the vortices soon dissipate into small scales and the mitral jet is not able to penetrate down to the apex and wash out the stagnant fluid. This is shown more clearly later.

We now compare the mean position of the left ventricle in the $\hat{e}_x$ and $\hat{e}_z$ directions to further validate the dynamics of the deforming ventricle from the numerical simulations. This is shown in Fig. 9 where the mean position x/d and z/d is plotted against time. The fluctuations in the mean x-position of the left ventricle occurs due to its asymmetrical geometry with respect to the mitral jet. The fluctuations seen in the numerical simulations are entirely physical as we consider full fluid–structure interaction without any kinematic models to govern the motion of the ventricle. The oscillations seen in the numerics cannot be fully resolved in the experimental measurements. The positions obtained from the numerical simulations have reasonably good agreement with its experimental counterpart except from small oscillations which can-

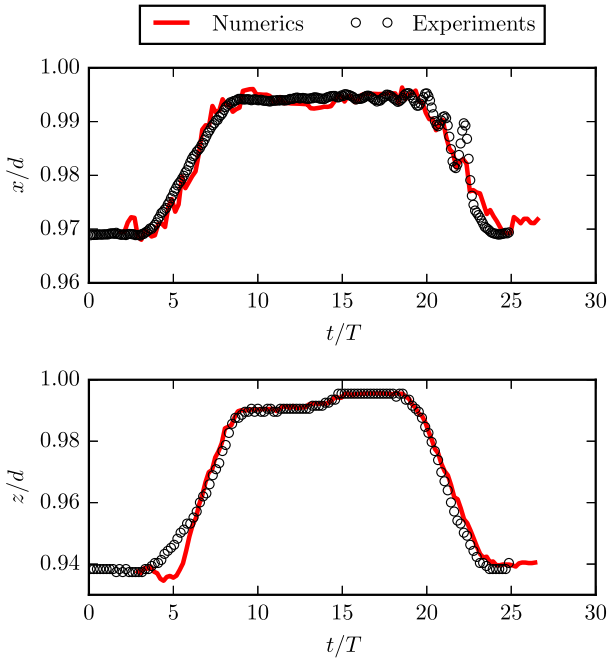

Fig. 9. Comparison of the mean position of the left ventricle in the $\hat{e}_x$ and $\hat{e}_z$ directions versus time.

not be captured in the experiments. This shows that not only the large scale flow structures, but also the dynamics of the deforming left ventricle which is modelled using the interaction potential approach can be simulated with reasonable accuracy.

4.2. Mechanical and natural mitral valve

An important element affecting the dynamics and nature of the flow is the presence of the mitral valve. To understand the effect of the mitral valve we couple our computational model of the left ventricle with two kinds of mitral valve (i) prosthetic mechanical valve (ii) natural mitral valve. While the mechanical valve is structurally rigid the natural valve is similar to a flexible membrane and can deform based on the local flow conditions (cf. Fig. 1). In Fig. 10 we show both the mechanical and natural valve during their initial state and when they are close to being fully open. Here we would like to again emphasize that since the numerical set up uses a fluid–structure interaction approach, the valve dynamics are solely determined by the hydrodynamic loads and any geometrical constraints set up by the user. The panels on the right in Fig. 10 show a clear difference in the shape of the ventricle. The shape of the ventricle depends heavily on the hydrodynamic loads exerted on it from the fluid inside it. The mechanical and natural mitral valves lead to different flow structures inside the ventricle and thus a different shape of the ventricle. We now show the difference in flow structures arising from the different valves used.

First, we consider the case of a prosthetic mechanical mitral valve which in a sense obstructs the flow through the mitral orifice. For the dynamics of the full valve, we allow each leaflet to rotate around a fixed axis which is symmetric about a plane situated in the centre of the mitral orifice. In Fig. 11 we show the dynamics of the leaflets which go in opposite directions and close asymmetrically since the backward flow induced by the systole comes from different regions of the ventricle for the anterior and posterior leaflets. The opening phase starts at the beginning of the diastole as the flow starts accelerating and finishes before the end of the flow acceleration when the fully open position is reached. The closing phase starts when the flow rate reaches its peak and ends when the minimum negative value of the flow rate function is achieved, thus positioning the leaflets in the fully closed position.

In Fig. 12 we show instantaneous velocity fields during the diastolic phase of the cardiac cycle with both a prosthetic mechanical and natural mitral valve. In the case of mechanical valves (top panels of Fig. 12), the leaflets start rotating during the early opening phase and destabilise the mitral jet. In the bottom panels of Fig. 12 we show the flow structure in the presence of a natural mitral valve which is also made up of two leaflets but has different dynamics due to the inherent deformability of the natural valves. In the presence of the mechanical valve, the flow is split into three different jets thus causing high vorticity regions in the wake of the valve. This results in the mitral jet not reaching the bottom of the ventricle as desired. It is evident that the disturbance generated by the mechanical leaflets destabilizes the mitral jet, creating vortex rings thus further decreasing its capability to penetrate the ventricular region. The flow soon degenerates into small scales that are dissipated during the diastatic phase of the cycle. Unlike the prosthetic mechanical valves, the natural valves can

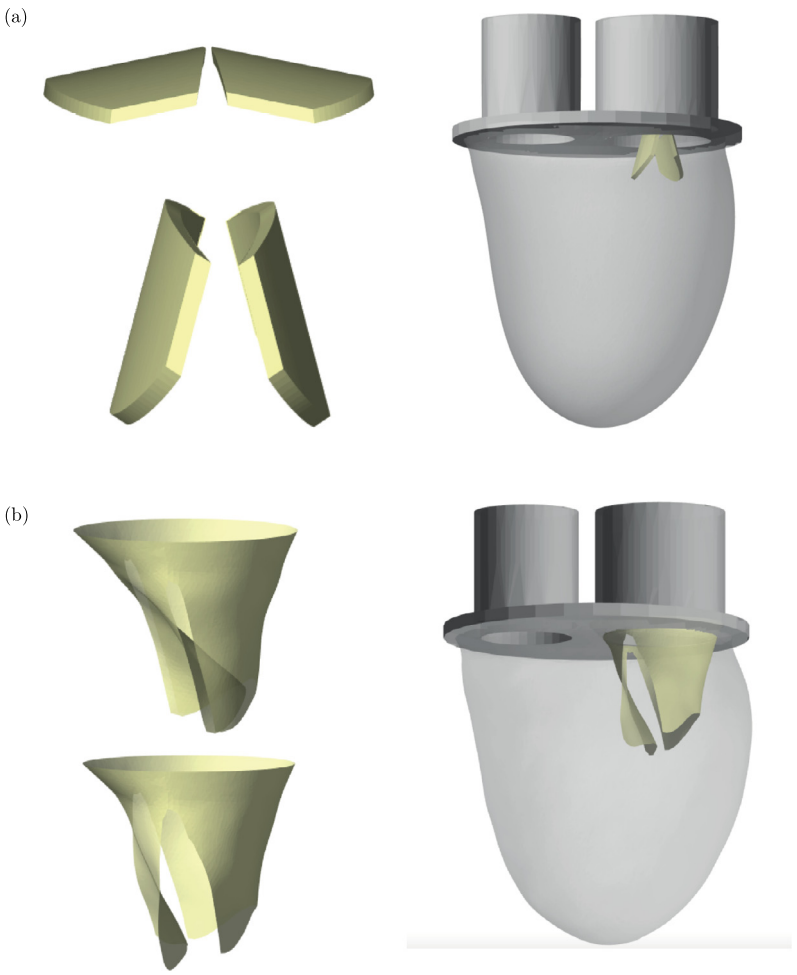

Fig. 10. Left panels show snapshots of (a) Mechanical and (b) Natural valves at two different time instants in the diastolic phase of the cardiac cycle. The right panels show the full set up of the ventricle along with the valves and the inflow/outflow channels.

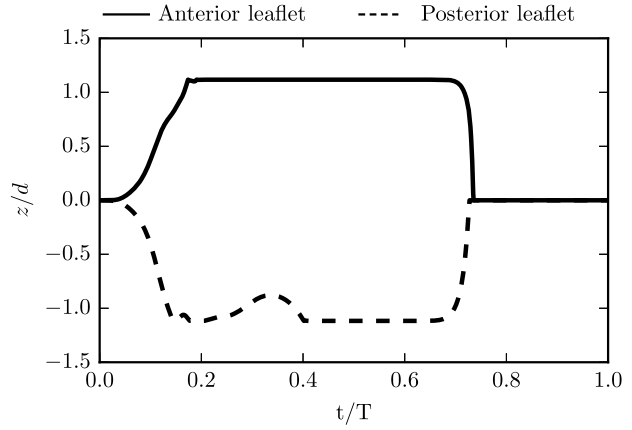

Fig. 11. Axial position of the centre of mass of prosthetic mechanical leaflets for a healthy left ventricle in a single cardiac cycle. The solid line and dashed line represent two different leaflets. d is the diameter of the mitral orifice, while T is the time taken for one full cardiac cycle.

deform based on the local hydrodynamics forces allowing for a much smoother flow of the mitral jet into the ventricle. Due to this the natural valves evolve differently resulting in a different flow structure in the ventricle which reaches the bottom of the ventricle which is a desired flow condition.

Fig. 12. Snapshots of the flow inside the ventricle at the two different time instants (left and right panels) in the diastolic phase of the cardiac cycle. Top panels show the ventricle with a mechanical natural valve while the bottom panels show the ventricle with a deformable natural valve. The colour represents the iso-surface of the velocity magnitude of the flow inside. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

From the discussion of Fig. 12 it is clear that the behaviour of the flow inside the left ventricle depends strongly on the kind of mitral valve used. Overall, we have been able to show that the complete dynamics of the left heart ventricle with either mechanical or biological valves can be simulated reliably using IBM coupled with an interaction potential approach for deformation.

5. Parallel performance

In this section we describe the parallelisation strategy implemented and the data structures required for parallelising the algorithms described in previous sections. For dealing with a suspension of spherical particles, Uhlmann [67] proposed a 'master' and 'slave' strategy, where each particle is allocated an individual 'master' processor which is responsible for all the computations related to it. Additional 'slave' processors may be allocated to help the 'master' processor. Wang et al. [68] employ a 'gathering-scattering' strategy where a single master processor is responsible for the computation of the Lagrangian force on the immersed bodies and advecting them and this information is scattered to the slave processors which solve the Navier–Stokes equations in parallel. While both parallelisation approaches have been shown to produce reasonable performances, there exist some drawbacks and challenges. The strategy implemented by Uhlmann [67] requires continuous exchange of control on the Lagrangian mesh by the processors which may lead to a complex programming environment. The approach of Wang et al. [68] eliminates this issue leading to a simple structure of the code while increase in the memory usage on the master processor and data transfer between the master and slaves are some hurdles. In this work, we propose a different easy to implement parallelisation approach for the IBM where the information of all triangle nodes is present with all processors. But the computation required for each Lagrangian node/structure is performed only by specific processors depending on the type of computation that needs to be performed. In other words the allocation of processor for the IBM depends on the task that needs to be performed which results in a task-based parallelism for the FSI-IBM computation.

We first describe in brief the parallelisation strategy employed for the flow solver and later explain the data structures and parallelisation implemented for the FSI-IBM in the appendix. For the flow solver we employ domain decomposition and split the Cartesian box into slabs i.e. 'one-dimensional slab' parallelisation. It is also possible to use a Cartesian box

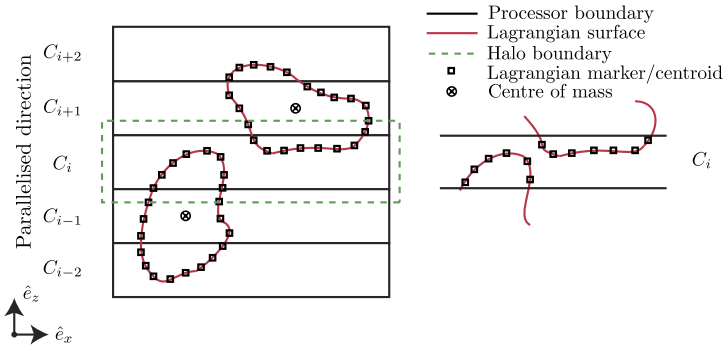

Fig. 13. Schematic of two bodies immersed in a flow. Flow solver is 1D slab parallelised. $\hat{e}_z$ is the wall-normal direction.

decomposed with 'two-dimensional pencil' parallelisation as shown in [33]. In such a fluid solver, the viscous terms are computed explicitly in the periodic directions to take advantage of the reduced ALL-to-ALL communications. This may prove fatal dispersed deforming bodies in the flow since we also need to resolve the boundary layers over the immersed bodies which have dominant velocity gradients. With pure MPI, slab based codes already give good scalability up to 10^3 cores and this can be further enhanced to 10^4 cores by having a hybrid MPI+OpenMP type parallelisation which is already sufficient to tackle a large scale of problems.

In addition to the slabs, each processor needs to store information from the neighbouring processors which would be required for computing the derivatives and is stored in what is called as a 'halo/ghost' layer. Since the flow solver employs a second-order finite difference spatial discretisation at most one halo layer is required on each side of a slab for single phase flows. However, as we explain later when this solver is coupled with a FSI-IBM solver for finite-size bodies which makes use of MLS interpolations, multiple halo layers become necessary. It is important to keep in mind that an unrestricted increase in the number of stored halo layers would automatically result in an increase in the communication time which may deteriorate the overall performance of the code. For the MLS interpolations which need a support domain of 27 ($3 \times 3 \times 3$) Eulerian points at most 3 halo layers are necessary.

Consider two arbitrarily shaped deformable bodies immersed in a flow as shown in Fig. 13, where the squares represent the Lagrangian markers/centroids of the triangular elements. The allocation of Eulerian slabs to each processor is straight forward as the Eulerian mesh stays fixed in time and this is done at the start of the simulation. For the flow solver each processor with identity `myid` is allocated the task of solving equation (1) on a slab of `[1:N_1,1:N_2,N_3_start:N_3_end]`. Given below are the steps undertaken to complete one full time step of the simulation. As given below there are four major steps and multiple sub-steps involved in completing one full iteration. The steps shown here are applicable for a loosely coupled approach which has been used for the simulations in this paper; details on the strongly coupled approach are elaborated in the paper by de Tullio and Pascazio [21].

1. Compute the indices of all markers/centroids on the Lagrangian mesh relative to the Eulerian mesh.
2. Compute the properties of the Lagrangian mesh, i.e. surface areas and normals of each face of the Lagrangian mesh.
3. Compute flow configuration along with IBM forcing i.e. all three sub-steps of RK3 integration.
   (a) Compute intermediate fluid velocity under the RK3 framework.
   (b) Interpolate velocity on the centroids of the Lagrangian mesh using MLS interpolation.
   (c) Communicate the forces in the halo cells to neighbouring processors.
   (d) Correct intermediate velocity using the MLS-interpolated force.
   (e) Solve pressure correction equation and compute the pressure and solenoidal velocity field.
4. Compute external and internal loads on the immersed body.
   (a) Compute the external loads which is the sum of pressure and viscous forces on each face using MLS interpolation.
   (b) Sum up external loads on all faces across all processors.
   (c) Compute internal loads which are derived from the potentials described in section 2.
   (d) Sum up internal loads across all processors.
   (e) Update the nodes of the triangles using Newton's law of motion.

The details on the data structures and the pseudo code for the parallelisation of the IBM-IP solver is given in the Appendix.

5.1. Scaling

In Fig. 14, we show the computational performance of the parallelisation strategy discussed in the Appendix. These simulations were performed on the thin nodes of the Dutch supercomputing facility 'Cartesius' where each node is composed of 2×12 core 2.6 GHz Intel Xeon E5-2690 v3 CPU's. As can be seen from the plots in Fig. 14 strong scaling is achieved up

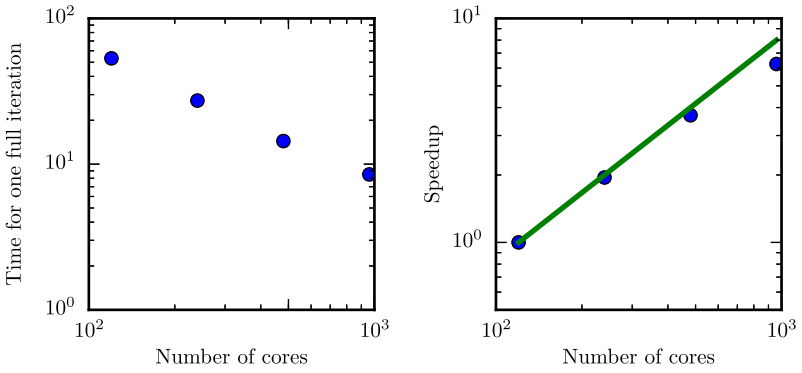

Fig. 14. (a) Scaling plot showing the time step for one full iteration of the solver versus the number of cores used. (b) Corresponding speed up versus the number of cores.

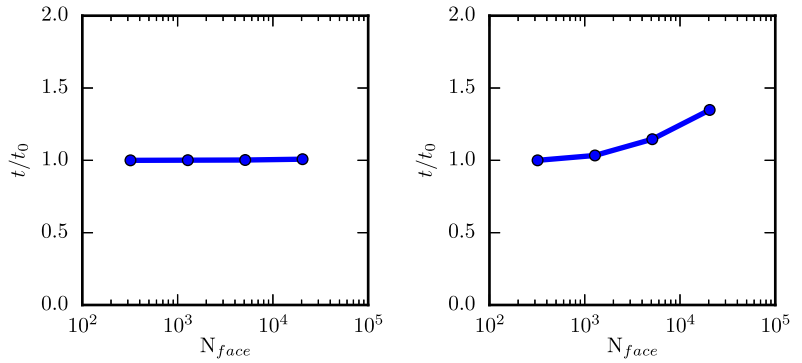

Fig. 15. Time taken for one full iteration normalised using the time taken for the first data point versus the total number of triangular elements or faces immersed in the domain. In the left panel the immersed bodies are stationary and fixed in shape while in the right panel the bodies can both move and deform.

to 1000 cores. The Eulerian resolution was set to $720 \times 720 \times 3840$ with a total of 25000 spherical particles each discretised using 320 faces, i.e. a total of 8 Million Lagrangian markers were simulated simultaneously. Due to global storage of the geometrical features and meta data of the Lagrangian markers, the memory consumption by the IBM-IP part of the flow solver for such simulation is approximately 600 MB.

For a given flow solver, the costliest steps in the IBM part of the solver are the ones involving MLS interpolation since each interpolation requires the construction of multiple coefficient matrices and a subsequent inversion of a 4×4 matrix (in 3D). For each Lagrangian marker (centroid) immersed in the flow two MLS interpolations are required; one at the Lagrangian marker itself to compute the IBM forcing and another at the position of the probe projected from the centroid which is used for the computing the value pressure and velocity gradients. Additional matrix operations are required for the velocity gradients since instead of the shape function we need to compute the derivative of the shape function [36]. On a single processor, increasing the number of Lagrangian markers by two times can result in a three to five fold increase in the simulation time. It is thus crucial to see how the parallel code performs with increase in the total number of Lagrangian markers or triangular faces.

In Fig. 15 we plot the non-dimensional time taken for one full iteration with increasing total number of faces for two different types of simulations. The time is normalised using the time taken for the first data point i.e. $N_{\text{face}} = 320$. For these simulations the Eulerian grid is kept fixed to $120 \times 120 \times 720$ and a total of 120 cores were used. In the left panel of Fig. 15, the immersed bodies are kept fixed in position and shape i.e. the computation of the structural solver is fully eliminated. Such simulations are useful to compute the hydrodynamic forces acting on stationary bodies with a mean flow imposed in the domain. As can be seen, with increase in the number of faces there is negligible increase in the computational time. In comparison to simulations involving moving and deforming objects, IBM simulations with stationary and fixed bodies require only one MLS interpolation and this is the reason for the negligible time increase. On the right panel we show the increase in time for simulations involving both moving and deforming bodies. These simulations require an additional MLS interpolation at the probe and also the computation of shape function derivatives. For such simulations, a 100 times increase in the number of faces results in approximately 1.3 times increase in the computational time. This shows that the increasing cost of MLS interpolations on the Lagrangian markers can be offset by parallelising the task over multiple processors.

6. Summary and outlook

In this paper we have demonstrated the implementation of a finite-difference based flow solver capable of handling several deforming drops/bubbles where the deformation dynamics is computed through an interaction potential approach. An immersed boundary method based on moving least squares interpolation is used to enforce the interfacial boundary condition on the underlying flow while the deformation dynamics is computed through minimising the potential energy of a spring network spread over the surface of an immersed drop or a bubble. The unique feature of this approach in comparison to conventional techniques used to simulate immersed drops or bubbles is its computationally efficiency which allows us to scale up both in terms of the number of drops or bubbles in the flow and also the turbulence level in the surrounding fluid. In the first part of the paper we discuss briefly why the presented approach can replicate the deformation dynamics of immersed drops/bubbles and the estimation of ad-hoc elastic constants required in the model. By comparing with already existing analytical solutions and experimental measurements of standard configurations (deforming drop in a shear flow and cross flow), we have shown that such an approach is practical, reliable and self-consistent. In the second part of the paper, we have shown that the same potential approach with minor modifications to the governing equations can also be used to successfully simulate complex FSI problems such as the dynamics of the left heart ventricle with a mechanical or biological valve. In contrast to previous studies where the motion of the ventricle or the valves are imposed a priori through kinematic models, in this paper a full fluid–structure interaction simulation of the left heart ventricle was carried out on a single computing processor which further throws light on the efficiency of the deformation model. The results from the simulations have been validated with ad-hoc in-house experiments. While the interaction potential approach for deformation is computationally inexpensive, parallelisation is a necessary step to simulate large scale turbulent flows with several thousands of simultaneously deforming bodies. To this effect, in the last part of the paper, we present a parallelisation strategy which is easy to implement for any generic single phase fluid solver. With pure MPI based parallelisation we have shown that decent scalability is achieved up to 10^3 cores and can be further enhanced to 10^4 by including OpenMP directives.

An exciting feature of the IBM-IP solver presented in this work is the versatility of the algorithms used. In particular, the interaction potential approach combined with an immersed boundary based flow solver can be used to study a large class of cardiac hemodynamics problems with no pre-determined valve or ventricle dynamics. Additionally, the approach can also be used to study multiphase flows involving deformation of thousands of deforming drops and bubbles in highly turbulent flows. In our future work we are focusing on eliminating some drawbacks of the IBM used here and improving further the speed of such simulations. For example, wall-bounded turbulent flows need enhanced resolution in the Eulerian mesh near the walls to capture the boundary layers which might add a severe constraint on the Lagrangian resolution. In order to tackle this we are currently implementing a fast moving least squares implementation with adaptive Lagrangian mesh refinement to decouple the Lagrangian mesh into one for IBM forcing and another for deformation. Novel algorithms are also required to implement collision dynamics between the immersed bodies which will become crucial when several thousands of such bodies are interacting within a flow.

This work was supported by the Netherlands Center for Multiscale Catalytic Energy Conversion (MCEC), an NWO Gravitation programme funded by the Ministry of Education, Culture and Science of the government of the Netherlands and the FOM-CSER program. We acknowledge PRACE for awarding us access to FERMI based in Italy at CINECA under PRACE project number 2015133124 and NWO for granting us computational time on Cartesius cluster from the Dutch Supercomputing Consortium SURFsara. This work has also been partially funded within the PRIN project number 2012HMR7CF by Italian Ministry of University and Research.

Appendix A. Data structures and pseudo code for Lagrangian mesh parallelisation

The Lagrangian meshes shown in Fig. 1 are unstructured and are exported in the form of a GTS (GNU Triangulated Surface) data format which contains information about the spatial positions of the vertices of the triangular elements, the various vertices which are connected by edges and also the edges which constitute a face. Using this information we construct additional auxiliary arrays which will be required while computing the total force acting on the triangle nodes based on the potentials described in the section 2. The total number of vertices, edges and faces on a single immersed body is stored in `N_vert`, `N_edge`, `N_face`, respectively while `N_particle` is the total number of immersed bodies to be simulated and `N_edge_vert` is the maximum number of edges that any single vertex can be connected to. A brief overview of the required auxiliary integer arrays is given below.

1. `vert_of_edge[2, N_edge, N_particle]` : Contains pairs of vertices sharing a single edge.
2. `face_of_edge[2, N_edge, N_particle]` : Contains pairs of faces sharing a single edge.
3. `vert_of_face[3, N_face, N_particle]` : Contains the three vertices that constitute a single face.
4. `edge_of_face[3, N_face, N_particle]` : Contains the three edges that constitute a single face.
5. `vert_of_vert[N_edge_vert, N_vert, N_particle]` : Contains all the vertices that a single vertex is connected to.
6. `edge_of_vert[N_edge_vert, N_vert, N_particle]` : Contains all the edges that a single vertex is connected to.

7. `v1234[4, N_edge, N_particle]` : Contains all the four vertices that is contained in two faces sharing an edge.
8. `pind[3, N_face, N_particle]` : Stores the [`N_x,N_y,N_z`] indices of each centroid relative to the Eulerian mesh and tells us inside which Eulerian computational cell the centroid resides in. This array is updated every time step.
9. `bboxind[6,N_particle]` : Stores the indices of the bounding box of each immersed body.

In the first step, we compute the indices of all the centroids on every triangular element and store it in a global array `pind`. In addition to the axial index of every triangular element we also compute the mean axial index of every immersed body i.e for an immersed body `i` the mean axial index is `bboxind[1:3,i]=0.5*(max(pind[1:3,:,i]+min(pind[1:3,:,i]))`. In step 2, we compute the geometrical properties of the triangulated mesh (i.e. surface areas and normals of each triangular element). Both steps (1 and 2) are done by all processors (i.e. `MPI_COMM_WORLD`) on all immersed bodies and at the end of this operation every processor has information on all three indices [`pind(1:3,N_face,N_particle)`] of every centroid immersed in the flow, surface areas and normals of every triangular element.

For steps 3(a), 3(d) and 3(e) each processor performs all the operations required on its respective slabs. Step 3(b), which consists of interpolation using MLS and computing the IBM force has to be performed on the Lagrangian markers (centroids here) and this is done only on the centroids lying within the processors slab (cf. right panel of Fig. 13). This allocation is regardless of which immersed body it belongs to. This is achieved by first performing a check on the axial index of every centroid (stored in `pind[3,:,:]` and computed in step 1); for example, if the processor C_i is responsible for the slab `[1:N_1,1:N_2,N_3_start:N_3_end]` the following procedure is undertaken.

```
do i=1,N_particle
  do j=1,N_face

    if pind(3,j,i) >= N_start(myid).AND.pind(3,j,i) < N_end(myid)
        - Perform MLS interpolation around the centroid.
        - Compute IBM forcing.
    end if

  end do
end do
```

As explained in section 2, MLS interpolations require a support domain built from 3 Eulerian grid nodes in each direction. Thus the forcing computed from a centroid lying right next to a processor boundary would be stored in a halo layer and this is communicated to the neighbouring processors in step 3(c). Every processor adds the IBM forcing received from the halo cells of the neighbouring processors to the already existing forcing thus accounting for the forcing from the centroids lying on processor boundaries.

Step 4(a) involves computing the external forces on the immersed body (i.e. pressure and viscous forces) which are performed following the procedure described in section 2. The allocation of processor for computing the external processors is done in a similar manner to step 3(b). Here it is important to note that for centroids lying on the processor boundaries the probes may lie in the neighbouring processor. For example, a centroid belonging to processor C_i may have an axial index of `N_3_start` and the axial index of the corresponding probe would be `N_3_start-1`. Building a support domain around `N_3_start-1` would require information from [`N_3_start-2, N_3_start-1, N_3_start`] i.e. at least two halo layers need to be stored by each processor.

```
do i=1,N_particle
  do j=1,N_face

    if pind(3,j,i) >= N_start(myid).AND.pind(3,j,i) < N_end(myid)
        - Compute probe and build support domain around the probe.
        - Perform MLS interpolation around the probe.
        - Compute pressure and viscous forces on faces.
        - Distribute the forces from faces to nodes.
    end if

  end do
end do
```

In step 4(b), we reduce the external forces ($\boldsymbol{F}_{\text{ext}}$) over all the triangle nodes immersed in the flow. `MPI_ALLREDUCE` is used to perform this operation which results in all the processors having information on the external forces acting on all triangular nodes. Here it is important to remember that the efficiency of the `MPI_ALLREDUCE` operation may depend on the system architecture and checks are necessary before proceeding to large scale runs with this algorithm. The total force

acting on each triangular node is computed as the summation of the external forces (pressure + viscous) and the internal forces arising from the elastic potentials i.e. $\boldsymbol{F} = \boldsymbol{F}_{\text{ext}} + \boldsymbol{F}_{\text{int}}$; for the first time step the immersed body is in its reference state and all internal forces are equal to zero and in every succeeding time step F_{int} is the internal elastic forces computed in the previous time step. With this step every processor updates the position of the triangle nodes based on the total force.

Step 4(c) involves computing the internal elastic forces derived from the potentials on each immersed body. Since this requires the full body to be treated as a whole, we compute the location of the mean axial index of every individual immersed body from the information in the array `pind`. The processor responsible for this axial index takes care of computing all the internal elastic forces (i.e. in-plane deformation, out-of-plane deformation, volume constraint and area constraint) and computing the net internal force acting on each node belonging to its allocated immersed body. While computing the internal forces on each immersed body does not require any information from the Eulerian mesh, such an allocation ensures the computing load is distributed evenly across all processors. Also it is important to note that MLS interpolations which are the computationally expensive steps in this FSI-IBM code are still performed only by processors containing the Lagrangian markers. The pseudo code for this operation is given below.

```
do i=1,N_particle

    if z_ave(i) >= N_start(myid).AND.z_ave(i) < N_end(myid)
        - Compute forces from in-plane deformation.
        - Compute forces from out-of-plane deformation.
        - Compute forces from volume potential.
        - Compute forces from area potential.
        - Sum up forces from all potential on the nodes.
    end if

end do
```

In step 4(d), we reduce the internal forces ($\boldsymbol{F}^{\text{int}}$) over all triangle nodes with an `MPI_ALLREDUCE` operation similar to the operation in 4(b). With this we complete all the steps required for one full iteration of the flow solver and the IBM coupled with the deformation.

References

[1] S. Balachandar, J. Eaton, Turbulent dispersed multiphase flow, Annu. Rev. Fluid Mech. 42 (2010) 111–133.
[2] G. Tryggvason, S. Dabiri, B. Aboulhasanzadeh, J. Lu, Multiscale considerations in direct numerical simulations of multiphase flows, Phys. Fluids 25 (3) (2013) 031302.
[3] J. Freund, Numerical simulation of flowing blood cells, Annu. Rev. Fluid Mech. 46 (2014) 67–95.
[4] F. Sotiropoulos, T. Le, A. Gilmanov, Fluid mechanics of heart valves and their replacements, Annu. Rev. Fluid Mech. 48 (2016) 259–283.
[5] R. Mittal, J.H. Seo, V. Vedula, Y.J. Choi, H. Liu, H.H. Huang, S. Jain, L. Younes, T. Abraham, R.T. George, Computational modeling of cardiac hemodynamics: current status and future outlook, J. Comput. Phys. 305 (2016) 1065–1082.
[6] E.H. Dowell, R. Clark, D. Cox, A Modern Course in Aeroelasticity, vol. 3, Springer, 2004.
[7] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (2) (1972) 252–271.
[8] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[9] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[10] M. Uhlmann, T. Doychev, Sedimentation of a dilute suspension of rigid spheres at intermediate Galileo numbers: the effect of clustering upon the particle motion, J. Fluid Mech. 752 (2014) 310–348.
[11] F. Picano, W.-P. Breugem, L. Brandt, Turbulent channel flow of dense suspensions of neutrally buoyant spheres, J. Fluid Mech. 764 (2015) 463–487.
[12] A. Prosperetti, Life and death by boundary conditions, J. Fluid Mech. 768 (2015) 1–4.
[13] W. Fornari, F. Picano, L. Brandt, Sedimentation of finite-size spheres in quiescent and turbulent environments, J. Fluid Mech. 788 (2016) 640–669.
[14] S. Schwarz, T. Kempe, J. Fröhlich, An immersed boundary method for the simulation of bubbles with varying shape, J. Comput. Phys. 315 (2016) 124–149.
[15] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, Appl. Mech. Rev. 56 (3) (2003) 331–347.
[16] M. de Tullio, P. De Palma, M. Napolitano, G. Pascazio, Recent advances in the development of an immersed boundary method for industrial applications, in: Computational Fluid Dynamics, Springer, 2011, pp. 601–606.
[17] A. Prosperetti, G. Tryggvason, Computational Methods for Multiphase Flow, Cambridge University Press, 2007.
[18] C. Crowe, T. Troutt, J. Chung, Numerical models for two-phase turbulent flows, Annu. Rev. Fluid Mech. 28 (1) (1996) 11–43.
[19] J. Magnaudet, I. Eames, The motion of high-Reynolds-number bubbles in inhomogeneous flows, Annu. Rev. Fluid Mech. 32 (1) (2000) 659–708.
[20] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (1) (1999) 567–603.
[21] M. de Tullio, G. Pascazio, A Moving-Least-Squares immersed boundary method for simulating the fluid–structure interaction of elastic bodies with arbitrary thickness, J. Comput. Phys. 325 (2016) 201–225.
[22] Y.J. Choi, J. Constantino, V. Vedula, N. Trayanova, R. Mittal, A new MRI-based model of heart function with coupled hemodynamics and application to normal and diseased canine left ventricles, Frontiers in Bioeng. and Biotech. 3 (2015).
[23] J.H. Seo, V. Vedula, T. Abraham, R. Mittal, Multiphysics computational models for cardiac flow and virtual cardiography, Int. J. Numer. Methods Biomed. Eng. 29 (8) (2013) 850–869.
[24] M. De Tullio, A. Cristallo, E. Balaras, R. Verzicco, Direct numerical simulation of the pulsatile flow through an aortic bileaflet mechanical heart valve, J. Fluid Mech. 622 (2009) 259–290.
[25] M. De Tullio, L. Afferrante, G. Demelio, G. Pascazio, R. Verzicco, Fluid–structure interaction of deformable aortic prostheses with a bileaflet mechanical valve, J. Biomech. 44 (9) (2011) 1684–1690.

[26] X. Zheng, J. Seo, V. Vedula, T. Abraham, R. Mittal, Computational modeling and analysis of intracardiac flows in simple models of the left ventricle, Eur. J. Mech. B, Fluids 35 (2012) 31–39.
[27] J.H. Seo, R. Mittal, Effect of diastolic flow patterns on the function of the left ventricle, Phys. Fluids 25 (11) (2013) 110801.
[28] J.H. Seo, V. Vedula, T. Abraham, A.C. Lardo, F. Dawoud, H. Luo, R. Mittal, Effect of the mitral valve on diastolic flow patterns, Phys. Fluids 26 (12) (2014) 121901.
[29] V. Vedula, S. Fortini, J.-H. Seo, G. Querzoli, R. Mittal, Computational modeling and validation of intraventricular flow in a simple model of the left ventricle, Theor. Comput. Fluid Dyn. 28 (6) (2014) 589–604.
[30] B. Baillargeon, N. Rebelo, D.D. Fox, R.L. Taylor, E. Kuhl, The living heart project: a robust and integrative simulator for human heart function, Eur. J. Mech. A, Solids 48 (2014) 38–47.
[31] M. Rai, P. Moin, Direct simulations of turbulent flow using finite-difference schemes, J. Comput. Phys. 96 (1) (1991) 15–53.
[32] R. Verzicco, P. Orlandi, A finite-difference scheme for three-dimensional incompressible flows in cylindrical coordinates, J. Comput. Phys. 123 (2) (1996) 402–414.
[33] E.P. van der Poel, R. Ostilla-Mónico, J. Donners, R. Verzicco, A pencil distributed finite difference code for strongly turbulent wall-bounded flows, Comput. Fluids 116 (2015) 10–16.
[34] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2) (2005) 448–476.
[35] P. Lancaster, K. Salkauskas, Surfaces generated by moving least squares methods, Math. Comput. 37 (155) (1981) 141–158.
[36] G. Liu, Y. Gu, An Introduction to Meshless Methods and Their Programming, Springer, Berlin, Germany, 2005.
[37] T. Belytschko, L. Gu, Y. Lu, Fracture and crack growth by element free Galerkin methods, Model. Simul. Mater. Sci. Eng. 2 (3A) (1994) 519.
[38] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, P. Krysl, Meshless methods: an overview and recent developments, Comput. Methods Appl. Mech. Eng. 139 (1) (1996) 3–47.
[39] Y. Krongauz, T. Belytschko, Enforcement of essential boundary conditions in meshless approximations using finite elements, Comput. Methods Appl. Mech. Eng. 131 (1) (1996) 133–145.
[40] D. Hegen, Element-free Galerkin methods in combination with finite element approaches, Comput. Methods Appl. Mech. Eng. 135 (1) (1996) 143–166.
[41] S. Atluri, J. Cho, H.-G. Kim, Analysis of thin beams, using the meshless local Petrov–Galerkin method, with generalized moving least squares interpolations, Comput. Mech. 24 (5) (1999) 334–347.
[42] S. Schaefer, T. McPhail, J. Warren, Image deformation using moving least squares, ACM Trans. Graph. 25 (2006) 533–540.
[43] S. Fleishman, D. Cohen-Or, C. Silva, Robust moving least-squares fitting with sharp features, ACM Trans. Graph. 24 (2005) 544–552.
[44] R. Kolluri, Provably good moving least squares, ACM Trans. Algorithms 4 (2) (2008) 18.
[45] Q.-H. Zeng, D.-T. Lu, Curve and surface fitting based on moving least-squares methods, J. Eng. Graph. 1 (3) (2004) 84–87.
[46] L. Kobbelt, M. Botsch, A survey of point-based techniques in computer graphics, Comput. Graph. 28 (6) (2004) 801–814.
[47] M. Vanella, E. Balaras, A moving-least-squares reconstruction for embedded-boundary formulations, J. Comput. Phys. 228 (18) (2009) 6617–6628.
[48] J. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, J. Comput. Phys. 215 (1) (2006) 12–40.
[49] A. van Gelder, Approximate simulation of elastic membranes by triangulated spring meshes, J. Graph. Tools 3 (2) (1998) 21–41.
[50] M. Chen, F. Boyle, Investigation of membrane mechanics using spring networks: application to red-blood-cell modelling, Mater. Sci. Eng. C 43 (2014) 506–516.
[51] D. Fedosov, B. Caswell, G. Karniadakis, Systematic coarse-graining of spectrin-level red blood cell models, Comput. Methods Appl. Mech. Eng. 199 (29) (2010) 1937–1948.
[52] M. Dupin, I. Halliday, C. Care, L. Munn, Lattice Boltzmann modelling of blood cell dynamics, Int. J. Comput. Fluid Dyn. 22 (7) (2008) 481–492.
[53] T. Krüger, Effect of tube diameter and capillary number on platelet margination and near-wall dynamics, Rheol. Acta (2016) 1–16.
[54] S. Schwarz, T. Kempe, J. Fröhlich, A temporal discretization scheme to compute the motion of light particles in viscous flows by an immersed boundary method, J. Comput. Phys. 281 (2015) 591–613.
[55] L. Landau, E. Lifshitz, Fluid Mechanics, 1959.
[56] M. Emans, C. Zenger, An efficient method for the prediction of the motion of individual bubbles, Int. J. Comput. Fluid Dyn. 19 (4) (2005) 347–357.
[57] G. Taylor, The viscosity of a fluid containing small drops of another fluid, Proc. R. Soc. Lond. (1932) 41–48.
[58] G. Taylor, The formation of emulsions in definable fields of flow, Proc. R. Soc. Lond. (1934) 501–523.
[59] P. Maffettone, M. Minale, Equation of change for ellipsoidal drops in viscous flow, J. Non-Newton. Fluid Mech. 78 (2) (1998) 227–241.
[60] M. de Tullio, J. Nam, G. Pascazio, E. Balaras, R. Verzicco, Computational prediction of mechanical hemolysis in aortic valved prostheses, Eur. J. Mech. B, Fluids 35 (2012) 47–53.
[61] L. Biferale, C. Meneveau, R. Verzicco, Deformation statistics of sub-Kolmogorov-scale ellipsoidal neutrally buoyant drops in isotropic turbulence, J. Fluid Mech. 754 (2014) 184–207.
[62] V. Spandan, R. Verzicco, D. Lohse, Deformation and orientation statistics of neutrally buoyant sub-Kolmogorov ellipsoidal droplets in turbulent Taylor–Couette flow, J. Fluid Mech. 809 (2) (2016) 480–501.
[63] S. Torza, R. Cox, S. Mason, Particle motions in sheared suspensions xxvii. Transient and steady deformation and burst of liquid drops, J. Colloid Interface Sci. 38 (2) (1972) 395–411.
[64] S. Guido, M. Villone, Three-dimensional shape of a drop under simple shear flow, J. Rheol. 42 (2) (1998) 395–415.
[65] E. Loth, Quasi-steady shape and drag of deformable bubbles and drops, Int. J. Multiph. Flow 34 (6) (2008) 523–546.
[66] M. Falchi, G. Querzoli, G. Romano, Robust evaluation of the dissimilarity between interrogation windows in image velocimetry, Exp. Fluids 41 (2) (2006) 279–293.
[67] M. Uhlmann, Simulation of Particulate Flows on Multi-Processor Machines with Distributed Memory, Ciemat, 2004.
[68] S. Wang, G. He, X. Zhang, Parallel computing strategy for a flow solver based on immersed boundary method and discrete stream-function formulation, Comput. Fluids 88 (2013) 210–224.