"Now it is our turn to study statistical mechanics.

Perhaps it will be wise to approach the subject cautiously."

D. L. Goodstein, *States of Matter*

3

A fully-integrated lattice Boltzmann method for fluid–structure interaction

3.1 Introduction

Simulation has become an indispensable tool for studying fluid–structure interaction (FSI) problems. We have full autonomy in setting the physical parameters, and thus we can gain more information from simulations that are experimentally hard to measure or theoretical difficult to approximate. All simulations (everything) could potentially be done on a single or multiple computers (everywhere) in a parallel fashion (all at once). A generalizable numerical method seems to be the key to unlocking the potential

This chapter is reformatted from Y. Sun and C. H. Rycroft, *A fully-integrated lattice Boltzmann method for fluid–structure interaction*, under review at Journal of Computational Physics (2024). (arXiv:2402.12696)

of simulations in studying FSI problems. However, as we have introduced in Chapter 1, developing FSI methods is a nontrivial task. The main challenge stems from the intrinsic dichotomy in the preferred simulation approach between solids and fluids: Because solid stress comes from strain, while fluid stress comes from strain rate, solid simulations often use Lagrangian approaches [16–19], but Eulerian methods are favored by fluid simulations [20–22]. State-of-the-art FSI methods can be loosely categorized into four types: mesh-free, Lagrangian, Eulerian–Lagrangian, and Eulerian (Fig. 1.1).

The basis of a fully Eulerian FSI method is to represent solids in an Eulerian framework [79–82]. One recent example is the reference map technique (RMT) [93, 95, 96], which uses the reference map field—an Eulerian mapping from the deformed state to the undeformed state of the solid—to describe finite-strain large solid deformation in the Eulerian framework. For FSI simulations, the RMT uses the level set field [77, 78, 195] to describe solid–fluid interfaces of multiple solid objects. It can be coupled with any Eulerian numerical methods for the fluid update. Jain *et al.* [97] developed a conservative implementation using the finite volume method. Rycroft *et al.* later introduced the IncRMT [98] for incompressible FSI simulations using Chorin's projection method [20, 196], which is further extended to three dimensions (RMT3D) [99] and mixtures of soft and rigid solids [100]. However, the authors reported [98] that up to two-thirds of the total simulation time is dedicated to solving an elliptic equation over the entire domain when imposing the fluid incompressibility constraint. One promising alternative to breaking this computation bottleneck while maintaining fluid (quasi-)incompressibility is to use the lattice Boltzmann (LB) method [13, 76, 197] for the fluid update, as proposed by Sun in her master's thesis [101].

Originating from the kinetic theory of gases [198, 199], the LB method uses mesoscopic probability distribution functions, known as populations, as the main simulation variables, rather than tracking macroscopic hydrodynamic fields as in other Eulerian methods. Macroscopic fields, particularly density and velocity, can be retrieved from this statistical view of fluid motion as moments of populations [200] through nested for-loops. Since calculations are local, the LB method requires no special adjustment to accommodate complex geometries of multi-body interactions [201, 202], and is well-suited for parallelization [203–205]. Unlike the projection method [20, 196] which imposes exact geometric incompressibility by projecting the velocity field to be divergence-free, the LB method does not solve the Poisson problem for pressure to impose the fluid incompressibility constraint. Instead, this constraint is automatically encoded in the LB method under the small Mach number limit for fully-developed simulations [76]. At the cost of losing exact geometric incompressibility, the LB method can substantially improve the code performance in the fluid update for the RMT-based FSI simulations.

However, existing LB boundary conditions do not constitute a fully-integrated LB method for FSI simulations [206] to simulate multiple moving finite-strain solids of different densities on a fixed computational grid. These boundary conditions can be broadly grouped into two types [206]: collision-based and force-based. Collision-based methods, like bounce-back methods [207–209], extrapolation methods [210–213], and Ladd's collision-based coupling method [214], require modifications to the collision operator and can only simulate rigid solids. Force-based methods, like coupling with stochastic particle dynamics [215] and immersed boundary methods [63], allow solid forces to be computed along the solid–

fluid interface as Lagrangian markers. However, they require two frameworks and spend considerable computation time in force interpolations between Eulerian and Lagrangian. Although force-based methods have successfully simulated deformable solids, such as polymer chains [215] and red blood cells [63], there is still a need for a fully Eulerian boundary condition to model moving deformable solid–fluid interfaces with density difference. This boundary condition is different from the multicomponent LB interface [216], which uses phase field to represent each fluid component on an Eulerian grid while imposing interfacial forces on the membrane of fluid-filled objects [217]. It is also different from the Cahn–Hilliard formulation [218], which uses a phase field [219] to represent a smooth function between two phases. Our Eulerian boundary condition aims to enable a fully-integrated LB method for modeling FSI [206] that can explicitly calculate solid stress and solid–fluid interaction on a single fixed Eulerian grid, thus permitting direct implementation to optimize the parallel processing capability of the LB method.

In this chapter, we present the lattice Boltzmann reference map technique (LBRMT), a fully-integrated lattice Boltzmann method for FSI simulations. It blends the parallel fluid calculation of the LB method and the Eulerian solid deformation of the RMT for a fully Eulerian FSI simulation approach. In addition to increasing the computation speed and the maximum number of solids in the simulation, the LBRMT introduces a new LB boundary condition method to couple finite-strain solids with fluids on the same computational grid. This boundary condition, *smooth flux correction*, is designed to preserve the flux across the solid–fluid interface, thus allowing us to maintain the density difference between the two phases while ensuring all computations are still locally parallelizable. We present the theoretical formula-

tion and numerical implementation of the LBRMT, where we introduce the RMT, the LB method, and the FSI configuration with the new LB boundary condition in Section 3.2. We detail the numerical methods in Section 3.3, with a highlight on the custom `multimaps` data structure for simple solid tracking and collision detection in multi-body contact. Finally, we establish the baseline accuracy of the LBRMT with a benchmark example in Section 3.4.1, then showcase its functionalities to model solid rotating in Section 3.4.2 and settling in Section 3.4.3, and bio-inspired simulations of collective behavior in complex suspension Section 3.4.4. We conclude our method and discuss future directions in Section 3.5.

3.2 Theoretical formulation

Both solids and fluids satisfy the Cauchy momentum equation,

$$\rho\left(\frac{\partial \boldsymbol{v}}{\partial t}+(\boldsymbol{v}\cdot\nabla)\,\boldsymbol{v}\right)=\nabla\cdot\boldsymbol{\sigma}+\boldsymbol{f}_{\text{ext}}, \tag{3.1}$$

where ρ, $\boldsymbol{v}$, $\boldsymbol{\sigma}$ represent the global density, velocity, and stress field, respectively. We absorb external body forces (such as gravity) into $\boldsymbol{f}_{\text{ext}}$. Naturally for $\boldsymbol{\sigma}$, we use solid stress for solids and fluid stress for fluids. For solids, we use the RMT to directly compute the solid stress $\boldsymbol{\sigma}_s$ from the deformation gradient tensor. For fluids, we mesoscopically reconstruct the fluid stress $\boldsymbol{\sigma}_f$ with the LB method. We build a smooth transition between the solid and fluid phases to maintain a global density, velocity, and stress field. This smooth transition, *the blur zone*, is defined based on the level set values of the solid–fluid interface. Within the blur zone, we introduce a *smooth flux correction* to maintain density difference between two phases.

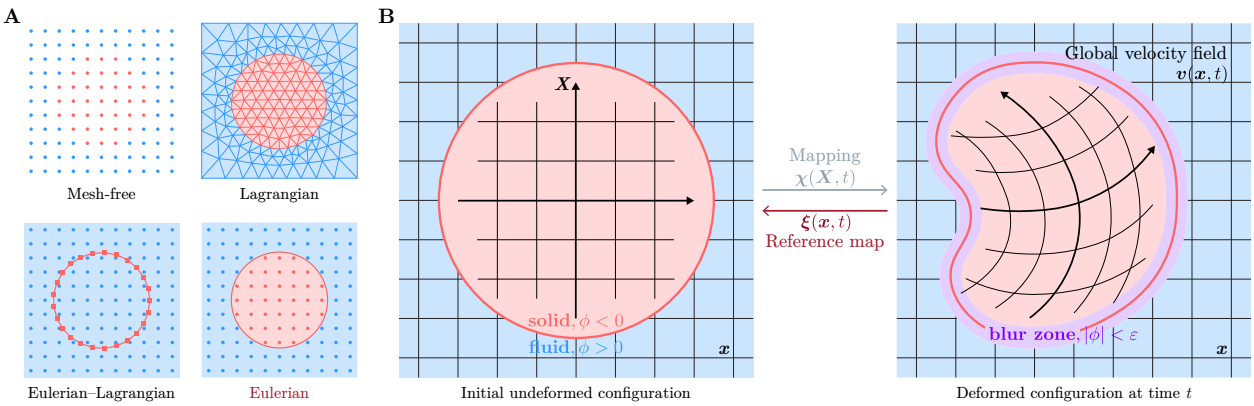

Figure 3.1: Illustration of large solid deformation and lattice Boltzmann reference map technique. (A) Reprise of Fig. 1.1 with an emphasis on Eulerian methods. Types of FSI methods based on the solid and fluid discretization frameworks. Mesh-free methods use particles to represent both phases; Lagrangian methods use unstructured adaptive meshes for both solids and fluids; Eulerian–Lagrangian methods use a fixed Eulerian mesh for the fluid, but moving Lagrangian markers for solids; and Eulerian methods only use one fixed computational grid for both phases. **(B)** Overview of the large solid deformation and the lattice Boltzmann reference map technique for FSI simulations on a fixed computational grid. A time-dependent mapping $\boldsymbol{\chi}(\boldsymbol{X}, t)$ is applied to an initially undeformed solid with a reference coordinate system $\boldsymbol{X}$, resulting in a deformed coordinate system at time t. The inverse mapping is the reference map $\boldsymbol{\xi}(\boldsymbol{x}, t)$, which maps a deformed solid back to its initial configuration on the same fixed grid. A level set function $\phi(\boldsymbol{x}, t)$ is employed to define solid geometries, whose signed value determines the solid and fluid phases. To transition between the two phases, a blur zone with half-width ε is defined as $|\phi| < \varepsilon$ to smooth out the density, velocity, and stress field.

3.2.1 Reference map technique

The reference map technique (RMT) is an Eulerian numerical method to simulate solids undergoing large deformation. We use a finite-strain hyperelastic model [84, 220] to describe the solid material. In Fig. 3.1(B), at time $t = 0$, the solid is at an undeformed reference configuration with coordinate system $\boldsymbol{X}$. After some time t, the reference configuration is deformed to a new coordinate system $\boldsymbol{x}$. Consider a continuous time-dependent mapping $\boldsymbol{\chi}(\boldsymbol{X}, t)$ from the undeformed coordinate $\boldsymbol{X}$ to the deformed coordinate system $\boldsymbol{x}$, *i.e.* $\boldsymbol{x} = \boldsymbol{\chi}(\boldsymbol{X}, t)$, we can denote the deformation gradient tensor $\boldsymbol{F}$ [85, 87, 221] as the derivative of each component of the deformed coordinate system $\boldsymbol{x}$ with respect to each component

of the reference coordinate system $\boldsymbol{X}$,

$$\boldsymbol{F} = \frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{X}}. \tag{3.2}$$

The deformation gradient tensor $\boldsymbol{F}$ can also be expressed in terms of the reference map $\boldsymbol{\xi}(\boldsymbol{x}, t)$ [84], which is an Eulerian mapping from the deformed coordinate system $\boldsymbol{x}$ to the undeformed coordinate system $\boldsymbol{X}$. Since $\boldsymbol{\xi}(\boldsymbol{x}, t)$ is the inverse mapping of $\boldsymbol{\chi}$, by the chain rule, it also gives rise to an Eulerian definition of the deformation gradient tensor $\boldsymbol{F}$,

$$\boldsymbol{F} = \left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}} \right)^{-1}. \tag{3.3}$$

The reference map field $\boldsymbol{\xi}$ is initialized as $\boldsymbol{\xi}(\boldsymbol{x}, 0) = \boldsymbol{x}$ and satisfies the advection equation with a material velocity $\boldsymbol{v}$ of the solid, which allows us to evolve solid deformation through time:

$$\frac{\partial \boldsymbol{\xi}}{\partial t} + (\boldsymbol{v} \cdot \nabla) \boldsymbol{\xi} = \mathbf{0}. \tag{3.4}$$

The RMT is an Eulerian method for simulating solid mechanics because the main simulation component, the reference map field $\boldsymbol{\xi}$, is an Eulerian field that can be easily tracked and calculated. On a fixed Eulerian grid, each node stores the reference map field $\boldsymbol{\xi}$ and material velocity $\boldsymbol{v}$. At a given time t, we use Eq. (3.3) to calculate the deformation gradient tensor $\boldsymbol{F}$. With a constitutive relation $\mathbf{f}$, we specify the solid stress $\boldsymbol{\sigma}_s = \mathbf{f}(\boldsymbol{F})$. Using the Cauchy momentum equation Eq. (3.1), the divergence of the solid stress $\nabla \cdot \boldsymbol{\sigma}_s$ formulates the update rule for the material velocity $\boldsymbol{v}$. After obtaining the updated material

velocity, we advect the reference map field $\boldsymbol{\xi}$ to step forward in time. Eqs. (3.1), (3.3) and (3.4) form one update on the solid deformation using the RMT, and this process can be discretized using any Eulerian discretization schemes.

3.2.2 Lattice Boltzmann method with forces

Based on the kinetic theory of gases [198, 199], the lattice Boltzmann (LB) method simulates fluid dynamics via a minimal form of the Boltzmann equation in a discrete velocity space [76]. In a two-dimensional discrete space–time domain with equal grid spacing $\Delta x^* = \Delta y^*$ and timestep Δt^*, the velocity space is reduced to nine discrete velocities $\boldsymbol{c}_i = \Delta \boldsymbol{x}^*/\Delta t^*$, known as the D_2Q_9 model [222, 223]. For a node $(\boldsymbol{x}, t)$ in the discrete D_2Q_9 space, it has nine probability distribution functions $f_i(\boldsymbol{x}, t)$—commonly referred to as *populations* in the LB literature [13]. Each population f_i represents the possibility of moving in the direction of velocity $\boldsymbol{c}_i$. Because the grid spacing and timestep are typically set to be dimensionless ($\Delta x^* = 1$, $\Delta t^* = 1$) in the LB literature [76], the discrete velocity $\boldsymbol{c}_i$ has unit velocity components (Table 3.1) and can only spatially increment to eight neighboring nodes in one timestep (Fig. 3.2(A)). This nondimensionalized unit choice also generalizes the LB simulations to physical systems of any size. To convert simulations back to real-life scale, we simply need to multiply the respective physical unit scales; this conversion is derived in detail in Appendix B.1.

Rather than directly solving the Navier–Stokes equations, the LB method reconstructs macroscopic fields, fluid density ρ and velocity $\boldsymbol{v}$, by tracking mesoscopic populations $f_i(\boldsymbol{x}, t)$ using the discretized

Direction i	Discrete velocity $\boldsymbol{c}_i$	Weight w_i
0	$(\ 0,\ \ 0)$	$4/9$
1	$(\ 1,\ \ 0)$	$1/9$
2	$(\ 0,\ \ 1)$	$1/9$
3	$(-1,\ \ 0)$	$1/9$
4	$(\ 0,-1)$	$1/9$
5	$(\ 1,\ \ 1)$	$1/36$
6	$(-1,\ \ 1)$	$1/36$
7	$(-1,-1)$	$1/36$
8	$(\ 1,-1)$	$1/36$

Table 3.1: Summary of the velocity sets $\boldsymbol{c}_i$ and their weights w_i in D_2Q_9 model. Each discrete velocity $\boldsymbol{c}_i$ indicates a direction for f_i at node (i, j) moving to its eight neighboring nodes with associated weights w_i. Since the LB simulations use dimensionless grid spacing and timestep, $\Delta x^* = 1$ and $\Delta t^* = 1$, in the discretized domain, $\boldsymbol{c}_i = \Delta \boldsymbol{x}^*/\Delta t^*$ also has unit velocity components.

lattice Boltzmann equation (LBE), with the Bhatnagar–Gross–Krook (BGK) collision operator Ω_i [224]:

$$f_i(\boldsymbol{x} + \underbrace{\boldsymbol{c}_i \Delta t^*}_{=\Delta \boldsymbol{x}^*}, t + \Delta t^*) = \underbrace{f_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x}, t)}_{=\widehat{f}_i} = f_i(\boldsymbol{x}, t) \underbrace{-\frac{1}{\tau}\left[f_i(\boldsymbol{x}, t) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t)\right]}_{=\Omega_i(\boldsymbol{x}, t)}, \tag{3.5}$$

where populations $f_i(\boldsymbol{x}, t)$ at position $\boldsymbol{x}$ move to the neighboring nodes $\boldsymbol{x} + \Delta \boldsymbol{x}^*$ with discrete velocity $\boldsymbol{c}_i$ in one timestep Δt^*, while relaxing towards their equilibrium population f_i^{eq} via the BGK collision operator Ω_i. The local equilibrium distribution function f_i^{eq} is valid only when populations are close to the Maxwell–Boltzmann equilibrium [225] and can be approximated by a second-order Taylor expansion [13],

$$f_i^{\mathrm{eq}}(\boldsymbol{x}, t) = w_i \rho \left[1 + \frac{\boldsymbol{v} \cdot \boldsymbol{c}_i}{c_s^2} + \frac{(\boldsymbol{v} \cdot \boldsymbol{c}_i)^2 - c_s^2 \boldsymbol{v}^2}{2 c_s^4}\right]. \tag{3.6}$$

In Eqs. (3.5) and (3.6), c_s is the LB speed of sound, chosen to be $c_s = \sqrt{1/3}(\Delta x^*/\Delta t^*)$, τ is the relax-

ation time to local equilibrium, related to the kinematic viscosity $\nu = c_s^2\left(\tau - \frac{1}{2}\right)$, and w_i is the weight associated with each population, determined by the Hermite polynomial of the D_2Q_9 model (Table 3.1).

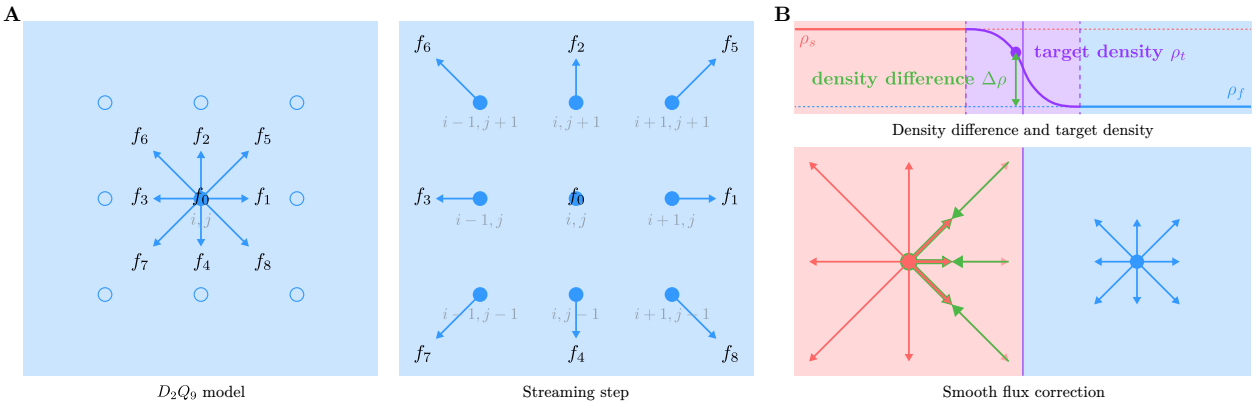

Figure 3.2: Diagram of the D_2Q_9 lattice model and the smooth flux correction boundary condition. (A) The blue arrows represent the nine discrete velocities $\boldsymbol{c}_i$. Each f_i is a probability distribution function of a particle velocity at (i, j) in the direction of the blue arrow. The empty circles represent the neighboring nodes. In the streaming step, each post-collision $\widehat{f}_i$ moves from its original position in the direction of the arrow to its neighboring nodes. Its value then becomes the new f_i of these nodes in the next timestep. **(B)** Illustration of the smooth flux correction (SFC) for solid–fluid interface with density difference. In order to remove the outgoing flux from the higher density region to the lower density region, we add a correction flux (green arrows) to the original outgoing populations (transparent red arrows) of a node to remove additional fluxes crossing the interface. We then add these additional fluxes back to f_0 to enforce mass conservation. The resultant outgoing populations are labeled with red arrows with green outlines. The amount of flux removed depends on the density differences between the two regions, which can be computed from the target density based on the blur zone (Eq. (3.11)).

The LBE (Eq. (3.5)) can be decomposed into two parts: a *collision* step and a *streaming* step. The collision step, characterized by the BGK collision operator Ω_i, computes post-collision populations $\widehat{f}_i$ and controls how populations f_i of one node locally interact with each other and relax toward their Maxwell–Boltzmann equilibrium due to the effect of microscopic particle collision. Momentum is conserved in the collision step, while the kinetic energy is not, hence resulting in energy dissipation in the form of viscosity. In the streaming step (Fig. 3.2(A)), post-collision populations $\widehat{f}_i$ move forward to the neighboring nodes along the $\boldsymbol{c}_i$ direction, becoming the updated f_i for the next timestep. There is no information loss

since the streaming step is local to each node thus exact up to machine precision. As the LB method steps forward in time with timestep Δt^* in a discretized space with grid spacing Δx^*, it is essentially a finite-difference method on a fixed Eulerian grid. This analogy makes the LB method an ideal complement to the RMT for FSI simulations.

When external forces are present, we can compute macroscopic fluid quantities as moments of populations f_i following the forcing scheme of Guo *et al.* [226] by modifying Eq. (3.5) to include a forcing term F_i:

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t^*, t + \Delta t^*) = f_i(\boldsymbol{x}, t) + \Omega_i + \Delta t^* \left(1 - \frac{1}{2\tau}\right) F_i. \tag{3.7}$$

Each forcing term F_i corresponds to one population f_i, and is related to a second-order approximation of the weighted macroscopic external force density $\boldsymbol{F}$ in the velocity space:

$$F_i = w_i \left(\frac{\boldsymbol{c}_i - \boldsymbol{v}}{c_s^2} + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{v})\, \boldsymbol{c}_i}{c_s^4} \right) \cdot \boldsymbol{F}. \tag{3.8}$$

In order to ensure a second-order time accuracy to prevent unstable simulations with uncontrollable noise caused by discrete lattice artifacts [76], a half-force correction is added to the equation of velocity (Eq. (3.9)). Macroscopic fluid fields can thus be retrieved from moments of population f_i. The zeroth moment is the fluid density ρ, and the first moment corresponds to the local fluid momentum, *i.e.* the fluid velocity $\boldsymbol{v}$:

$$\rho = \sum_i f_i, \quad \boldsymbol{v} = \frac{1}{\rho} \sum_i \boldsymbol{c}_i f_i + \frac{\Delta t^*}{2\rho} \boldsymbol{F}. \tag{3.9}$$

The LB fluid update with some generic force density $\boldsymbol{F}$ shares a similar configuration to the RMT solid update. On a fixed Eulerian grid, each node stores the nine populations f_i, the macroscopic density ρ and velocity $\boldsymbol{v}$, and if needed, the nine equilibrium populations f_i^{eq}. At a given time t, we first use Eq. (3.6) to calculate the equilibrium population f_i^{eq} with the current fluid density ρ and velocity $\boldsymbol{v}$. We then perform the collision step and calculate the collision operator Ω_i with Eq. (3.5) and the external force density F_i with Eq. (3.8). By combining these two terms together, we obtain the post-collision populations $\widehat{f_i}$. Following the streaming step, these post-collision populations $\widehat{f_i}$ become the updated populations f_i at the neighboring nodes. Lastly, we reconstruct the updated density ρ and velocity $\boldsymbol{v}$ using Eq. (3.9). For now, we have not yet specified the macroscopic external force density $\boldsymbol{F}$: It can be related to gravity, or a pressure gradient of a channel flow. In Section 3.2.4, we connect this force density $\boldsymbol{F}$ with the divergence of the solid stress for FSI simulations.

3.2.3 Smooth flux correction

Here we introduce our new fully Eulerian boundary condition for a fully-integrated LB FSI method [206]. The *smooth flux correction* is a framework to preserve density differences between solids and fluids across their interfaces. This framework does not imply that solid mechanics can be described using kinetic theory—the assumptions on mesoscopic particles do not hold for solids. Instead, we employ the LB method as a computational tool to enforce three constraints that are fundamental in FSI simulations: mass conservation, momentum conservation, and density difference.

Suppose we have a one-dimensional (1D) density domain with two regions of densities ρ_f and ρ_s,

where

$$\rho = \begin{cases} \rho_f & \text{if } x < 0 \\ \\ \rho_s & \text{if } x \geq 0 \end{cases} \quad \text{and} \quad \rho_s > \rho_f. \tag{3.10}$$

If we do not impose any modifications on x, there will be flux led by diffusion from the higher density region to the lower density region. Even though the total mass is conserved in diffusion, ρ will eventually average to $(\rho_f + \rho_s)/2$ in the entire domain, thus losing the density difference. To counteract this outgoing flux from ρ_s to ρ_f, we impose a *correction* at $x = 0$ to make up for the loss of flux in ρ_s and remove excessive flux in ρ_f.

The basic idea of our correction is based on flux balance: We subtract the excessive flux $\boldsymbol{q}_e$ from the total flux $\boldsymbol{q}$ going from the higher density region to the lower density region to ensure density difference. Denote the velocity at $x = 0$ as $\boldsymbol{v}$, then $\boldsymbol{q}_e = (\rho_s - \rho_f)\boldsymbol{v}$. If we subtract $\boldsymbol{q}_e$ from the outgoing flux $\boldsymbol{q} = \rho_s \boldsymbol{v}$, then there should be no diffusion flux from the higher density region to the lower density region, thus preserving the density difference. Since density flux $\boldsymbol{q}$ is equivalent to momentum $\boldsymbol{J} = \rho\boldsymbol{v}$, we modify the LB populations to ensure no excessive population leaving the higher density region. By putting the outgoing flux back onto f_0, we have also ensured mass and momentum conservation; see Appendix B.3 for the derivation.

We extend this 1D correction to the two-dimensional (2D) D_2Q_9 model and develop the smooth flux correction (SFC) to preserve the density difference (Fig. 3.2(B)). We compute the excessive flux based on the density difference $\Delta\rho$ between solids and fluids. $\Delta\rho$ is a smooth transition between the two phases

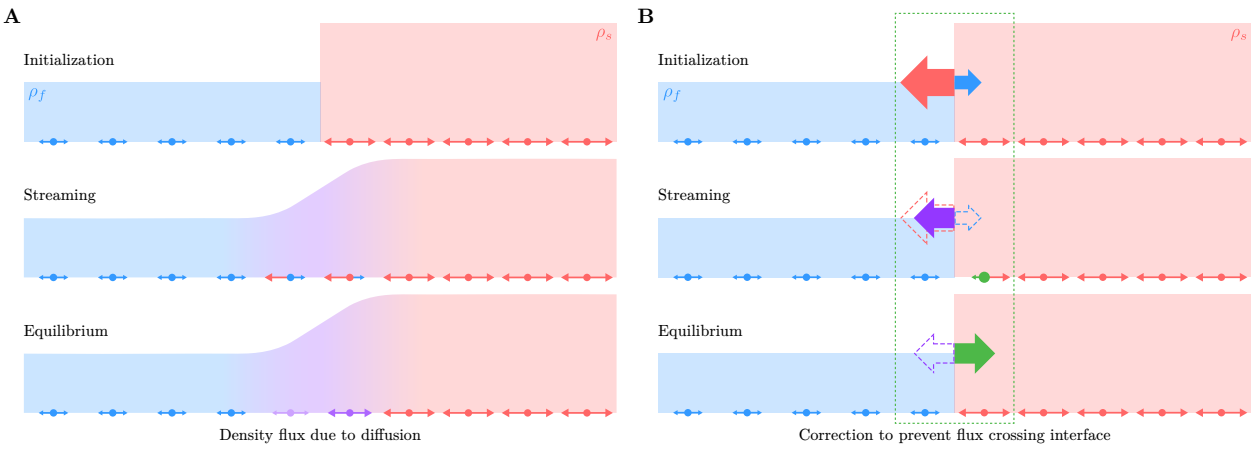

Figure 3.3: Illustration of one-dimensional smooth flux correction. **(A)** Without any constraints on the solid–fluid interface, density flux goes from the higher density region (x_s) to the lower density region (x_f). This flux blurs the density difference between the two phases in the streaming and equilibrium steps, eventually averaging out the density in the domain. **(B)** By adding a correction to the solid node closest to the interface, *i.e.* removing additional outgoing flux and putting it back to the green node, we can preserve the density difference. The outgoing flux, illustrated by the purple arrow, is the difference between the density flux from x_s to x_f (the red arrow) and the density flux from x_f to x_s (the blue arrow). The correction flux (the green arrow) is equal and opposite to the outgoing flux.

in the blur zone, computed as the difference between the target density ρ_t and fluid density ρ_f:

$$\Delta\rho = \rho_t - \rho_f \quad \text{where} \quad \rho_t = H_\varepsilon(\phi)\rho_f + (1 - H_\varepsilon(\phi))\,\rho_s. \tag{3.11}$$

The target density field is a reference density field based on the geometric configuration of solids defined by the level set function. By correcting the flux, we let the populations relax toward the reference density field, which preserves the density difference. With the SFC, the equilibrium populations f_i^{eq} become

$$\begin{aligned} f_0^{\text{eq}} &= w_i\rho\left[1 - \frac{c_s^2\boldsymbol{v}^2}{2c_s^4}\right] + \sum_{i=1}^{8} w_i\Delta\rho, \\ f_i^{\text{eq}} &= w_i\rho\left[1 + \frac{\boldsymbol{v}\cdot\boldsymbol{c}_i}{c_s^2} + \frac{(\boldsymbol{v}\cdot\boldsymbol{c}_i)^2 - c_s^2\boldsymbol{v}^2}{2c_s^4}\right] - w_i\Delta\rho \quad (i = 1, \ldots, 8). \end{aligned} \tag{3.12}$$

This correction essentially acts as a zeroth-order correction on the flux, enough to capture the density difference. Although in Fig. 3.3 we implicitly assume that the solid has higher density ($\rho_s > \rho_f$), the SFC remains the same for lighter solids ($\rho_s < \rho_f$).

3.2.4 Fluid–structure interaction

In this paper, we limit the FSI configuration to deformable solids immersed within fluids (Fig. 3.1(B)). Solids are modeled as incompressible neo-Hookean material [227], and fluids as quasi-incompressible. Under this assumption, we can decompose the Cauchy stress $\boldsymbol{\sigma}$ into a pressure field p and a deviatoric stress $\boldsymbol{\tau}$ for both phases. In Section 3.2.1 and Section 3.2.2, we have discussed how to simulate solids and fluids respectively. For solids, we use the reference map field $\boldsymbol{\xi}$ to construct the deformation gradient tensor $\boldsymbol{F}$ and the deviatoric solid stress $\boldsymbol{\tau}_s$,

$$\boldsymbol{\tau}_s = \mathbf{f}(\boldsymbol{F}) = G\left(\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}} - \frac{1}{3}\mathbf{1}\left(\mathrm{tr}\left(\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}\right) + 1\right)\right), \tag{3.13}$$

where $\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}$ is the left Cauchy–Green deformation tensor and G is the small-strain shear modulus. For fluids, we use the LB populations to reconstruct ρ and $\boldsymbol{v}$ without calculating the fluid stress. For consistency in notation, we also list the deviatoric fluid stress $\boldsymbol{\tau}_f$, which obeys the Newtonian fluid assumption,

$$\boldsymbol{\tau}_f = \mu_f\left(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^{\mathsf{T}}\right), \tag{3.14}$$

where the kinematic viscosity is $\nu_f = \mu_f/\rho_f$. In the LB method, we could compute the fluid stress locally as the non-equilibrium populations $f_i^{\text{neq}} = f_i - f_i^{\text{eq}}$; but it is not needed in most simulations.

We use a level set function $\phi(\boldsymbol{x}, t)$ [77, 78] to represent the solid geometry. The signed distance to the solid–fluid interface follows the convention that $\phi < 0$ in the solid and $\phi > 0$ in the fluid. To form a continuous description of the solid–fluid interface, we build a smooth transition between the solid and fluid phases. This transition region is called the *blur zone*, which can be realized through a smoothed Heaviside function $H_\varepsilon(\phi)$ with a transition region of width 2ε following the IncRMT implementation [98]:

$$H_\varepsilon(\phi) = \begin{cases} 0 & \text{if } \phi \le -\varepsilon \quad \text{(pure solid)}, \\ \dfrac{1}{2}\left[1 + \dfrac{\phi}{\varepsilon} + \dfrac{1}{\pi}\sin\left(\dfrac{\pi\phi}{\varepsilon}\right)\right] & \text{if } |\phi| < \varepsilon \quad \text{(blur zone)}, \\ 1 & \text{if } \phi \ge \varepsilon \quad \text{(pure fluid)}. \end{cases} \tag{3.15}$$

This Heaviside function is twice-differentiable and has been used frequently for smooth transitions[228–231]. The blur zone is equivalent to the no-slip boundary conditions on the solid–fluid interface.

We define the global deviatoric stress $\boldsymbol{\tau}$ as a smooth transition between the solid stress and the fluid stress:

$$\boldsymbol{\tau} = H_\varepsilon(\phi)\boldsymbol{\tau}_f + (1 - H_\varepsilon(\phi))\,\boldsymbol{\tau}_s. \tag{3.16}$$

Similarly, the global density ρ forms a smooth transition between the solid density and the fluid density:

$$\rho = H_\varepsilon(\phi)\rho_f + (1 - H_\varepsilon(\phi))\,\rho_s. \tag{3.17}$$

The reference map field $\boldsymbol{\xi}(\boldsymbol{x}, t)$ is defined only within the solid region ($\phi < 0$). To perform the Heaviside calculation in Eq. (3.16), we need the deviatoric solid stress $\boldsymbol{\tau}_s$ in the second half of the blur zone ($0 < \phi < \varepsilon$), which can be computed by extrapolating $\boldsymbol{\xi}$ out of regions enclosed by the solid–fluid interface ($\phi = 0$). We define the extrapolation zone with a width w no smaller than $1.5\Delta x^* + \sqrt{2}\varepsilon$, and use a least-square regression procedure to reconstruct the extrapolated reference map values [98]—see Section 3.3.2 for further details.

The LBRMT takes a mesoscopic approach to solving the smoothed Cauchy momentum equation:

$$\rho\left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v}\cdot\nabla)\,\boldsymbol{v}\right) = \underbrace{-\nabla p + \nabla\cdot\boldsymbol{\tau}_f}_{\substack{\text{background fluid}\\ \text{stress of LB nodes}}} + \underbrace{\nabla\cdot\left[\left(1-H_\varepsilon(\phi)\right)\boldsymbol{\tau}_s\right]}_{\substack{\text{large-deformation}\\ \text{solid stress}}} + \underbrace{\left(1-H_\varepsilon(\phi)\right)\boldsymbol{f}_{\text{ext},s}}_{\substack{\text{external forces}\\ \text{on solid only}}} + \underbrace{H_\varepsilon(\phi)\boldsymbol{f}_{\text{ext},f}}_{\substack{\text{external forces}\\ \text{on fluid only}}}. \tag{3.18}$$

For a pure fluid node, Eq. (3.18) simplifies to the Navier–Stokes equations. Whereas for a pure solid node, Eq. (3.18) becomes the Cauchy momentum equation with an additional fluid stress term. The additional term can be viewed as an artificial viscous stress onto the solid node. Numerically speaking, its presence adds damping to simulation and prevents numerical instability. Compared to the IncRMT [98], whose implementation additionally incorporates a similar artificial viscous stress inside the solid region, the LBRMT obtains this artificial viscous stress as a natural outcome of the LB method. In the current LBRMT, this stress is tied to fluid viscosity rather than being tunable, but it could be altered in future implementations, such as modifying the equilibrium populations to factor out the fluid stress [218].

Since all nodes in the LBRMT are structured as LB nodes, we only need to calculate the divergence of

the solid stress $\nabla \cdot \boldsymbol{\tau}_s$ and not that of the fluid stress $\boldsymbol{\tau}_f$. The effect of $\boldsymbol{\tau}_f$ is built into the first moment in the LB method—no need to explicitly calculate the deviatoric fluid stress for the global deviatoric stress. Since the divergence of stress has the same units (L/T^2) as the force density, we can smoothly combine the divergence of the solid stress $\nabla \cdot \boldsymbol{\tau}_s$, external force densities on solids $\boldsymbol{f}_{\text{ext},s}$ and on fluids $\boldsymbol{f}_{\text{ext},f}$ into a force density $\boldsymbol{F}$:

$$\boldsymbol{F} = \nabla \cdot [(1 - H_\varepsilon(\phi))\, \boldsymbol{\tau}_s] + (1 - H_\varepsilon(\phi))\, \boldsymbol{f}_{\text{ext},s} + H_\varepsilon(\phi) \boldsymbol{f}_{\text{ext},f}. \tag{3.19}$$

This force density $\boldsymbol{F}$ is then passed into Eq. (3.7) as the macroscopic external force density, which can be rewritten into the LB populations using Eq. (3.8). Finally, we arrive at a general formulation to include the effect of large-deformation solid stress into LB populations f_i. Eq. (3.19) is the cornerstone of the LBRMT: It connects the solid stress computed with the RMT to the LB external force density.

3.3 Numerical implementation

The LBRMT essentially combines two Eulerian methods onto one fixed computational grid. The solid update follows the IncRMT [98]: We first update the reference map field $\boldsymbol{\xi}$ via advection, then extrapolate $\boldsymbol{\xi}$ and update the solid–fluid interface by relabeling the solid and the fluid nodes based on the new signed distance values. We then calculate the solid stress $\boldsymbol{\tau}_s$, and finally pass the divergence $\nabla \cdot \boldsymbol{\tau}_s$ as external force densities to the LB method. The fluid update follows the LB routines [76]: We update the global density ρ and velocity $\boldsymbol{v}$ fields as moments of populations f_i. We summarize the LBRMT in Algorithm 1, where blue represents the LB routines, red represents the RMT routines, and purple

represents hybrid routines. The main loop involves ten steps. It can be considered as a standard LB fluid solver coupled with the RMT to calculate external force densities. Here we pay special attention to the three purple steps (11, 12, and 15) as they carry the essence of the LBRMT: These two steps incorporate a smooth description of both solids and fluids, as well as a unified implementation of the no-slip solid–fluid interface using the blur zone.

Algorithm 1: The LBRMT pseudocode.

```
1  Begin
2      Initialize the global density field ρ, global velocity field v_0, and populations f_i ;
3      Initialize the solid reference map field ξ_0 ;
4      Label the solid and fluid nodes using the level set function φ(ξ_0) ;
5      LOOP ρ^{n+1}, v^{n+1}, ξ^{n+1}, f_i^{n+1} ← ρ^n, v^n, ξ^n, f̂_i^n, φ
6          Update the reference map ξ^{n+1} ;
7          Extrapolate the reference map values in the extrapolation zone ;
8          Relabel the fluid and solid nodes in the extrapolation zone using φ(ξ^{n+1}) ;
9          Compute the divergence of solid stress ∇ · τ_s^{n+1} ;
10         Calculate the equilibrium populations f_i^{eq n} and collision operators Ω_i^n ;
11         Calculate the smoothed external force densities F_i^n ;
12         Calculate the post-collision populations f̂_i^n ;
13         Apply the wall boundary conditions ;
14         Stream f̂_i^n to neighboring lattices to update f_i^{n+1} ;
15         Compute updated ρ^{n+1}, and v^{n+1} as moments ;
16 end
```

The `LBRMT` software code is custom implemented in `C++` and multithreaded with `OpenMP` for parallelization. We denote the simulation domain with length L and height H, divided into an $n_x \times n_y$ grid of nodes with equal grid spacing $\Delta x^* = \Delta y^*$. Two extra layers of nodes are padded to each domain direction for considerations of boundary conditions and second-order stencils. We use subscripts i and

j to represent x and y indices for $i = -2, \ldots, n_x + 1$ and $j = -2, \ldots, n_y + 1$. We use superscript n to denote timestep and advance the simulation from timestep n to $n + 1$ with interval Δt^*. (We drop the asterisk superscript in subsequent subsections for simplicity of notation.) Each node (i, j) stores simulation variables (Fig. 3.7(B)) including density $\rho_{i,j}$, velocity $\boldsymbol{v}_{i,j}$, LB populations f_i, LB force densities F_i, and a custom-developed data structure `multimaps`—holding the reference map field $\boldsymbol{\xi}_{i,j}$ and the corresponding level set value $\phi_{i,j}$ for solid nodes; see Section 3.3.6 for details. Certain temporary variables are instantiated between nodes (*i.e.* half-edge) only for solid stress calculation purposes. Following the FSI configuration in Fig. 3.1(B), we employ a level set function ϕ to denote the solid geometry. The blur zone is centered at the solid–fluid interface ($\phi = 0$) with a half-width ε and an extrapolation zone of l layers. We initialize a reference map field $\boldsymbol{\xi}_0$ within the solid region (including the extrapolation zone), a global density field ρ, and a global velocity field $\boldsymbol{v}_0$—we further discuss the advantages of one global velocity field for multi-body contact in Section 3.3.6. We summarize relevant simulation parameters and variables in Table 3.2.

Simulation parameter	**Symbol**	**Dimension**
Reynolds number	Re	1
Density	ρ	M/L^3
Kinematic viscosity	ν	L^2/T
Shear modulus	G	$M/(LT^2)$
Gravitational constant	g	L/T^2
Relaxation time	τ	T
Simulation time	t	T

Table 3.2: Relevant LBRMT simulation parameters with their symbols and physical dimensions. M, L, T represent units of mass, length, and time. Details about unit conversions and parameter choices of the simulation variables and parameters are in Appendix B.1.

3.3.1 Reference map advection

We first update the reference map field $\boldsymbol{\xi}$ in the solid region ($\phi < 0$; not including the blur zone). Denote the components of the reference map field and the velocity field as $\boldsymbol{\xi} = (X, Y)$ and $\boldsymbol{v} = (u, v)$, the advection equation of the reference map field in Eq. (3.4) can be discretized as

$$\boldsymbol{\xi}_{i,j}^{n+1} = \boldsymbol{\xi}_{i,j}^{n} - (u\partial_x + v\partial_y)\,\boldsymbol{\xi}_{i,j}^{n}. \tag{3.20}$$

An upwinding second-order finite difference method (Fig. 3.5(A)) is used for calculating the derivatives:

$$\begin{aligned}
\frac{\partial \boldsymbol{\xi}_{i,j}}{\partial x} &= \begin{cases} \dfrac{3\boldsymbol{\xi}_{i,j} - 4\boldsymbol{\xi}_{i-1,j} + \boldsymbol{\xi}_{i-2,j}}{2\Delta x} & \text{if } u > 0, \\ \dfrac{-3\boldsymbol{\xi}_{i,j} + 4\boldsymbol{\xi}_{i+1,j} - \boldsymbol{\xi}_{i+2,j}}{2\Delta x} & \text{if } u < 0, \end{cases} \\
\frac{\partial \boldsymbol{\xi}_{i,j}}{\partial y} &= \begin{cases} \dfrac{3\boldsymbol{\xi}_{i,j} - 4\boldsymbol{\xi}_{i,j-1} + \boldsymbol{\xi}_{i,j-2}}{2\Delta y} & \text{if } v > 0, \\ \dfrac{-3\boldsymbol{\xi}_{i,j} + 4\boldsymbol{\xi}_{i,j+1} - \boldsymbol{\xi}_{i,j+2}}{2\Delta y} & \text{if } v < 0. \end{cases}
\end{aligned} \tag{3.21}$$

3.3.2 Reference map extrapolation and level set update

We extrapolate the reference map field $\boldsymbol{\xi}$ out of the solid region so that each node within the blur zone has updated $\boldsymbol{\xi}$ values when computing the solid stress. The extrapolation routine is based on fitting a weighted least-squares regression [98, 99] instead of partial differential equation (PDE)-based methods [81, 96, 232]. This alternative reduces the complexity of explicitly keeping track of the level set values

ϕ and the reference map field $\boldsymbol{\xi}$ in the extrapolation. The extrapolation zone is wider than the blur zone, defined with a width no smaller than $1.5\Delta x + \sqrt{\varepsilon}$ to ensure valid second-order stencils in solid stress calculations. The first layer in the extrapolation zone is marked by extending one node in four directions (up, down, left, right) of the exterior solid solids (Fig. 3.4(A)). Each subsequent layer of index l is marked one-by-one by extending one node outward from previous layers. The extrapolation procedure (Fig. 3.4) follows that in the IncRMT [98] and RMT3D [99], starting at the target node (i, j) at position $\boldsymbol{x} = (x, y)$ in the first layer $l = 1$:

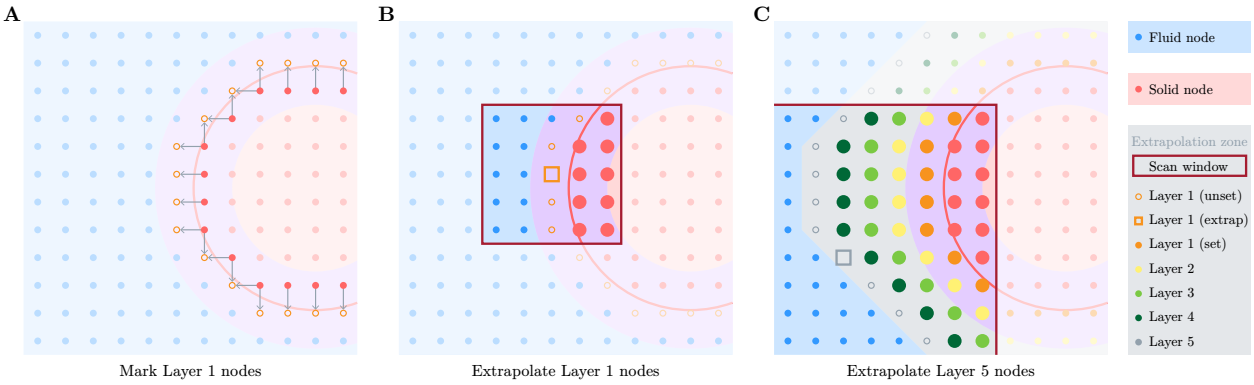

Figure 3.4: Illustration of the reference map extrapolation. The extrapolation procedure starts from the first layer in the extrapolation zone and then moves outward to the next layer after all nodes have been extrapolated. **(A)** The first layer $l = 1$ is initialized by extending one node in four directions (up, down, left, right) of the exterior solid solids (red nodes). Layer 1 nodes have only been marked with their positions, meaning they are unset (empty orange nodes) with no extrapolated reference map values. After marking all nodes in Layer 1, we proceed to compute their extrapolated reference map values $\boldsymbol{\xi}_{\text{extrap}}$. **(B)** For the target node (i, j) (empty orange square), we initialize a scan window centered at it with a half-width $r = 2$. Within the scan window, we compute its extrapolated reference map values using the valid nodes (enlarged red nodes). **(C)** We perform the extrapolation layer by layer. In the scenario when the linear map is ill-defined or we find fewer than three valid nodes within the scan window, we gradually increase the scan window half-width by 1. For an example case of a target node in Layer 5 (empty gray square), its scan window has been increased to a half-width $r = 5$ to include more valid nodes from previous layers (enlarged red, orange, yellow, light and dark green nodes) in the extrapolation procedure.

1. Initialize a scan window centered at node (i, j) with an initial half-width $r = 2$. Count the number of valid nodes (i', j') at position $\boldsymbol{x}' = (x', y')$ in the scan window (Fig. 3.4(B)) such that

$r_i = |i - i'| \leq r$ and $r_j = |j - j'| \leq r$. If there are fewer than three valid nodes within the scan window, we increase the half-width r by 1 to include more valid nodes (Fig. 3.4(C)) and repeat Step (1). Note that a valid node is either a solid node or in the previous layers $l_{(i',j')} < l_{(i,j)}$ with existing reference map values.

2. Use weighted least-squares regression to fit a linear map $\boldsymbol{\xi}_{\text{extrap}}(x, y) = w\,(Ax + By + C)$ with available reference map values of enclosed valid nodes and their positional indices. We also employ coordinate-based weighting with an exponential decaying kernel centered at (i, j). In the first two layers ($l \leq 2$), we encode complex geometric information with an approximated surface normal $\widehat{\boldsymbol{n}}_e$ of ϕ and a physical distance vector $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}'$ between the target node and the valid node. The weighting w is defined as

$$w = \begin{cases} \max\left(0, \dfrac{\boldsymbol{d} \cdot \widehat{\boldsymbol{n}}_e}{|\boldsymbol{d}|}\, 2^{-(r_i + r_j)}\right) & \text{if } l \leq 2, \\ 2^{-(r_i + r_j)} & \text{if } l > 2. \end{cases} \tag{3.22}$$

   If the linear map is ill-defined, we increase the half-width r by 1 and repeat Step (1) and (2);

3. Assign $\boldsymbol{\xi}_{\text{extrap}}$ to be the reference map value at the target node (i, j).

Once all nodes in the first layer have been processed, we move outward to the next layer. After completing the extrapolation of all layers, we need to re-calculate the level set values ϕ for all nodes in the extrapolation zone because they may change from solid to fluid, and vice versa. This extrapolation proce-

dure offers a smooth transition between the solid and fluid phases, requiring no additional bookkeeping about density, velocity, or no-slip solid–fluid interface. It also provides valid reference map values for the subsequent solid stress computation (Section 3.3.3) because we use a second-order method to compute gradients (Eq. (3.24)), *i.e.* we need at least four additional nodes in each direction (starting the count from the exterior solid nodes at $\phi = 0$).

3.3.3 Solid stress computation

Our goal is to construct $\nabla \cdot \boldsymbol{\tau}_s$, which is used as the macroscopic external force density $\boldsymbol{F}$ in Eq. (3.7). At node (i, j), this divergence is calculated based on four half-edge solid stresses $[\boldsymbol{\tau}_s]_{i-\frac{1}{2},j}$, $[\boldsymbol{\tau}_s]_{i+\frac{1}{2},j}$, $[\boldsymbol{\tau}_s]_{i,j-\frac{1}{2}}$ and $[\boldsymbol{\tau}_s]_{i,j+\frac{1}{2}}$ respectively to the left, right, bottom, and top of the node. Each half-edge solid stress is com-

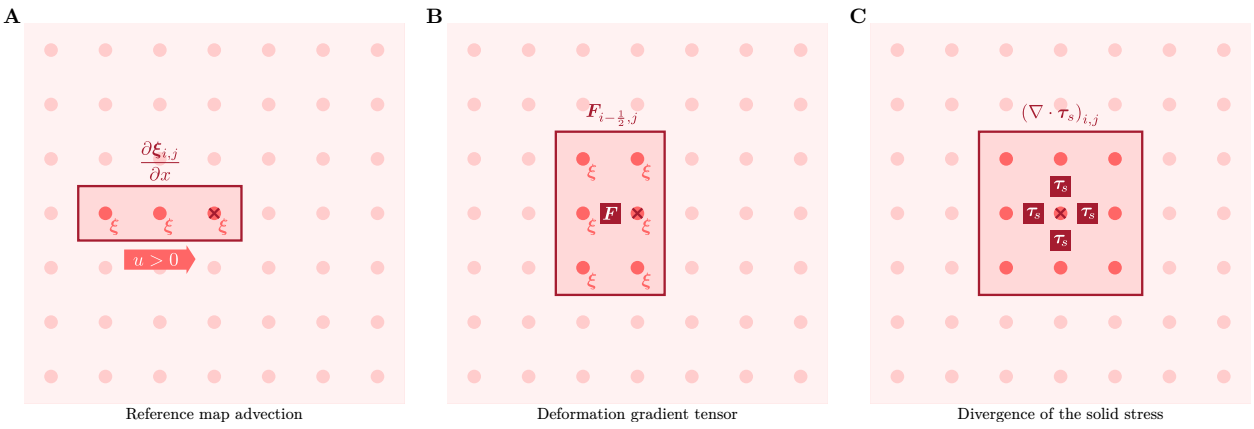

Figure 3.5: Stencils for the reference map advection and solid stress computation. There are three key steps to compute the divergence of the solid stress at node (i, j): **(A)** We first calculate the gradients of the reference map field $\partial \boldsymbol{\xi} / \partial \boldsymbol{x}$ from the reference map advection; **(B)** then we build the half-edge deformation gradient tensor $\boldsymbol{F}$ using the computed gradients; **(C)** after converting the half-edge $\boldsymbol{F}$ into half-edge solid stress $\boldsymbol{\tau}_s = \mathsf{f}(\boldsymbol{F})$ with a constitutive relation f, we use the four half-edge $\boldsymbol{\tau}_s$ around node (i, j) to construct $\nabla \cdot \boldsymbol{\tau}_s$. Each of these steps corresponds to a panel illustrating the stencils required for discretization, with the example of the left half-edge solid stress: **(A)** Three nodes are used for reference map advection in the x direction when $u > 0$, **(B)** six nodes are used for constructing the left half-edge deformation gradient tensor, and **(C)** nine nodes are involved for all four half-edge solid stresses.

puted from the Jacobian of the reference map field $\boldsymbol{\xi}$ with a second-order finite difference scheme [98]. For example, to compute the left half-edge solid stress $[\boldsymbol{\tau}_s]_{i-\frac{1}{2},j}$, the gradients involved (Fig. 3.5(B)) to calculate the Jacobian are

$$\begin{aligned}\left(\frac{\partial \boldsymbol{\xi}}{\partial x}\right)_{i-\frac{1}{2},j} &= \frac{\boldsymbol{\xi}_{i,j} - \boldsymbol{\xi}_{i-1,j}}{\Delta x}, \\ \left(\frac{\partial \boldsymbol{\xi}}{\partial y}\right)_{i-\frac{1}{2},j} &= \frac{\boldsymbol{\xi}_{i,j} + \boldsymbol{\xi}_{i-1,j+1} - \boldsymbol{\xi}_{i,j-1} - \boldsymbol{\xi}_{i-1,j-1}}{4\Delta y}.\end{aligned} \tag{3.23}$$

Since we focus on two-dimensional simulations, the Jacobian is denoted as a 2×2 matrix. The corresponding deformation gradient tensor is

$$\boldsymbol{F}_{i-\frac{1}{2},j} = \left(\left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}\right)_{i-\frac{1}{2},j}\right)^{-1}. \tag{3.24}$$

In the LBRMT, we model the solid phase as an incompressible neo-Hookean solid. For two-dimensional simulations, this constitutive relation for solid stress $\boldsymbol{\tau}_s = \mathbf{f}(\boldsymbol{F})$ follows a plane-strain formulation:

$$[\boldsymbol{\tau}_s]_{i-\frac{1}{2},j} = G\left(\boldsymbol{F}_{i-\frac{1}{2},j}\boldsymbol{F}^{\mathsf{T}}_{i-\frac{1}{2},j} - \frac{1}{3}\mathbf{1}\left(\mathrm{tr}\left(\boldsymbol{F}_{i-\frac{1}{2},j}\boldsymbol{F}^{\mathsf{T}}_{i-\frac{1}{2},j}\right) + 1\right)\right), \tag{3.25}$$

where G is the small-strain shear modulus. We think of two-dimensional simulations being infinitely extruded in the third dimension, thus the $+1$ term in Eq. (3.25) is originated from zero stretch in that third dimension. Similarly, we can compute the three other solid stresses using the same discretization

scheme. Once the four intermediate half-edge stresses are set (Fig. 3.5(C)), the divergence of the solid stress at node (i, j) is

$$[\nabla \cdot \boldsymbol{\tau}_s]_{i,j} = \frac{\left([\boldsymbol{\tau}_s]_{i+\frac{1}{2},j}\right)_x - \left([\boldsymbol{\tau}_s]_{i-\frac{1}{2},j}\right)_x}{\Delta x} + \frac{\left([\boldsymbol{\tau}_s]_{i,j+\frac{1}{2}}\right)_y - \left([\boldsymbol{\tau}_s]_{i,j-\frac{1}{2}}\right)_y}{\Delta y}, \tag{3.26}$$

where subscripts x and y represent the tensor components acting on the horizontal and vertical directions. Eq. (3.26) is then passed into the LB updates via Eq. (3.19) to build the external force density $\boldsymbol{F}$.

3.3.4 Density and velocity updates

The calculations of macroscopic quantities, *i.e.* the global density field ρ and global velocity field $\boldsymbol{v}$, follow the standard LB update routines. We first compute the equilibrium populations f_i^{eq} for all nodes with their respective density and velocity values using Eq. (3.12), where the density difference $\Delta\rho$ is defined in Eq. (3.11). For better code performance [76], we use the expanded forms of the nine equilibrium populations in Eq. (3.27) when calculating updates (and all subsequent calculations involving populations):

$$\begin{aligned}
f_0^{\text{eq}} &= \frac{2\rho}{9}\left(2 - 3\left(u^2 + v^2\right)\right) + \frac{5}{9}\Delta\rho, \\
f_1^{\text{eq}} &= \frac{\rho}{18}\left(2 + 6u + 9u^2 - 3\left(u^2 + v^2\right)\right) - \frac{1}{9}\Delta\rho, \\
f_2^{\text{eq}} &= \frac{\rho}{18}\left(2 + 6v + 9v^2 - 3\left(u^2 + v^2\right)\right) - \frac{1}{9}\Delta\rho, \\
f_3^{\text{eq}} &= \frac{\rho}{18}\left(2 - 6u + 9u^2 - 3\left(u^2 + v^2\right)\right) - \frac{1}{9}\Delta\rho, \\
f_4^{\text{eq}} &= \frac{\rho}{18}\left(2 - 6v + 9v^2 - 3\left(u^2 + v^2\right)\right) - \frac{1}{9}\Delta\rho,
\end{aligned} \tag{3.27}$$

$$
\begin{aligned}
f_5^{\mathrm{eq}} &= \frac{\rho}{36}\left(1+3\left(u+v\right)+9uv+3\left(u^2+v^2\right)\right)-\frac{1}{36}\Delta\rho,\\
f_6^{\mathrm{eq}} &= \frac{\rho}{36}\left(1-3\left(u-v\right)-9uv+3\left(u^2+v^2\right)\right)-\frac{1}{36}\Delta\rho,\\
f_7^{\mathrm{eq}} &= \frac{\rho}{36}\left(1-3\left(u+v\right)+9uv+3\left(u^2+v^2\right)\right)-\frac{1}{36}\Delta\rho,\\
f_8^{\mathrm{eq}} &= \frac{\rho}{36}\left(1+3\left(u-v\right)-9uv+3\left(u^2+v^2\right)\right)-\frac{1}{36}\Delta\rho.
\end{aligned}
$$

These equilibrium populations f_i^{eq} are then used to construct the collision operators $\Omega_i = -\frac{1}{\tau}\left(f_i - f_i^{\mathrm{eq}}\right)$. To include forces in the density and velocity updates, we first compute the macroscopic external force density $\boldsymbol{F}$ using Eq. (3.19), which is a smooth combination of all solid and fluid force densities. We then use Eq. (3.8) to discrete the macroscopic force density in the D_2Q_9 velocity space mesoscopically in Eq. (3.28):

$$
\begin{aligned}
F_0 &= \frac{4\rho}{9}\left(-3uF_x-3vF_y\right),\\
F_1 &= \frac{\rho}{9}\left(3(1-u)F_x-3vF_y+9uF_x\right),\\
F_2 &= \frac{\rho}{9}\left(-3uF_x+3(1-v)F_y+9vF_x\right),\\
F_3 &= \frac{\rho}{9}\left(3(-1-u)F_x-3vF_y+9uF_x\right),\\
F_4 &= \frac{\rho}{9}\left(-3uF_x+3(-1-v)F_y+9vF_x\right),\\
F_5 &= \frac{\rho}{36}\left(3(1-u)F_x+3(1-v)F_y+9(u+v)F_x+9(u+v)F_y\right),\\
F_6 &= \frac{\rho}{36}\left(3(-1-u)F_x+3(1-v)F_y+9(u-v)F_x+9(-u+v)F_y\right),\\
F_7 &= \frac{\rho}{36}\left(3(-1-u)F_x+3(-1-v)F_y+9(u+v)F_x+9(u+v)F_y\right),
\end{aligned}
\tag{3.28}
$$

$$F_8 = \frac{\rho}{36}\left(3(1-u)F_x + 3(-1-v)F_y + 9(u-v)F_x + 9(-u+v)F_y\right).$$

Having computed Ω_i and F_i, we can assemble the post-collision populations $\widehat{f_i}$ with

$$\widehat{f_i} = f_i + \Omega_i + \Delta t\left(1 - \frac{1}{2\tau}\right)F_i. \tag{3.29}$$

After applying the wall boundary conditions (Section 3.3.5) specific to the simulation setup, we stream $\widehat{f_i}$ to update the populations. Both solid and fluid (even blur-zone) nodes share the same macroscopic quantities calculations, where we use the zeroth and the first moments of the updated populations f_i to calculate the updated density ρ and velocity $\boldsymbol{u} = (u, v)$ for the next timestep:

$$\begin{aligned}
\rho &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8, \\
u &= \frac{1}{\rho}\left(f_1 + f_5 + f_8 - f_3 - f_6 - f_7\right) + \frac{1}{2\rho}\left(F_1 + F_5 + F_8 - F_3 - F_6 - F_7\right), \\
v &= \frac{1}{\rho}\left(f_2 + f_5 + f_6 - f_4 - f_7 - f_8\right) + \frac{1}{2\rho}\left(F_2 + F_5 + F_6 - F_4 - F_7 - F_8\right).
\end{aligned} \tag{3.30}$$

3.3.5 Wall boundary conditions

There are two types of wall boundary conditions defining the simulation domain: periodic (Fig. 3.6(A)) and no-slip (Fig. 3.6(B)). Since the domain is padded with two layers of nodes, they act as buffers for copying or reflecting fluid nodes. These buffer nodes streamline the implementation of boundary conditions,

with no need to separately compute populations at the wall boundaries and at the corners [233]. Only the inner buffer layer is used in the boundary conditions calculation (the outer one is used for second-order stencils calculations). This setup of periodic boundary conditions has no information loss, whereas the no-slip involves halfway bounce-back [214] to ensure second-order accuracy at the wall boundaries.

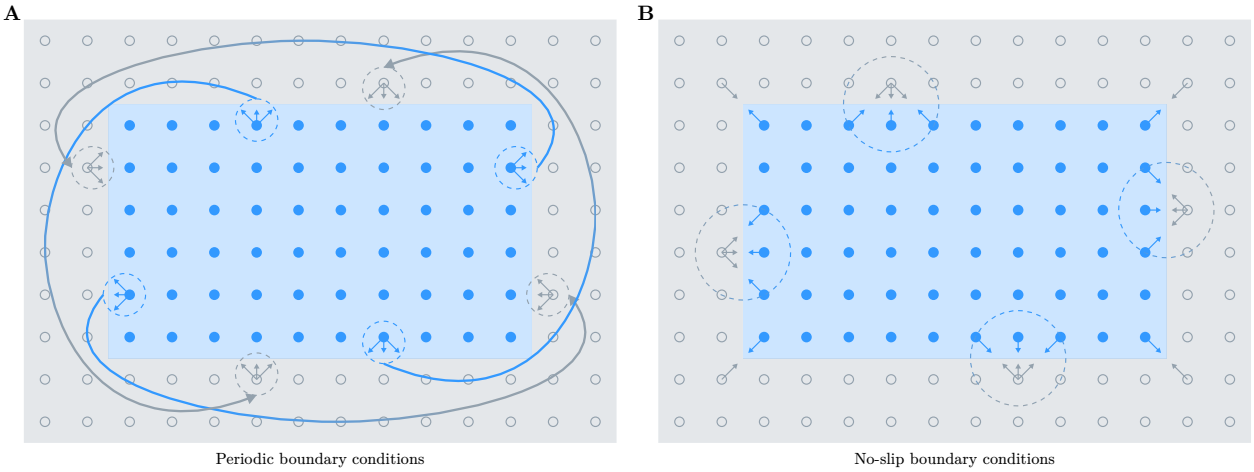

Figure 3.6: Diagram of wall boundary conditions. (A) The periodic boundary conditions on a simplified simulation domain. On the x-axis, the left periodicity is imposed by filling the first buffer on the left (grey arrows) with the outward populations from the rightmost column of fluid nodes (blue arrows), and the right periodicity is imposed by filling the first buffer on the right with the outward populations from the leftmost column of fluid nodes. The periodicity along the y-axis is analogous. **(B)** The no-slip boundary conditions on a simplified simulation domain. This direct on-node reflection bounces back the populations leaving the fluid region. The outward populations (blue arrows) at each wall boundary are copied as the opposite direction populations (grey arrows) in the corresponding buffer nodes. These populations are then streamed back to their original nodes, only reversed. The four corners of the first buffer layer need special handling since they have only one population associated with the fluid region.

3.3.6 Multi-body contact

Since the LBRMT simulation grid contains only one global velocity field, it is straightforward to define the collision of two or more objects as the overlaps of their geometries. To efficiently handle multi-body contact, we develop a custom data structure—`multimaps`—to store reference map fields of many solid

objects onto one node. Since all reference map fields are created only within their local solid (plus blur zone) region, one node can represent two or more solids. We identify each solid with an ID number, then organize all reference map field information (*e.g.* $\boldsymbol{\xi}$, ϕ, object ID) of each solid into a custom `ref_map` structure (Fig. 3.7(B)). Instead of carrying the reference map information of all solids directly, each node carries a `C++` pointer to `multimaps` (Fig. 3.7(B)), which is a list of `ref_map` objects present at this node. We use two integers, `counter_mmap` and `max_mmap`, to count the current number of `ref_map` objects (*i.e.* solids) and the maximum number of solids a node can hold. All simulation nodes are instantiated with a `multimaps` structure of length four.

For a fluid node, its counter `counter_mmap` remains zero. For a generic node associated with a solid (including its blur zone), we instantiate a `ref_map` object, append it to the `multimaps`, and advance `counter_mmap` by one. A node can contain `ref_map` objects associated with multiple solids. If the counter exceeds half of the `multimaps` length (*i.e.* `max_mmap`), we double the length to account for more solids present on the node. When a node is no longer part of a solid, we remove the associated `ref_map` object from the `multimaps`. This dynamic update of valid `ref_map` objects allows us to keep the length of `multimaps` short, typically limited to ten objects at most. The `multimaps` thus provides simplicity in collision detection when simulating hundreds of solids: We do not need exhaustive search over all solids for collision pairs; only need to search through the solids with IDs currently present in the `multimaps` of one node (Fig. 3.7(B)). It brings additional advantages in data storage: We do not need to store the reference map information of every solid on every node.

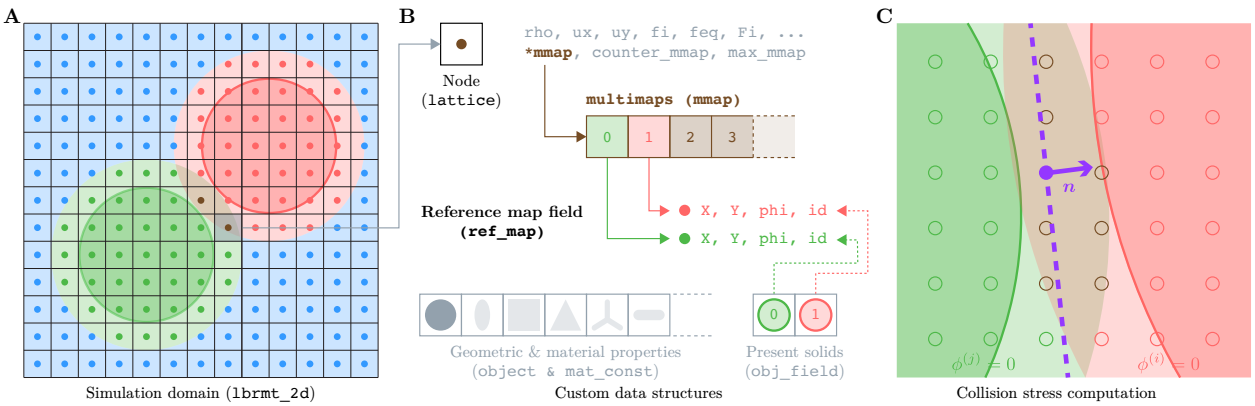

Figure 3.7: Schematics of simulation domain, custom data structures, and collision stress computation. The LBRMT code is designed to model multiple solids interacting with fluids on the same computational grid. **(A)** Example of simulation nodes when two solids come into contact. Blue nodes represent the fluid phase, red nodes represent solid i, and green nodes represent solid j. The light-red and light-green areas represent halves of the blur zone outside the solid region, $0 < \phi^{(i)} < \varepsilon$ and $0 < \phi^{(j)} < \varepsilon$, respectively. We identify collision in the overlapping light-brown area. Brown nodes represent the collision nodes. **(B)** Custom data structures involved for a simulation node. Each node is instantiated as a custom `lattice` object, which carries variables like `rho` for density, `ux` and `uy` for velocity components, `fi`, `feq`, `Fi` for LB updates. It also holds a C++ pointer to `*mmap`—a custom `multimaps` structure that can contain a list of custom `ref_map` objects, an integer `counter_mmap`—the current number of `ref_map` objects (*i.e.* solids) present on the node (red and green boxes), and an integer `max_mmap`—the maximum number of `ref_map` objects the node can currently contain (solid brown boxes). Each `ref_map` object contains the reference map field information like the components `X` and `Y`, level set value `phi` and the corresponding object ID `id`. The object ID is respectively associated with the present solids on the simulation domain, contained in a custom C++ array `obj_field`. The material and geometric properties of the solids are specified through two custom structures, `mat_const` and `object`, where users can define the solid density and softness, as well as arbitrary shapes using level set functions (currently supporting circles, ellipses, squares, triangles, rotors, and rods.) **(C)** We identify collision nodes by testing whether the counter `counter_mmap` is larger than one , *i.e.* there are at least two `ref_map` objects in the `mmap`. A collision stress is added to the collision node to mimic a repulsive force pushing the solids apart, which is defined using the unit normal vector $\boldsymbol{n}$ (purple arrow) between solids i and j.

We identify collision when the blur zone of two or more objects overlap, which means one collision node (solid or blur zone) has at least two `ref_map` objects (Fig. 3.7(B)), or equivalently in C++ implementation `counter_mmap>1`. In the occurrence of collision, we add a local collision stress $\boldsymbol{\tau}_{\text{col}}$ on the collision node to push the solids apart following the IncRMT [98] implementation:

$$\boldsymbol{\tau}_{\text{col}} = -\eta \min\left[f\left(\phi^{(i)}\right), f\left(\phi^{(j)}\right)\right]\left(G^{(i)} + G^{(j)}\right)\left(\boldsymbol{n}\otimes\boldsymbol{n} - \frac{1}{2}\mathbf{1}\right), \tag{3.31}$$

where η is a dimensionless constant to tune the collision effects between solids, $G^{(i)}$ are the shear moduli of object i, and f is a function of the contact force between two colliding solids:

$$f(x) = \begin{cases} \frac{1}{2}\left(1 - \frac{x}{\varepsilon}\right) & \text{if } \phi < \varepsilon, \\ 0 & \text{if } \phi \geq \varepsilon. \end{cases} \tag{3.32}$$

The unit normal vector $\boldsymbol{n}$ (Fig. 3.7(C)) between a pair of solids i and j indicates the direction of the repulsive contact force, which can be computed with finite-difference schemes:

$$\boldsymbol{n} = \frac{\nabla\left(\phi^{(i)} - \phi^{(j)}\right)}{\|\nabla\left(\phi^{(i)} - \phi^{(j)}\right)\|_2}. \tag{3.33}$$

Similar to the IncRMT [98], we also modify the global stress $\boldsymbol{\tau}$ to reflect the solid fraction $\lambda^{(i)} = 1 - H_\varepsilon\left(\phi^{(i)}\right)$ of each object i on one node:

$$\boldsymbol{\tau} = \begin{cases} \boldsymbol{\tau}_f + \sum_i \lambda^{(i)} \boldsymbol{\tau}_s^{(i)} & \text{if } \sum_i \lambda^{(i)} \leq 1, \\ \frac{\sum_i \lambda^{(i)} \boldsymbol{\tau}_s^{(i)}}{\sum_i \lambda^{(i)}} & \text{if } \sum_i \lambda^{(i)} > 1. \end{cases} \tag{3.34}$$

When the simulation only has one solid, Eq. (3.34) simplifies to Eq. (3.16). When the simulation has multiple solids, Eq. (3.34) collects individual solid stress to the global stress based on the solid fraction.

3.4 Results

Since we aim to showcase the diverse applications of the numerical method—not limited to a specific application to one physical problem—we nondimensionalize all simulation parameters and variables in all presented results. We follow the LB simulation conventions and set the fluid density ρ_f, timestep Δt^*, and grid spacing Δx^* to be $\rho_f = 1$, $\Delta t^* = 1$, and $\Delta x^* = \Delta y^* = 1$ for equal grid spacing. To convert the nondimensional simulations to physical reality, we multiply the results and variables by the corresponding density, time, and length scales. We refer readers to Appendix B.1 for conversions between physical and dimensionless LB units and choices for simulation parameters, Appendix B.4 for timing results and code performance, and Appendix B.5 for simulation movies. For subsequent results, we set the blur zone half-width $\varepsilon = 1.5$ and 11 extrapolation zone layers to accommodate all solid deformation cases.

In the following subsections, we highlight the versatility of the LBRMT in simulating the interactions of multiple solids in fluids undergoing large deformation through representative simulation examples. We first establish the baseline accuracy of our method with a benchmark example in Section 3.4.1. We then start with an example of anchored rotors in Section 3.4.2 to demonstrate the robustness of our method in modeling contact, bending, and stretching of soft solids. In Section 3.4.3, we use examples of settling and floating of different-shaped solids in fluid to highlight our method in modeling solids of different densities with the smooth flux correction (Section 3.2.3). Moving to Section 3.4.4, we elevate the complexity of single solid settling/floating by adding more solids, demonstrating our method in efficiently simulating hundreds of solids while capturing their mixing behavior. In particular, we study whether

softness enhances mixing rate, and our findings indicate that softness can assist in more efficient mixing.

3.4.1 Comparison to benchmark example

The lid-driven cavity is a classic benchmark problem in computational fluid dynamics that simulates fluid flow in a square cavity with the top lid moving at a constant speed. Its simple, yet non-trivial geometric setup has led to extensive experimental and numerical studies in both two and three dimensions [234–236] to understand the vortex structures formed by the fluid flow at different Reynolds numbers. Lid-driven cavity simulations thus enable researchers to test and compare the accuracy and efficiency of different numerical methods for solving the Navier–Stokes equations, ranging from finite-difference [237, 238], to multigrid [235, 239] and the LB [240, 241] methods. Although the results on deformable solids immersed in lid-driven cavity flow remain comparatively limited, Zhao *et al.* [242] simulated a deformable disk in a two-dimensional square lid-driven cavity, which has been widely used to validate later works [89, 97, 99, 243, 244].

We first validate the LBRMT as a fluid solver by comparing with the benchmarks of Ghia *et al.* [235] and Hou *et al.* [240] for lid-driven cavity without a solid. Our results match well with velocity profiles along the geometric center (see Appendix B.2). We then introduce a neutrally buoyant deformable solid into the cavity. The simulation parameters are matched with Zhao *et al.* [242], where a circle of radius $0.2L$ and shear modulus $G = 0.1$ is centered at $(0.6L, 0.5L)$ in a square lid-driven cavity flow of size $L \times L$ and the Reynolds number $Re = 100$. The top wall moves at a lid-driven velocity, and other stationary walls have no-slip boundary conditions. We do not apply repulsive force when the solid is

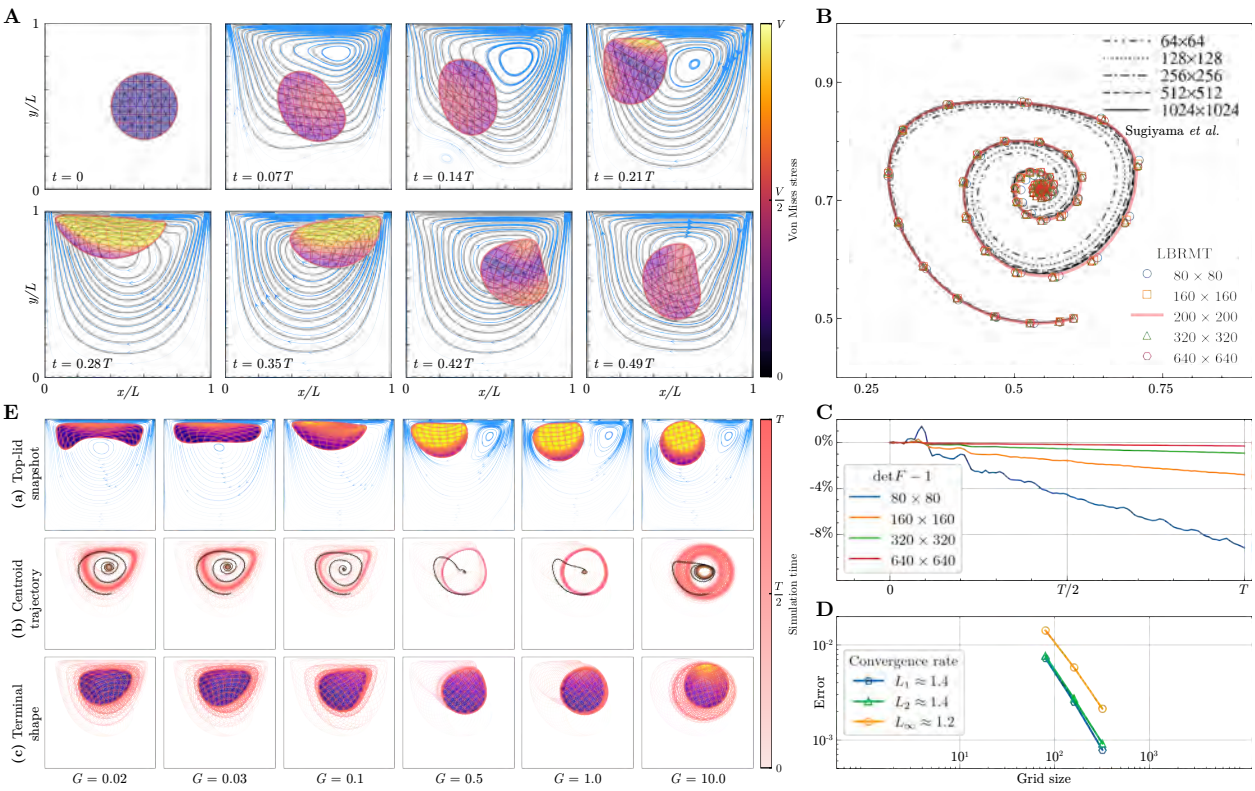

Figure 3.8: Benchmark example of a soft solid in a lid-driven cavity. **(A)** Snapshots of solid deformation in lid-driven flow. We overlay the solid–fluid interface (thick red lines), reference map contours (thin red lines, which indicate how the solid deforms), and streamlines (blue contours) of the LBRMT simulations with 75% transparency on Fig. 16 by Zhao *et al.* [242]. Our results match with the benchmark streamlines (black solid lines) and solid outlines (black triangular mesh). The colors inside the solid represent the intensity of von Mises stress, which measures how much is the solid sheared. The colormap is normalized by the maximum stress value $V_{\text{max}} \approx 0.004$ in the time range but clipped at $V = 0.001$. Simulation parameters are $(L, \tau, \mathit{Re}, \rho_f, \rho_s, G, T) = (200, 1.0, 100, 1.0, 1.0, 0.1, 40000)$. **(B)** Solid centroid trajectory at $G = 0.1$ on different grid sizes overlaid on Fig. 11 by Sugiyama *et al.* [89]. In addition to good agreements, our results match to their finest grid results on smaller grids. **(C)** Volumetric deviation on different grid sizes, from about 9.2% on the coarsest grid to less than 1% on finer grids. **(D)** Spatial convergence rate of the LBRMT with 640×640 results as the reference. **(E)** Snapshots of solid (a) deformation at the lid top, (b) centroid trajectory, and (c) terminal shape with softness $G \in [0.02, 0.03, 0.1, 0.5, 1.0, 10.0]$. The colormap is normalized by the maximum stress value of all cases $V_{\text{max}} \approx 0.17$ (at $G = 10.0$) in the time range but clipped at $V = 0.003$.

close to the top lid to capture the effect of lubrication forces. Given the parameters in Zhao *et al.* [242] are dimensionally different from the LB units, we first convert their dimensionless parameters to physical units and then convert back to LB units.

Fig. 3.8A confirms the LBRMT results are in good agreement with the streamlines and the solid out-

lines in Zhao *et al.* [242], who used an overlapping Lagrangian mesh of 73 triangles to compute the solid elastic stresses using a fixed-mesh algorithm. Since the LBRMT requires only one fixed Eulerian grid to represent large solid deformation, this comparison has demonstrated the accuracy and simplicity of our method in simulating deformable solids immersed in fluids. We also compare the trajectory of the solid centroid (Fig. 3.8B) with that by Sugiyama *et al.* [89], whose used a fully Eulerian finite-difference approach to discretizing the solid stress. The LBRMT results also match with their highest-resolution centroid trajectory on a smaller grid.

To analyze the volume conservation of solids, we calculate the volumetric deviation $\|\det \boldsymbol{F} - 1\|$ at different grid resolutions $L \in [80, 160, 320, 640]$ at the same physical time. Coarser grids have larger compressibility errors [76], thus we observe a maximum of 9.2% decrease (80×80 grid with simulation time $T = 20000\Delta t$). As we refine the grid size, we report a minimum of 0.3% decrease (640×640 grid with simulation time $T = 1280000\Delta t$) (Fig. 3.8C). In addition, the LBRMT convergence rate is approximately 1.4 (Fig. 3.8D), consistent with the FSI convergence rates reported in Rycroft *et al.* [98]. Even though the LB method [13] and the RMT [98] are respectively second-order methods, the smoothed transition between the solid and fluid phases in the LBRMT creates a blur zone of size $\mathcal{O}(\Delta x)$, which lowers the convergence rate. We refer readers to Appendix B by Rycroft *et al.* [98] for a more detailed analysis of the RMT convergence and accuracy.

We also report our results of lid-driven cavity with a solid at different softness to expand this benchmark example. In particular, we span the solid shear modulus G over a range of values and examine (a) its shape

close to the lid top, (b) the centroid trajectory, and (c) the terminal shape (Fig. 3.8E). When the solid is very soft (*e.g.* $G = 0.02$), it is stretched across the entire top lid due to the initial vortical motion of the fluid, then moves with the vortex until stabilizing at the vortex center. Whereas a stiffer solid retains its shape and does not travel across the top lid, but is stopped by the vortex formed at the top lid.

3.4.2 Rotating

Similar to propellers or marine life tentacles, flexible rotors can excite surrounding fluid through rotational actuation. Such active fluid models can create thrust or generate flow, important in energy generation for marine propeller [245], aquatic locomotion [246], and wind turbine [247]. Here we focus

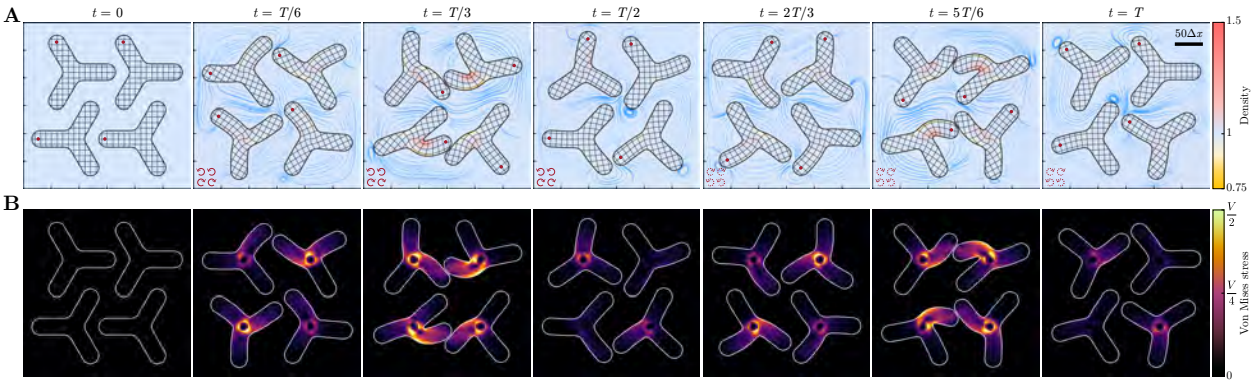

Figure 3.9: Bending, twisting, and stretching of anchored rotors. Four neutrally buoyant rotors with a prong length of $0.2L$ and period T are positioned in a confined fluid box of size $L \times L$. Anchoring forces are applied at each rotor center area with a radius of $0.025L$ to induce twisting, resulting in paired rotor rotations in opposite directions. During the first $T/2$ of the simulation, the top two rotors rotate counter-clockwise while the bottom two clockwise, for a full revolution of 2π. In the subsequent $T/2$, anchoring forces reverse, causing the top two to rotate clockwise and the bottom two to rotate counter-clockwise, for another 2π revolution. Simulation parameters are $(L, \tau, \rho_f, \rho_s, G, T) = (300, 1.0, 1.0, 1.0, 5.0, 45000)$. **(A)** Snapshots of density field and streamlines. Blue streamlines show fluid flow due to rotor motion, with streamline density signaling flow speed. Thick black lines are the solid–fluid interfaces and thin black lines are the reference map contours which illustrate rotor deformation. Colors represent the density field, where red indicates higher-density regions caused by compression due to rotor contact, and yellow indicates lower-density regions caused by stretching due to rotor bending. **(B)** Snapshots of the von Mises stress, with colormap normalized by the maximum stress value $V \approx 0.17$ in the time range. Highlighted areas indicate increased shear, reflecting rotor deformation intensity when they come into contact and slide past each other.

on a simplified scenario and use the LBRMT to model flexible rotors in a confined fluid box to test the contact, bending, twisting, and stretching of soft bodies under extreme deformation. In particular, we set up the simulation such that there are four neutrally buoyant rotors with a prong length of $0.2L$ and period T anchored in a confined fluid box of size $L \times L$. The four rotors are placed so that they come into contact while rotating. We apply anchoring forces $\boldsymbol{f}_a$ at each rotor center area with a radius of $0.025L$ to induce twisting. $\boldsymbol{f}_a$ acts like a spring, periodically twisting the anchored area. This actuation allows the flexible prongs to follow, enabling the rotor to spin. During the first $T/2$ of the simulation, the top two rotors rotate counter-clockwise, while the bottom two rotate clockwise, for a full revolution of 2π. In the subsequent $T/2$ of the simulation, anchoring forces reverse, causing the two pairs of rotors to rotate in the opposite direction, for another full revolution of 2π.

We visualize the density field and streamlines (Fig. 3.9A) and solid von Mises stress field (Fig. 3.9B) at selected times for one revolution T. When the rotors come into contact and slide past each other, we observe significant stretching, bending, and twisting (Fig. 3.9B) captured by the LBRMT. In addition, when the rotor is compressed due to contact (Fig. 3.9A), the local density increases; whereas when the rotor is stretched due to stretching, the local density decreases. We also observe that rotation can excite an initially quiescent fluid and create flow, which can be used to transport or mix up objects. Insights of this rotor simulation can be applied to model more complex FSI systems such as an array of cilia or seaweed.

3.4.3 Settling and floating

When we submerge a solid under fluid and then release it, two scenarios can arise. If the solid density is higher than the fluid ($\rho_s > \rho_f$), the solid will settle until it hits the fluid bottom due to gravity. If the solid density is lower than the fluid ($\rho_s < \rho_f$), the solid will float. Examples of settling and floating are common to see daily, and we start with a simplified scenario of only one deformable solid. We place a solid of radius (or half-edge length) $0.2L$ in the middle of a long confined fluid box with an aspect ratio $L{:}H = 1{:}3$, then let it either settle or float based on its density. We set up a force balance between gravity and buoyancy $\rho_s \boldsymbol{a} = \rho_s \boldsymbol{g} - \rho_f \boldsymbol{g}$. The resulting acceleration $\boldsymbol{a} = \left(1 - \frac{\rho_f}{\rho_s}\right) \boldsymbol{g}$ is the force density that drives the solid motion.

Since the solid has some softness, as it is moving through fluids, the fluid motion can deform the solid. Therefore, falling speeds and terminal shapes of such solids can be affected by the solid density ρ_s and softness G (shear modulus). We vary the solid density $\rho_s \in \{0.75, 0.83, 0.9, 1.125, 1.25, 1.5\}$ and solid shear modulus $G \in \{0.075, 0.1, 0.2, 0.3, 0.5, 1.0, 5.0\}$ to study the effects of these two parameters on settling and floating. The smooth flux correction (Section 3.2.3) allows us to simulate non-neutrally-buoyant solids with no need to modify accelerations for lighter solids, valid for all solid density ranges—equal to ($\rho_s = \rho_f$), bigger than ($\rho_s > \rho_f$), and smaller than ($\rho_s < \rho_f$) the fluid density. Fig. 3.10 summarizes the results of soft solids settling or floating, characterized by varying shapes, densities ρ_s, and softness G. As these solids move through the fluid, their deformation can be observed in the curved reference map contours, and the intensity of solid stress is illustrated in Fig. 3.10B.

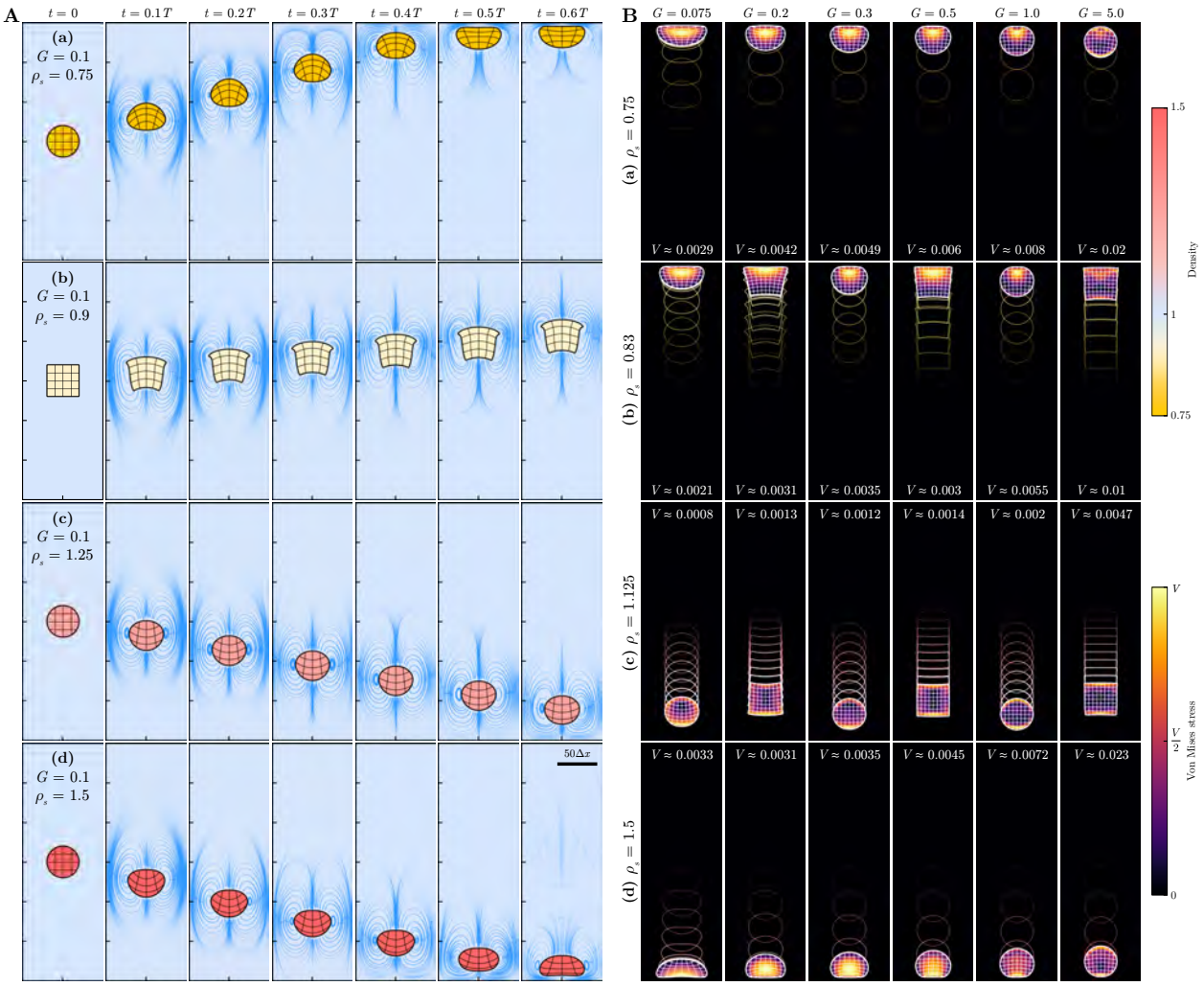

Figure 3.10: Settling and floating of a solid. A solid of radius (or half-edge length $0.2L$) is released at the center of a confined fluid box. It moves through the fluid at different speeds to its top or bottom based on its density and softness. Simulation parameters are $(L, H, \tau, \rho_f, T) = (100, 300, 1.0, 1.0, 20000)$. **(A)** Snapshots of density field and streamlines. Blue streamlines show fluid flow resulting from solid settling or floating, with streamline density signaling flow speed. Thick black lines are the solid–fluid interfaces and thin black lines are the reference map contours which illustrate the solid deformation. Colors represent the density field. **(B)** Snapshots of the von Mises solid stress field, with colormap normalized by the maximum stress value V of each case. Highlighted areas indicate increased shear, reflecting solid deformation intensity when the solid moves through the fluid or is stopped at the fluid box wall.

Fig. 3.10A shows the effects of density in settling when softness is kept constant ($G = 0.1$) with four cases of $\rho_s \in \{0.75, 0.9, 1.25, 1.5\}$. When $\rho_s < \rho_f$ (Fig. 3.10A(a,b)), the solid floats to the top. The

lighter the solid density is, the faster it moves (Fig. 3.10A(a)). When $\rho_s > \rho_f$ (Fig. 3.10A(c,d)), the solid settles to the bottom. The heavier the solid density is, the faster it moves (Fig. 3.10A(d)). Fig. 3.10A(c) also highlights that the LBRMT can simulate shapes beyond circles, but also squares with corners. The reference map setup ensures that the sharpness of corners gets preserved via the level set. The LBRMT can also model contact between solids and wall boundaries, indicated in Fig. 3.10A(a,d) where the solid gets deformed after reaching the wall boundary.

Fig. 3.10B shows the effects of softness in the terminal shape at the end of simulation ($t = T$). We consider four cases of solid densities $\rho_s \in \{0.75, 0.83, 1.125, 1.5\}$ with six cases of solid shear moduli $G \in \{0.075, 0.2, 0.3, 0.5, 1.0, 5.0\}$. We plot the solid–fluid interface with at same time interval $\Delta T = T/10$ to illustrate the solid trajectory via gradient-coded outlines. More overlapping outlines indicate that the solid moves faster and remains at the wall boundary longer. Fig. 3.10B(a,d) show that for solids at the wall boundary, softer solids deform more due to their contact with the wall boundary. In contrast, stiffer solids tend to maintain their shape. Fig. 3.10B(a,b) show that lighter solids move faster, consistent with the results in Fig. 3.10A. With settling and floating, we demonstrate that the LBRMT can simulate solids across a range of densities and softness, making it suitable for modeling a complex suspension of different solids. In the next subsection, we simulate two cases of complex suspensions with hundreds of solids and investigate the role of softness in efficient mixing.

3.4.4 Mixing

An intriguing extension to the previous example of one solid settling and floating (Section 3.4.3) is the settling and floating of many solids, *i.e.* mixing. When there are density differences among the suspended solids in fluid, gravity or buoyancy drives solids to their steady-state phases, at which mixing occurs. Mixing of complex suspensions is a common process across scales in engineering, nature, and everyday life. From separating particulate flow [248] or manufacturing fibers [249] to forming riverbeds [250] even pouring boba pearls into milk tea, mixing is a frequent phenomenon with pivotal applications. However, our understanding of how the softness of solids affects mixing efficiency remains limited. In the first example, we study the mixing of 100 solids with densities $\rho_1 = 1.25$ and $\rho_2 = 0.83$ in a confined fluid box of density $\rho_f = 1$ and size $L \times L$. We vary the softness (shear modulus) $G \in \{0.2, 0.3, 0.4, 0.5, 1.0, 5.0\}$. With this simulation setup, we aim to answer the question: *Does softness enhance mixing rate?*

Our simulations (Fig. 3.11) indicate that softness indeed enhances mixing rate, *i.e.* softer suspensions mix faster. Mixing rate, loosely defined as the speed at which solids reach their steady-state phases due to gravity or buoyancy, is visualized in Fig. 3.11A through snapshots at selected time intervals. We color heavier solids ($\rho_1 = 1.25$) in red and lighter solids ($\rho_2 = 0.83$) in yellow. Fig. 3.11A(a–d) show results of increasing stiffness ($G \in \{0.2, 0.5, 1.0, 5.0\}$). Fig. 3.11A(a) demonstrates that softer suspensions initiate mixing earlier, proceed at a faster rate, and reach equilibrium more quickly. The material property of softness promotes efficient mixing, allowing softer solids to deform and navigate through narrower openings instead of getting jammed or clogged [251]. As the suspensions become stiffer (Fig. 3.11A(c,d)), mixing

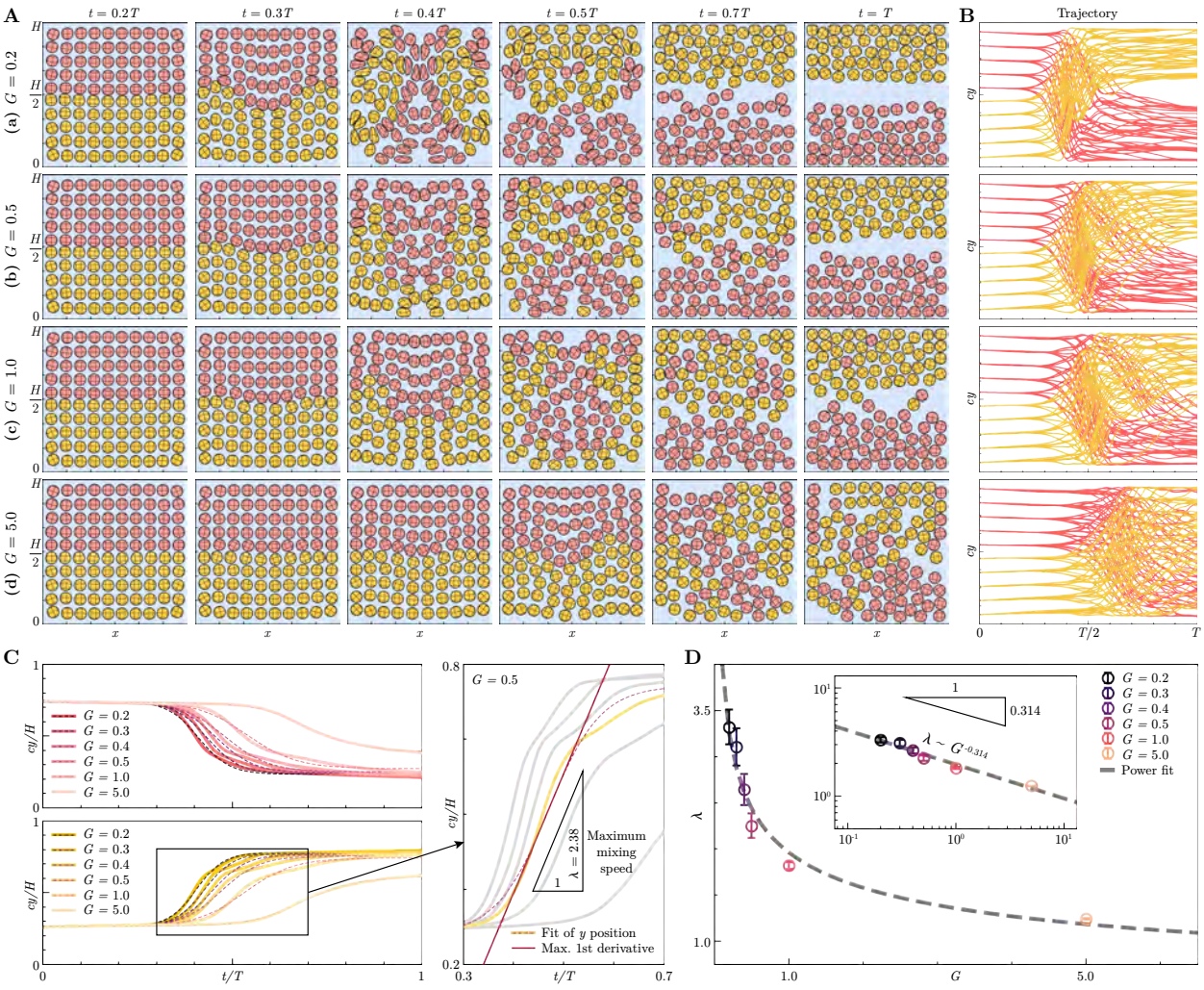

Figure 3.11: Softness enhances mixing rate. 100 densely-packed solids (50 solids of $\rho_1 = 1.25$ and 50 solids of $\rho_2 = 0.83$) settle and float in a confined fluid box of size $L \times L$. We use red to color-code heavier solids and yellow for lighter solids. Simulation parameters are $(L, \tau, \rho_f, T) = (300, 1.0, 1.0, 50000)$. **(A)** Snapshots of 100 solids mixing at selected time intervals. (a) shows that softer solids exhibit more efficient mixing, *i.e.* two species reach equilibrium faster, separating at the top and bottom. (d) shows that stiffer solids take a longer time to mix. **(B)** Trajectories of the y component of solid centroids c_y indicate mixing efficiency and evolution. In the pre-mixing state, the red and yellow trajectories remain separate yet slowly moving. Subsequently in the mixing stage, trajectories overlap, and the yellow ones move to the top while the red ones go to the bottom. Softer solids have the trajectories overlapping earlier and separated into two phases sooner. In contrast, stiffer solids experience a delay in mixing and take a longer time to reach equilibrium. **(C)** Normalized averaged centroid trajectories $\widehat{y} = c_y/H$ for each solid species at different softness G. A hyperbolic tangent is fitted to the simulation data, whose first derivative is then computed to represent the speed of mixing. We extract the maximum mixing speed λ to represent the mixing efficiency of each softness. **(D)** Maximum mixing speed λ as a function of softness G, fitted with a power law $\lambda \sim G^{-0.314}$.

occurs at later times. Notably, Fig. 3.11A(d) shows instances of clogging in certain areas, where stiffer solids struggle to create openings to reconfigure their states. Qualitatively, softer suspensions exhibit higher efficiency in mixing.

To quantitatively study our question, we extract the y position of solid centroid, c_y, for each solid of each species (red and yellow). We plot in Fig. 3.11B the c_y trajectories during simulation time $t \in [0, T]$. We observe three stages of mixing: pre-mixing, mixing, and post-mixing. In the pre-mixing stage, red and yellow trajectories remain separate yet slowly move. During the mixing stage, c_y rapidly changes and trajectories overlap, with yellow ones moving to the top while the reds to the bottom. Fig. 3.11B(a) shows that mixing occurs earlier and takes a shorter time for softer solids, leading to the separation of red and yellow trajectories in the post-mixing stage. However, as solids become stiffer, mixing occurs later and longer. For very stiff solids (Fig. 3.11B(d)), they are still in the mixing stage at the end of the simulation.

To abstract the relation between softness and mixing rate further, we average c_y trajectories of 50 solids of each species for $G \in \{0.2, 0.3, 0.4, 0.5, 1.0, 5.0\}$. The normalized average trajectories $\widehat{y} = c_y/H$ are plotted against normalized time $\widehat{t} = t/T$ in Fig. 3.11C with solid lines (colors representing softness). These trajectories are fitted with $\widehat{y} = a \tanh\left(b\left((\widehat{t} - c)\right) - d\right)$ for a position function $\widehat{y} = f(\widehat{t})$ using dashed lines. For all G values (except $G = 5.0$), we observe three mixing stages: pre-mixing (remaining at initial position), mixing (moving with increasing then decreasing speed), and post-mixing (remaining at equilibrium). We then extract the mixing speed by calculating the derivatives of position functions $\widehat{y}$ for each species of different softness. Fig. 3.11C shows an example where we compute the maximum

mixing speed λ of $G = 0.5$ by taking the maximum value of its first derivative. We use λ to determine mixing efficiency with respect to G. In Fig. 3.11D, we plot λ as a function of G and observe a negative decreasing trend between the maximum mixing speed and softness. We fit the λ values with a power law $\lambda = G^{\alpha}$ and report $\alpha \approx -0.314$. This trend implies mixing is more efficient when solids are softer and less when solids are stiffer. In this example, we have explored the relation between mixing rate and softness in a limited case. Our findings demonstrate that softness enhances mixing rate, which could be applied to improve efficient mixing in confined geometries. Other factors, such as fluid viscosity, solid shape, and wall-confinement geometry, could also influence this relation.

We conclude this subsection with the second mixing example, wherein we test the LBRMT with 506 solids. These ellipses are initialized both vertically and horizontally to guarantee contact during mixing and allow a clearer indication of motion and deformation with the orientation of the ellipses. We use the same setup as the first mixing example of 100 solids to ensure collision, mixing, and contact, where we put the 242 heavier solids ($\rho_1 = 1.25$) in the upper half and the 264 lighter solids ($\rho_2 = 0.83$) in the lower half of a confined fluid box of size $L \times H$. Fig. 3.12 shows the density field, streamlines, and solid von Mises stress field of selected snapshots during simulation time T. Given the simulation is set up so that there is space along the wall boundary, in the pre-mixing stage, we see mixing onset starting near the walls, where the two species slide against each other to create space for openings. Since the solids are soft, their contact leads to more deformation, thus more space to promote mixing. Towards the end of the pre-mixing stage, we see formation of two vortices near the walls ($t = T/16$), which start to develop and drive the motion

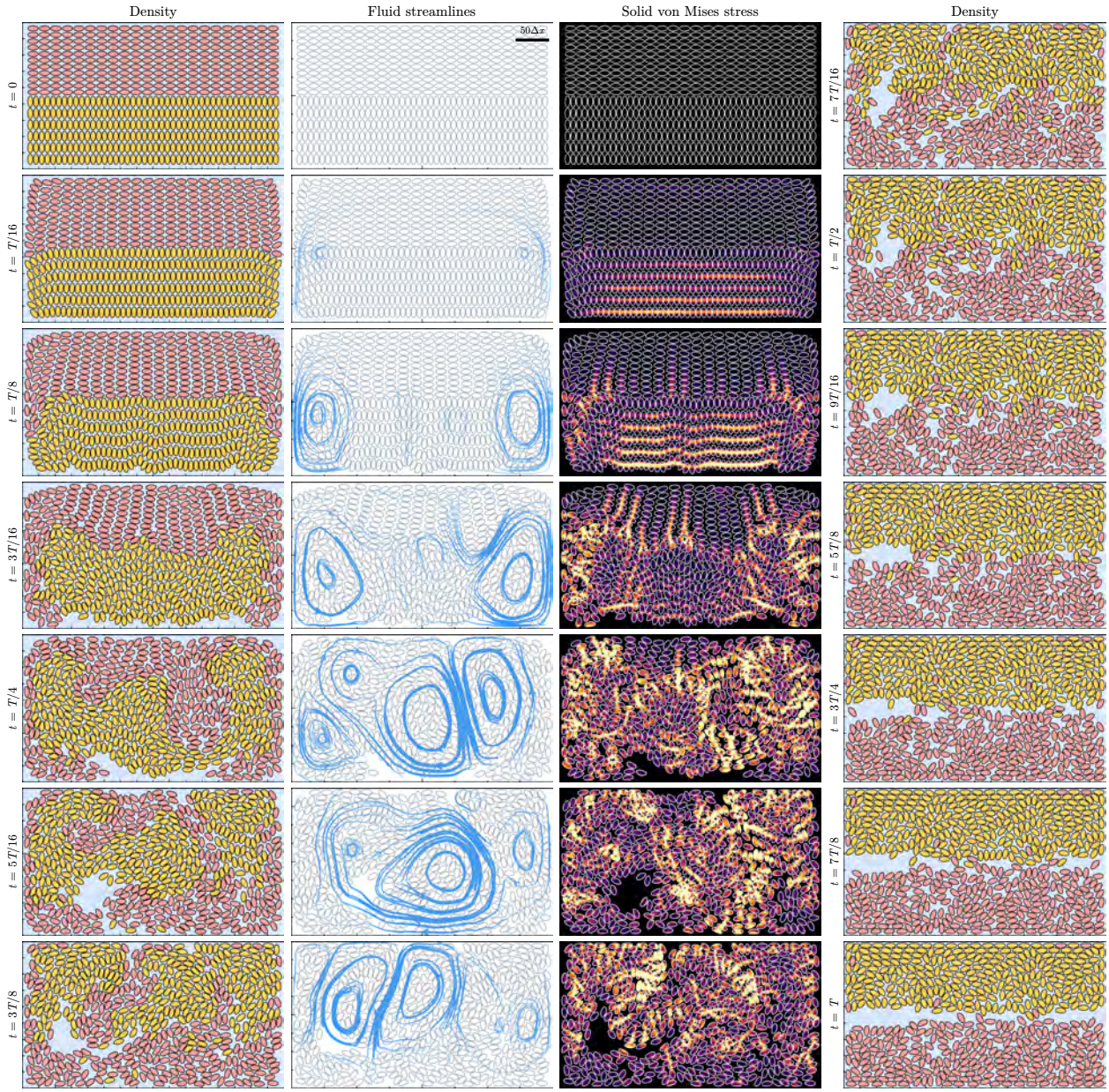

Figure 3.12: Mixing of 506 ellipses. Snapshots of 242 heavier solids ($\rho_1 = 1.25$) and 264 lighter solids ($\rho_2 = 0.83$) settling and floating in a confined fluid box of size $L \times H$ at selected times. Simulation parameters are $(L, H, \tau, \rho_f, G, T) = (800, 450, 1.0, 1.0, 1.0, 200000)$. We visualize the density field, streamlines, and solid von Mises stress (normalized over the maximum value over the entire time T).

of the ellipses. Additionally, there is an onset of instability along the interface between the two species. The instability starts to grow ($t = T/8$ to $T/4$), leading to the mixing stage and bringing lighter solids to the top and heavier solids to the bottom. The streamline plots further indicate that mixing is driven by large vortices, which facilitate motion on a larger scale. After the solids finish exchanging their positions, they enter the post-mixing stage and settle or float to their equilibrium and phase separate.

The von Mises stress field provides an alternative probe to analyze mixing by examining the evolution and intensity of solid deformation. In the pre-mixing stage ($t = T/16$), the bottom yellow solids experience compression against each other, shown by the highlighted horizontal long strips. As time progresses ($t = T/8$), these strips undergo a transition and become wavy—an onset of instability—resulting in solids deforming and morphing out of their original global configuration. During the mixing stage ($t = T/4$ to $3T/4$), solids rapidly move and come into contact, propelled by large vortices. Regions with faster flow show more pronounced von Mises stress, indicating larger solid deformation in these areas.

With this example, we have demonstrated that the LBRMT can effectively and reliably model large-scale collective motion via a full FSI simulation of individual agents. This capability opens up potential applications in active matter and liquid crystals research, enabling simulations of complex phenomena such as swarming bacteria, schooling fish, flowing cells, active filaments, or actuated autonomous agents.

3.5 Discussion

We have presented a fully-integrated lattice Boltzmann method for FSI simulations which is accurate, versatile, and straightforward to implement and parallelize. The lattice Boltzmann reference map tech-

nique (LBRMT) provides a simulation framework on one fixed computational grid that couples large-deformation solids with fluids. We believe our method provides promising future applications, particularly for the lattice Boltzmann community, to simulate finite-strain solids with an Eulerian boundary condition for the solid–fluid interface (the smooth flux correction), as demonstrated in the previous sections. We have shown that the LBRMT has significant improvements in the fluid update compared with the IncRMT [98] for FSI simulations. We have also demonstrated its capability in modeling extreme bending, twisting, and mixing of soft structures in fluid. By efficiently capturing the solid deformability, the LBRMT is a valuable tool for studying the spatiotemporal dynamics of collective motion in biological systems, while being able to probe the dynamics of individual agents through a two-way coupling of fluid–structure interaction simulation.

However, our method also comes with limitations. One constraint is on the fluid: The Mach number of the system needs to be small ($Ma < 0.3$) [76] to maintain the fluid quasi-incompressibility assumption; the LBRMT is not designed to model fast-moving or turbulent flow but is limited to slow-moving and laminar flow. Another constraint is on the LB relaxation time: The current implementation assumes $\tau = 1$, meaning the LB populations completely relax toward their equilibrium. Although this specific choice is closer to unity for numerical stability and is common in the LB simulations [76], it limits the range of fluid viscosity to be simulated as well as the grid spacing and timestep. A more general forcing scheme needs to be developed to incorporate both fluid and solid force densities (especially the divergence of the solid stress) for FSI simulations. Despite these limitations, the LBRMT remains a ro-

bust and promising tool to simulate systems with finite inertia that involve different densities and flow at small and intermediate Reynolds numbers.

For future directions, a natural extension is to incorporate rigid solids to model moving rigid solids [252] and non-standard wall boundaries. Possible implementations include integrating moving rigid solids onto the same computational grid [214] and developing a new contact model for collisions between soft and rigid solids. This direction opens avenues for experiment–simulation integration such as in microfluidics [253]. In addition to creating a "digital twin", the LBRMT can further our understanding of collective behavior in tapered channels [254] by probing mechanical fields that are difficult to measure in experiments and help in designing new strategies to prevent clogging [255], such as digitally finding optimal positions to place rigid solids in microfluidic channels [256]. Another extension is to use pseudopotential [257] to incorporate multiphase flow for fluid–structure–gas simulations. A third extension is to model solid self-contact, which requires a generic reference map data structure to track gradients of the level set function for the solid orientation. This extension can be relevant in bioengineering applications, such as simulating the dynamics of elongated slender objects like a collection of flexible rod-like worm blobs in fluid [258]. Our long-term goal is to expand the LBRMT beyond hyperelasticity to plasticity, and even fracture. We envision our method as a physically accurate tool for investigating spatiotemporal patterns of collective behavior while keeping track of individual dynamics in active matter.