Interaction of fluid interfaces with immersed solid particles using the lattice Boltzmann method for liquid–gas–particle systems

Kevin W. Connington a,*, Taehun Lee b, Jeffrey F. Morris a, c

a The Levich Institute, The City College of New York, New York, NY, 10031, United States
b The Department of Mechanical Engineering, The City College of New York, New York, NY, 10031, United States
c The Department of Chemical Engineering, The City College of New York, New York, NY, 10031, United States

Article history:
Received 2 May 2014
Received in revised form 27 September 2014
Accepted 30 November 2014
Available online 8 December 2014

Keywords:
Lattice Boltzmann
Multiphase
Particles
Interface

ABSTRACT

Due to their finite size and wetting properties, particles deform an interface locally, which can lead to capillary interactions that dramatically alter the behavior of the system, relative to the particle-free case. Many existing multi-component solvers suffer from spurious currents and the inability to employ components with sufficiently large density differences due to stability issues. We developed a liquid–gas–particle (LGP) lattice Boltzmann method (LBM) algorithm from existing multi-component and particle dynamics algorithms that is capable of suppressing spurious currents when geometry is fixed while simulating components with liquid–gas properties. This paper presents the LGP algorithm, with several code validations. It discusses numerical issues raised by the results and the conditions under which the algorithm is most useful. The previously existing particle dynamics algorithm was augmented to capture surface tension forces arising from the interface, which was validated for the case of a 2D capillary tube. Using the full algorithm, a particle situated in a region of bulk fluid in an otherwise quiescent situation remained in its original location, indicating that spurious currents were suppressed. A particle brought into the interface of a drop (without gravity) achieved its expected depth of immersion into the drop, demonstrating that all aspects of the code work together to produce the correct equilibrium state when a particle is in the interface. As in an experiment, two particles on a flat interface approached each other due to capillary effects. The simulation approach velocity was faster than that of the experiment, but agreed qualitatively, achieving the same equilibrium state. Given the validations and the favorable, though imperfect, experimental comparison, this algorithm can be a useful tool for simulating LGP systems. The motion of particles normal to the interface can be considered reliable, and the motion tangent to the interface can be considered qualitatively accurate, leading to the correct equilibrium state.

© 2014 Elsevier Inc. All rights reserved.

* Corresponding author.
E-mail address: kevin.connington@gmail.com (K.W. Connington).

http://dx.doi.org/10.1016/j.jcp.2014.11.044
0021-9991/© 2014 Elsevier Inc. All rights reserved.
1. Introduction

The presence of interest in solid particles can drastically alter the behavior of a multi-phase/multi-component flow. These types of flows are of interest in many areas such as electronics fabrication by ink-jet printing [1] and emulsion/foam stabilization in the food, cosmetics, and oil industries [2–4]. Particles in bulk fluid regions typically alter the effective viscosity of the carrier fluid [5], but some of the most interesting behavior is observed when particles are immersed in a fluid–fluid interface. Any particle that is immersed in an interface is only removed with difficulty. Specifically, the energy required to remove a particle from the interface is typically large relative to the thermal energy $k_B T$ ($k_B$ is Boltzmann’s constant and $T$ is the temperature); i.e., adsorption effectively cannot be reversed by Brownian motion [6–8]. Due to their weight and wetting properties, particles deform the interface they are immersed in, typically leading to capillary force interactions with other particles or solid objects [9,11,12]. These capillary interactions are responsible for particle self-assembly and other aggregation phenomena [13–22].

As any physical situation that involves multiple fluid components and solid particles may be difficult to investigate experimentally, insight can be gained by supplementing experiments with an appropriate computational model. Any numerical algorithm that attempts to capture the relevant physics of a liquid–gas–particle (LGP) system will encounter difficulties, primarily the moving boundary problem for finite-sized particles [23,24] and the interface tracking/capturing scheme for the fluid components [25].

It is important to keep in mind that there is a two-way coupling between the particles and fluids. Particles are free to move based on the forces acting on them, where forces come from the bulk fluids and the interface, resulting in a moving boundary problem. The forces acting on the particle in this model are macroscopic in nature, i.e. there are no van der Waals forces, electrostatic forces, or Brownian motion models incorporated; only pressure, viscous, and interfacial tension forces. As the solid boundaries move, either the domain must be remeshed, as in finite element simulations [26], or boundary/fluid nodes must be updated, as in fixed grid methods such as the lattice Boltzmann method (LBM) [24,27–29] or the immersed boundary method (IBM) [30]. At the same time, the moving boundaries disturb the fluid, where the full two-way coupling arises from the finite-sized nature of the particles. Many algorithms simulate the presence of particles by incorporating point forces, associated with particle drag and other forces, that disturb the fluid locally. The particles are in turn convected by the fluid, but the actual particle geometry is not directly resolved [31]. This approach is useful for gleaning macroscopic properties when particles are much smaller than any relevant length scale of the fluid motion, but it fails to capture the detailed interaction between particles and fluid when the particle size is not negligible [32], as is critically important for our topic of particles transported by capillary interactions. Here we explicitly resolve the finite size of the particles which are transported by the fluid, and the fluid in turn experiences stresses in response to the particle motion, resulting in a two-way coupling.

The LBM has become a popular choice for simulating both particle dynamics and multi-phase/component flow [33–35], individually. Stratford et al. [36–38] were the first to use a combination of both methods to simulate a fluid–fluid–particle (FFP) system using the LBM. We use LGP instead of FFP to distinguish between systems with large differences in fluid properties. They showed that particles heavier than either fluid will bow an otherwise flat interface to support it by surface tension forces, and will detach close to the critical value of Bond number ($B_0 = \alpha^2 \Delta \rho g/\sigma = 0.75$, where $\alpha$ is the particle radius, $\Delta \rho$ is the difference in density between the particle and heavier fluid, $g$ is gravitational acceleration, and $\sigma$ is the fluid–fluid surface tension). They also described various issues related to the parallel computation of such algorithms. Cates et al. [8,39,40] demonstrated that particles immersed in the interface of a fluid cylinder can arrest breakup into droplets by Rayleigh–Plateau instability. They also proposed a new class of soft materials know as bicontinuous interfacially jammed emulsion gels (bijels) which were confirmed by experiment. Shinto et al. [41,42] used a 2D multiphase LBM [43] to compare the lateral capillary force between objects to the analytical solution. They computed the lateral force between floating squares and cylinders, allowing the floating objects to move vertically, but not laterally, for different separations. They confirmed that the LBM-calculated capillary force decreased exponentially with the separation of the objects, but no quantitative comparison was made. The maximum density ratio ($\rho_1/\rho_2$) simulated was about 5. Onishi et al. [44] were the first to employ the so-called Shan–Chen (SC) multiphase formulation, but particles were represented by fluid forced to [execute] rigid body motion. They too computed the lateral force between floating particles at a fixed separation. Kim et al. [45,46] studied bijels formed by sweeping up magnetic particles during the spinoval decomposition of bicontinuous fluids, as well as the field induced breakup of emulsion drops stabilized by colloidal particles. Joshi and Sun [47–49] studied spontaneous wetting of particle laden drops in two and three dimensions. They clearly detailed the implicit Newtonian dynamics scheme for particle velocity that is used in this paper [48]. Jansen and Harting [50] studied the transition from bijels to Pickering emulsions. Günther et al. [51] studied ellipsoidal particles at interfaces. Liang et al. [52] studied the self-assembly of particles on a flat interface due to capillary interactions. This paper introduces a unique combination of existing multi-component and particle dynamics algorithms, with several key differences between our method and existing FFP/LGP schemes.

In all other existing schemes, the multi-component algorithm is handled differently, either with the SC formulation or a different free energy model. The SC method has its benefits, but our free energy-based formulation can eliminate spurious currents at equilibrium for fixed, i.e. non-moving, solid boundary geometry [53]. The importance of eliminating spurious currents, an artificial velocity field driven by an unbalanced discretization of the surface tension force, has been reviewed previously [54]. For freely moving particles, the spurious currents are not eliminated, but suppressed to the point where they do not interfere with the dynamics of the system. Spurious currents can be eliminated when the boundaries
(particles) are anchored (Section 3.1). Another difference is our ability to simulate fluid components with large contrasts in fluid properties, e.g., water–air systems. This is important as many flows of engineering interest use such fluids. None of the previous references reported using property contrasts as large as we report here. The last main difference is that an adhesion force between solid and fluid is calculated for all fluid nodes connected to a particle node in the SC formulation. The contact angle is recovered by specifying the potential used in this calculation. Our contact angle is recovered by specifying the wall free energy, so we do not have the ability to compute an adhesion force directly. As a result of our boundary conditions, we had to include a supplemental force to recover the appropriate physics when any particle is immersed in the fluid–fluid interface. The increased effort to compute the interfacial force is not too onerous as it can be performed during the Momentum Exchange (ME) sweep (discussed in Section 2.2.1), and it allows us to retain our free energy formulation.

In this paper, we develop an LBM to combine the aspects of existing multi-component and particle dynamics solvers. In particular, we use the multi-component algorithm developed by Lee et al. [55]. We demonstrate that the combination of these existing methods is sufficient when particles are contained inside bulk fluid regions of the flow. Due to the implementation of our boundary conditions, we must add a supplemental force to recover the appropriate physics when a particle is immersed in the fluid–fluid interface. We verify the inclusion of this supplemental interfacial force, and the algorithm as a whole, with a series of three progressively rigorous validations. We first show that, for our method, the additional surface tension force must be included to achieve the correct total force on a fixed object. We use a fixed object in order to remove the complication of moving boundaries. Then, using the validated force algorithm, we show that an unrestrained particle achieves its predicted position when immersed in an interface; i.e., the depth of immersion into the interface. This demonstrates that the algorithm achieves the right equilibrium configuration, but says nothing about the approach to equilibrium. Finally, we demonstrate that the algorithm recovers the dynamics of a transient flow qualitatively by comparing our results to that of an experiment [56] where two particles approach each other on a flat interface due to capillary forces. The results are not quantitatively exact, but they provide insight into the nature of such simulations, and allow us to offer some conclusions on the subject. Given the favorable result of our tests, we conclude that our algorithm is appropriate to implement in most LGP systems, as we can expect the dynamics to be qualitatively correct and to yield the proper equilibrium configuration.

The rest of the paper is organized as follows. We detail the numerical method in Section 2, including the supplemental surface tension force in Section 2.2.2. We show that existing methods are sufficient to simulate the case when a particle occupies only the bulk fluid regions, and that spurious currents are suppressed by our method in Section 3.1. We describe three progressively rigorous validations in Sections 3.2–3.4, respectively. We discuss the strengths and weaknesses of our algorithm and offer some conclusions in Section 4.

2. Numerical method

The numerical method consists of a multi-component lattice Boltzmann algorithm for the fluid coupled with particle dynamics. A free energy-based algorithm was used for the fluid in conjunction with solid boundary interactions. An established algorithm for particle dynamics was adapted to this particular multi-component system. Due to our imposed boundary conditions, a supplemental interfacial force was imposed to complete the particle dynamics.

2.1. Multi-component algorithm

We implement the multi-component algorithm developed by Lee et al. [55], which is from the class of free energy-based multi-component lattice Boltzmann algorithms. The lattice Boltzmann equations (LBE) that are used recover the Cahn–Hilliard, pressure evolution, and momentum equations. Since most of the method has been described previously [53,55], new aspects of the algorithm are described in detail below, while established aspects are summarized to provide context.

2.1.1. Cahn–Hilliard equation

A system of binary fluids evolves so that the total free energy ($Ψ$) is minimized. There is a contribution from the bulk ($Ψ_{b}$) which contains both local and nonlocal terms [57]. It is assumed that interactions between the solid surface and the liquid–gas interface ($Ψ_{s}$) are short-range, and are accounted for by a surface integral [58]. The total free energy is written:

$$Ψ = Ψ_{b} + Ψ_{s} = \int_{V} \left( E_{0}(C) + \frac{K}{2} |∇C|^2 \right) dV + \int_{S} \left( φ_{0} - φ_{1}C_{S} + φ_{2}C_{S}^{2} - φ_{3}C_{S}^{3} + \cdots \right) dS. \quad (1)$$

Here the local contribution to the bulk free energy is $E_{0} = βC^{2}(C - 1)^{2}$, where $β$ is a constant and $C$ is the composition or order parameter. The relative strength of the nonlocal contribution is specified by the gradient parameter $κ$. The interface thickness ($ξ = √(8κ/β)$) and fluid–fluid surface tension ($σ = √(2κ/β)$) are determined by choosing a suitable combination of local and nonlocal parameters. The composition at the surface is $C_{S}$, and the $φ_{i}$’s are constant coefficients, where any number of terms may be retained. The system progresses toward its minimal energy state according to the Cahn–Hilliard equation:

$$\frac{∂C}{∂t} + \mathbf{u} · ∇C = M∇^{2}μ, \quad (2)$$

where $\mathbf{u}$ is the bulk velocity, $μ$ is the chemical potential, and $M$ is the mobility.
According to Eq. (2), the system is out of equilibrium when there are nonzero gradients of the chemical potential. Free energy minimization is realized by extracting the chemical potential from the terms of the total free energy, upon variation. The chemical potential is written \( \mu = \mu_0 - \kappa \nabla^2 C \), where \( \mu_0 = \partial E_0 / \partial C \) is the classical (local) part. Terms from the surface integral are included in the definition of \( \mu \) when computing \( \nabla^2 C \) near a boundary, as will be discussed later.

The solution to Eq. (2) requires two boundary conditions. The boundary condition for \( \nabla^2 \mu \) prevents a mass flux normal to a solid boundary:

\[
\mathbf{n} \cdot \nabla \mu|_s = 0, \tag{3}
\]

where \( \mathbf{n} \) is the normal unit vector to the surface. The boundary condition for \( \nabla^2 C \) is established by minimizing the surface free energy. We use the cubic boundary condition [59,60]:

\[
\mathbf{n} \cdot \nabla C|_s = \frac{\phi_c}{k} (C_s - C_c^2), \tag{4}
\]

where \( \phi_c \) is a constant chosen to recover the desired contact angle at equilibrium.

### 2.1.2. Lattice Boltzmann equations

The discrete Boltzmann equation (DBE) for the transport of binary fluids is written [61–65]

\[
\frac{\partial f_{\alpha}}{\partial t} + \mathbf{e}_\alpha \cdot \nabla f_{\alpha} = -\frac{1}{\lambda} (f_{\alpha} - f_{eq}^{\alpha}) + \frac{1}{c_s^2} (\mathbf{e}_\alpha - \mathbf{u}) \cdot \mathbf{F} \Gamma_{\alpha}, \tag{5}
\]

where \( f_{\alpha} \) is the distribution function corresponding to the discretized velocity \( \mathbf{e}_\alpha \). \( \mathbf{u} \) is the volume-averaged velocity, \( c_s \) is the speed of sound, \( \lambda \) is the relaxation time, \( \mathbf{F} \) is the intermolecular force, \( \Gamma_{\alpha}(\mathbf{u}) = f_{\alpha}^{eq} / \rho \), and \( f_{eq}^{\alpha} \) is the equilibrium distribution function defined by:

\[
f_{eq}^{\alpha} = t_\alpha \rho \left[ 1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_\alpha \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right]. \tag{6}
\]

where \( t_\alpha \) is the weight corresponding to \( \mathbf{e}_\alpha \), and \( \rho \) is the mixture density.

Surface tension effects arise from the inclusion of the intermolecular force which is written [66]

\[
\mathbf{F} = \nabla \rho c_s^2 - (\nabla p + \mu \nabla C), \tag{7}
\]

where \( p \) is the dynamic pressure that enforces incompressibility.

The DBE for mass and momentum is transformed into the DBE for pressure evolution and momentum [67,68] by defining a new distribution function \( \tilde{g}_\alpha = f_{\alpha} c_s^2 + (p - \rho c_s^2) \Gamma_{\alpha}(0) \) with corresponding equilibrium distribution function \( \tilde{g}_{eq}^{\alpha} = f_{eq}^{\alpha} c_s^2 + (p - \rho c_s^2) \Gamma_{\alpha}(0) \). The DBE becomes

\[
\frac{\partial \tilde{g}_\alpha}{\partial t} + \mathbf{e}_\alpha \cdot \nabla \tilde{g}_\alpha = -\frac{1}{\lambda} (\tilde{g}_\alpha - \tilde{g}_{eq}^{\alpha}) + (\mathbf{e}_\alpha - \mathbf{u}) \cdot \left[ \nabla \rho c_s^2 \left( \Gamma_{\alpha} - \Gamma_{\alpha}(0) \right) + \mu \nabla C \Gamma_{\alpha} \right]. \tag{8}
\]

Eq. (8) recovers the pressure evolution and momentum equations:

\[
\frac{\partial p}{\partial t} + \rho c_s^2 \nabla \cdot \mathbf{u} = 0, \quad \text{and} \tag{9}
\]

\[
\rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla C + \left[ \rho c_s^2 \lambda \cdot \left( \nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \right]. \tag{10}
\]

Here the dynamic viscosity is identified as \( \eta = \rho c_s^2 \lambda \).

Another distribution function is introduced to track the order parameter: \( h_\alpha = (C / \rho) f_{\alpha} \) and \( h_{eq}^{\alpha} = (C / \rho) f_{eq}^{\alpha} \). The corresponding DBE is written:

\[
\frac{\partial h_\alpha}{\partial t} + \mathbf{e}_\alpha \cdot \nabla h_\alpha = -\frac{1}{\lambda} (h_\alpha - h_{eq}^{\alpha}) + M \nabla^2 \mu \Gamma_{\alpha} + (\mathbf{e}_\alpha - \mathbf{u}) \cdot \left[ \nabla C - \frac{C}{\rho c_s^2} \nabla p - \mu \nabla C \right] \Gamma_{\alpha}. \tag{11}
\]

Eq. (11) recovers

\[
\frac{\partial C}{\partial t} + \nabla \cdot (\nabla \mathbf{C}) = M \nabla^2 \mu, \tag{12}
\]

which is the advective Cahn–Hilliard equation when \( \nabla \cdot \mathbf{u} = 0 \).

The LBEs are obtained by integrating the DBEs, Eqs. (8) and (11), along characteristics using a trapezoidal approximation over a time step \( \delta t \). Modified distribution functions \( \tilde{g}_\alpha \) and \( h_\alpha \) are introduced as in [55], and the resulting LBEs become:

\[
\tilde{g}_\alpha(x, t) = \tilde{g}_\alpha(x - \mathbf{e}_\alpha \delta t, t - \delta t) - \frac{1}{\tau + 1/2} (\tilde{g}_\alpha - \tilde{g}_{eq}^{\alpha}) |(x - \mathbf{e}_\alpha \delta t, t - \delta t) + \delta t (\mathbf{e}_\alpha - \mathbf{u}) \cdot [\nabla^\text{MD} \rho c_s^2 \left( \Gamma_{\alpha} - \Gamma_{\alpha}(0) \right) + (\mu \nabla^\text{MD} C - \rho g_y \nabla H) \Gamma_{\alpha}] |(x - \mathbf{e}_\alpha \delta t, t - \delta t), \quad \text{and} \tag{13}
\]

\[
\tilde{g}_\alpha(x, t) = \nabla \cdot (\mathbf{u} \mathbf{C}) = M \nabla^2 \mu, \tag{12}
\]
2.1.3. scheme because of the nodes.

The relaxation time $\tau = \lambda / \delta t$ is related to the kinematic viscosity by $v = \tau C_s^2 \delta t$, and is prescribed by $1 / \tau = C / \tau_1 + (1 - C) / \tau_2$, where $\tau_1$ and $\tau_2$ correspond to the bulk kinematic viscosities of the respective fluids. Gravity is denoted by $g_y$, and the term $\nabla H$ is the gradient of the height, which is just a unit vector in the vertical direction. The macroscopic variables are computed from respective moments of the distribution function:

$$ C = \sum_{\alpha} \bar{h}_{\alpha}, $$

$$ \rho \mathbf{u} = \sum_{\alpha} \mathbf{e}_{\alpha} \bar{g}_{\alpha} + \frac{\delta t}{2} \mu \nabla^{CD} C, $$

$$ p = \bar{g}_{\alpha} + \frac{\delta t}{2} \mathbf{u} \cdot \nabla^{CD} \rho c_s^2. $$

The superscript labels MD and CD refer to mixed difference and central difference gradients [53,55].

2.1.3. Boundary conditions

Given the particle position and velocity, boundary conditions on the fluid can be imposed. This section describes how the fluid behaves in response to the particle motion, i.e., the first half of the two-way coupling.

A bounce-back-on-the-link (BBL) scheme is used to determine unknown distribution functions streaming from wall nodes. This scheme assumes that the location of the boundary is located half-way along a link connecting one node that is in the fluid domain to one that is inside the solid as depicted in Fig. 1. This results in a staircase shaped representation of curved boundaries, which is first-order accurate. A first-order accurate boundary treatment degrades the overall order of the algorithm as it is second-order accurate in the fluid domain, but a more sophisticated boundary treatment was avoided because the simple bounce-back scheme minimizes changes in mass.

Ladd first performed lattice Boltzmann simulations with finite-sized particles, and demonstrated that the bounce-back algorithm must be supplemented for the case of moving walls so that the fluid velocity matches the wall velocity [27]. The scheme is written:

$$ f_{\tilde{\alpha}} = f_{\alpha} + 2 \frac{\rho}{c_s^2} t_{\alpha} (\mathbf{u}_B \cdot \mathbf{e}_{\tilde{\alpha}}) $$

Here $\mathbf{u}_B$ is the boundary velocity which can be computed from the particle position and velocity. Using our modified distribution functions, the scheme is written:

$$ \bar{f}_{\tilde{\alpha}} = \bar{f}_{\alpha} + 2 \rho t_{\alpha} (\mathbf{u}_B \cdot \mathbf{e}_{\tilde{\alpha}}), \quad \text{and} $$

$$ \bar{h}_{\tilde{\alpha}} = \bar{h}_{\alpha} + 2 \frac{C}{c_s^2} t_{\alpha} (\mathbf{u}_B \cdot \mathbf{e}_{\tilde{\alpha}}). $$

Fig. 1. A representation of a curved boundary mapped onto the lattice. The dashed line represents the theoretical boundary of a curved wall such as a particle. The actual boundary points, designated $x_0$ (filled black circles), are located half-way along links that connect a fluid node $x_f$ (open circle) to an interior solid node $x_s$ (filled gray circle) along a direction $e_{\alpha} \delta t$. The direction $e_{\alpha} \delta t$ correspondingly connects the solid node to the fluid node. The $\delta t$'s were omitted from the figure for clarity.
The evaluation of gradients and Laplacians also requires boundary conditions. To avoid unphysical mass and momentum transfer through the boundaries, a normal-gradient-free boundary condition was applied for any variable unrelated to the free energy. Since a wall is located half-way along a link connecting a fluid node to a solid node, as seen in Fig. 2, a normal-gradient-free condition along a link \( \delta t \mathbf{e}_y \cdot \nabla \phi |_{s} = 0 \) can be obtained by assuming a mirror image profile for any fluid variable inside the wall, and proceeding with the calculation as if that node were in the fluid domain [55]. For example, if \( \mathbf{x} \) is in the fluid domain and \( \mathbf{x} + \mathbf{e}_y \delta t \) is inside the solid, setting \( \phi(\mathbf{x} + \mathbf{e}_y \delta t) = \phi(\mathbf{x}) \) will achieve this goal. Furthermore, if \( \mathbf{x} - \mathbf{e}_y \delta t \) and \( \mathbf{x} + 2\mathbf{e}_y \delta t \) are in the fluid and solid domain, respectively, we set \( \phi(\mathbf{x} + 2\mathbf{e}_y \delta t) = \phi(\mathbf{x} - \mathbf{e}_y \delta t) \). This type of substitution is performed whenever the directional derivative for a gradient or Laplacian is evaluated near a wall.

The exception is when evaluating \( \nabla^2 C \) near a wall when calculating the chemical potential. In this case the boundary condition of Eq. (4) must be utilized. To compute \( \nabla^2 C \), the directional derivative along \( \mathbf{e}_y \) must be computed [53]. This requires the value of \( C \) at the nodes \( \mathbf{x}, \mathbf{x} + \mathbf{e}_y \delta t, \) and \( \mathbf{x} - \mathbf{e}_y \delta t \). However, \( \mathbf{x} + \mathbf{e}_y \delta t \) is located inside the solid. Normally we would prescribe \( C(\mathbf{x} + \mathbf{e}_y \delta t) = C(\mathbf{x}) \) as with other variables, but in this case we impose Eq. (4). We do that by evaluating a simple finite difference approximation to the normal gradient at the surface:

\[
n \cdot \nabla C|_s = \frac{C(\mathbf{x}) - C(\mathbf{x} + \mathbf{e}_y \delta t)}{\delta x}.
\]

We now solve for the unknown composition while using Eq. (4).

\[
C(\mathbf{x} + \mathbf{e}_y \delta t) = C(\mathbf{x}) - \delta x n \cdot \nabla C|_s = C(\mathbf{x}) - \delta x \frac{\partial C}{\partial C_2} (C_s - C_2^2).
\]

Any values of unknown compositions inside walls are evaluated this way [53]. One caveat is that the normal might not coincide with the direction of the directional derivative, or at a sharp corner might not be defined. Therefore, the normal direction in Eq. (22) is always assumed to coincide with \( \mathbf{e}_y \). It is important to realize that Eq. (22) is how contributions from the wall–interface interaction are included in the evaluation of the free energy in Eq. (1).

### 2.2. Particle dynamics

This section describes the second half of the two-way coupling, i.e. how the particles move in response to the fluid. The forces from the fluid on the particle must first be calculated. Then the particle position and velocity are updated according to Newtonian dynamics. Pressure, viscous stress, and surface tension contribute to the total force on a particle. The forces and torques from the fluid are computed by ME [27]. Due to our normal gradient-free boundary conditions, a supplemental force was introduced to account for the effects of interfacial tension. Also, several artifacts of particles moving over a fixed Cartesian grid must be addressed. A lubrication and/or spring force was added to particles near contact and an algorithm for specifying fluid properties at nodes uncovered by the particle motion was implemented.

#### 2.2.1. Momentum exchange

As seen in Section 2.1.3, boundary conditions for the distribution functions are applied when they are near walls. Distribution functions stream from their node of origin, hit a wall (macroscopic particle) located half-way along a link, and bounce back to the original node during a time step, but traveling in the opposite direction. Effectively, the distribution function changes its "momentum" from \( \vec{\theta} \alpha \mathbf{e}_y \) to \( \vec{\theta} \alpha \mathbf{e}_y \), where \( \vec{\theta} \alpha \) is evaluated using Eq. (19), taking the boundary velocity into account. Therefore the change in momentum of the distribution function is \( \Delta p = \vec{\theta} \alpha \mathbf{e}_y - \vec{\theta} \alpha \mathbf{e}_y = (\vec{\theta} \alpha + \vec{\theta} \alpha) \mathbf{e}_y \). For the macroscopic particle to cause this change in momentum of the distribution function, an equal and opposite change in momentum must be applied to the macroscopic particle. So for any link \( \alpha \xi \), that is cut by the boundary surface at \( \mathbf{x}_b \), an impulse of force and corresponding torque is applied to the macroscopic particle:

\[
F(\alpha \xi; \mathbf{x}_b) = (\vec{\theta} \alpha + \vec{\theta} \alpha) \mathbf{e}_y, \quad \text{and}
\]

\[
T(\alpha \xi; \mathbf{x}_b) = (\mathbf{x}_b - \mathbf{X}) \times F(\alpha \xi; \mathbf{x}_b).
\]
where $\mathbf{x}_b$ is the boundary node along a link where $\mathbf{e}_d \Delta t$ connects a fluid node to a solid node, and $\mathbf{X}$ is the center of mass of the particle. The total force on a particle is computed by taking the summation of all impulses that were computed: $\mathbf{F}_T = \sum_{d} \sum_{b} \mathbf{F}(\alpha_d; \mathbf{x}_b)$ and $\mathbf{T}_T = \sum_{d} \sum_{b} (\mathbf{x}_b - \mathbf{X}) \times \mathbf{F}(\alpha_d; \mathbf{x}_b)$.

It was found that the algorithm is unstable if $\mathbf{g}_\partial$ is computed explicitly, i.e. using the particle velocity of the current time step. It was shown that a stable scheme can be achieved if the particle velocity at the next time step is calculated implicitly, and the resulting velocity used to compute the boundary velocity in Eqs. (19) and (20). This requires two “sweeps” through the domain; the first sweep assembles the data needed to solve for the velocities implicitly, and the second sweep uses the new velocities to impose boundary conditions (19) and (20). The details are specified clearly by Joshi and Sun [48].

2.2.2. Surface tension force

Surface tension forces are responsible for phenomena such as fluid rise in capillary tubes and objects floating on the surface of fluid less dense than the object. We demonstrate in Section 3.2 that, due to our boundary conditions, our implementation of ME resolves pressure and viscous stress, but not the surface tension force. It becomes necessary to add a supplementary force to include this effect. This section describes the algorithm that computes the surface tension force on a particle.

Surface tension at a fluid–fluid interface can be interpreted as a force per unit length. If an interface intersects the surface of a particle, known as the three-phase contact line, the interface exerts a force on that particle of the form [9]

$$F_{ST} = \int_{C} \sigma \mathbf{m} \, d\ell,$$  

where $d\ell$ is a line element tangent to the contact line $C$, a closed path, and $\mathbf{m}$ is a unit vector tangent to the interface locally. This is not a trivial integral to evaluate numerically, as the problem is to locate a contact line (Fig. 3), discretize the line into elements, and determine the tangent to the interface at each element. We present two iterations of our scheme, where the first illustrates the ideas that lead to the second, which is more suitable for use in a diffuse interface scheme.

In general, we do not assume anything about the location of the contact line; therefore our algorithm must be able to find it. We sweep through the entire domain (can be performed at same time as the ME sweep) to test each lattice cube to see if it contains a line element. The cube must meet two criteria to contain a line element. 1) It must contain the interface, i.e. at least one of the eight nodes defining the cube must have $C < 0.5$ and at least one other node must have $C > 0.5$. 2) It must contain at least one particle boundary node. A particular cube that meets both criteria is pictured in Fig. 4. It contains four nodes in the fluid domain and four inside the particle, as can be ascertained by the orthogonal views. Two of the four fluid nodes lie below the interface. This first iteration of the algorithm is executed as if a sharp interface were located on the $C = 0.5$ surface.

There are many possible local tangents to the interface for a particular line element, but we choose the tangent that is in the plane defined by the normal to the interface and the normal to the particle surface at that location. The normal to the interface is the gradient of the composition by definition, which is a quantity available from previous calculations (Section 2.1). We use the average of $\nabla C$ taken from all fluid nodes of the cube. The average of $(−\nabla C)$ is pictured in Fig. 4 for illustrative purposes, where the averaged vector is denoted as $\mathbf{A}$. Taking $\mathbf{A}$ to be the average of $\nabla C$ or $−\nabla C$ makes no difference, as long as the final vector $\mathbf{m}$ does not point into the particle. To avoid the additional effort of determining the exact location of the contact line inside the cube (usually not as important as interface tangent orientation), we assume the origin of all vectors associated with this calculation are located in the center of the cube. We further assume that the magnitude of the contact line element inside the cube $d\ell \approx 1$. The normal vector is defined as $\mathbf{N} = \mathbf{x}_c - \mathbf{X}$, where $\mathbf{x}_c$ is the position of the center of the cube.

To determine the vector that is both tangent to the interface and in the plane defined by $\mathbf{N}$ and $\mathbf{A}$, we take a combination of cross products of the available vectors. The vector $\mathbf{A} \times \mathbf{N}$ is orthogonal to the $\mathbf{A} \cdot \mathbf{N}$ plane. Therefore it is tangent to the
interface, but also tangent to the particle surface. Taking an additional cross product, \( \mathbf{m} = (\mathbf{A} \times \mathbf{N}) \times \mathbf{A} \), gives us a vector that is in the \( \mathbf{A} \cdot \mathbf{N} \) plane and is tangent to the interface. The total surface tension force is the sum of contributions from all cubes containing line elements. A separate force is calculated and applied for each particle. It may happen that wall nodes corresponding to two distinct particles are part of the interface element cube. The contribution from the interface element is added to the force calculated for both particles in that case.

The force that is calculated by this algorithm is dependent on the configuration of the contact line relative to the lattice, and is therefore prone to large fluctuations. For example, as the particle moves, the cube that calculates a contribution from a particular line element could change, leading to a spike in the force. As a second iteration, we tweak the algorithm by using information throughout the diffuse interface to perform a weighted integral. Instead of computing the force corresponding to each line element from the single cube that meets the above criteria, we take a weighted average of contributions across the diffuse interface (normal to the contact line) to smooth the computation. We use a weighted average to give more importance to contributions near the \( C = 0.5 \) surface (by \( C \) we mean the cube-averaged \( C \) as discussed above). In effect, we perform the same algorithm as before, except we relax criterion 1 so that any cube near the surface of the particle contributes to the total integral. The weight function, a function of the composition (explained below), goes to zero as \( C \to 0 \) or \( C \to 1 \), which effectively discounts the contribution from any cube that is not in the interface.

It is important to remember that the discretization of Eq. (25) is, figuratively, a two-step process (Fig. 5): 1) the contact line is discretized into line elements, and 2) the contribution from each element is averaged across the interface. However, this two-step process is not to be taken literally. We do not first find the line element and then average across the interface; the algorithm is performed as a generic sweep through the domain. It is not necessary to determine the line element to which a particular cube belongs because all the contributions are added together anyway. The critical assumption is that the number of cubes that participate in the “average” is the size of the diffuse interface, which is not rigorously guaranteed, but is accurate most of the time. The overall effect of the total numerical integration is effectively the same as the two-step process.

We compute the average surface tension force for each element by taking the integral

\[
F_{dt} = \frac{\int_{C=0}^{1} w(C) f(C) dC}{\int_{C=0}^{1} w(C) dC},
\]

where \( F_{dt} \) is the force contribution for the element \( dt \) resulting from the integral average, \( w(C) \) is the weight function, and \( f(C) \) is the continuous analog of the surface tension force described previously. If the weight function is normalized, i.e. \( \int_{C=0}^{1} w(C) dC = 1 \), Eq. (26) can be approximated as

\[
F_{dt} \approx \frac{1}{N} \sum_{i=1}^{N} w(C_i) f_i(C_i),
\]

where the sum is taken over \( N \) cubes and \( f_i \) is the contribution from each cube. We assume that \( N \) is equal to the size of the interface, i.e. 5 in this paper.

To determine the weight for each cube, we would need to know its location in the interface (center of the interface gets more weight), but since we do not assume anything about the orientation of the contact line, we cannot know this location a priori. The only information we have is the average composition for each cube. Since \( C \) varies like a hyperbolic tangent

\[
\int C < 0.5
\]

\[
\int C > 0.5
\]
Fig. 5. An illustration of the discretization process. The contact line, $C$, is discretized into line elements. They are not necessarily the evenly spaced ones depicted here for illustrative purposes as they are dependent on the configuration inside the lattice. For each line element, the force is computed by taking the average from all cubes across the interface. The $C = 0.1–0.9$ surfaces of the interface on the particle are depicted, with the $C = 0.5$ surface drawn with the thickest line.

Fig. 6. The weight function of Eq. (29) favors contributions to the surface tension force from the center of the interface, while avoiding contributions from outside the interface.

(cf. Section 2.1.1) across the interface, we assume the location, and therefore the weight, varies like the inverse hyperbolic tangent:

$$w(C) = A \tanh^{-1}(2C - 1) + B.$$  \hspace{1cm} (28)

We specify that $w(C \leq 0.1) = 0$ and $w(C \geq 0.9) = 0$ to discount the contribution of cubes outside the interface. This function diverges as $C \to 1$, so we cut the curve off at $C = 0.5$ and use the mirror image to obtain a symmetric weight function over the interval $[0, 1]$. To specify the weight function completely, we need to normalize it: $\int_0^{0.5} w(C) dC = 1$. The weight function we use is shown in Fig. 6 and is written

$$w(C) = 1.9595 \tanh^{-1}(2C - 1) + 2.153.$$ \hspace{1cm} (29)

Once the weight function is determined, it becomes a trivial function call during the simulation; if $C > 0.5$, $C$ gets mapped to $1 - C$. We will show that this second iteration of the algorithm produces a smoothed version of the first iteration result.

Below is a pseudocode for this part of the algorithm. It starts by searching each cube of the domain, and if one of the eight nodes of the cube is inside a particle:

- Store particle number $p$
- Average $C$ from all fluid nodes
  - Compute $w(C)$ from Eq. (29)
- Average $\nabla C$ from all fluid nodes to obtain $A$
- $N$ is vector from $X_p$ to center of cube
- Compute $m = (A \times N) \times A$
- $STForce(p) = STForce(p) + w(C) \sigma m / \xi$
In the last line, $\text{STForce}$ is the vector surface tension force applied to particle $p$, $w(C)$ is the weight function, $\sigma$ is the surface tension, $m$ is the vector tangent to the interface, and $x_i$ is the interface thickness.

3. Results

The multi-component algorithm [55] and the algorithm that transports solid particles [27] in a single-component carrier fluid have been validated previously. The case where the multi-component algorithm is combined with particle dynamics must now be validated. We first find that particles in the bulk of a particular component behave as in the case of a single component carrier fluid. Then we demonstrate that it is necessary to include the supplemental force detailed in Section 2.2.2 to recover surface tension effects. We show that a particle reaches the correct equilibrium position when placed in the interface of a drop, without gravity. Finally, we show that our algorithm qualitatively recovers the dynamics recorded by an experiment of two particles approaching each other on a flat interface due to capillary interactions.

3.1. Single particle inside bulk

There have been a few LBM studies of particles in multi-component flows (see Section 1 for descriptions and references), but one of the major drawbacks of these works is that they cannot remove (or suppress to irrelevance) the detrimental artifact of spurious currents. A spurious current is a small-amplitude artificial velocity field which arises from an imbalance between discretized forces at curved interfaces in multi-component flows [54]. If the multi-component algorithm is not designed to eliminate this phenomenon, the spurious velocity will persist indefinitely, preventing the achievement of a true equilibrium state. In some instances, the spurious velocity can be of the same order of magnitude as the relevant velocities of the flow, resulting in ambiguity. Spurious currents are typically exacerbated by large values of numerical surface tension or large differences in the density of the two fluid components. In Section 2.1, we detailed a number of important steps that were taken in order to remove this unwanted phenomenon at equilibrium. It should be noted that all multi-component diffuse interface LBM algorithms suffer from spurious currents, but our method, as well as some other approaches, is formulated such that spurious currents decay to machine error at equilibrium when the solid boundaries are fixed. When solid boundaries (particles) are free to move based on applied forces, the spurious currents in our simulations do not necessarily decay to machine error, but are suppressed to the point that they do not interfere with the dynamics of the simulation. We show here that spurious currents decay towards zero when a particle resides in a region of bulk fluid.

The problem of persisting spurious currents in a multi-component simulation with particles was illustrated by Joshi and Sun [47]. In their simulation, they placed a drop in a domain with periodic boundary conditions. The system was initially at rest, but spurious currents developed due to reasons specified above. For a particle located in the bulk vapor region, the spurious currents that developed carried the particle into the interface. Even though the interface is a region of minimum energy, if the particle is not initially immersed in the interface for a system that is initiated, and is supposed to remain, at rest, the particle should not be driven either towards or away from the interface.

We use our method to simulate a particle that is initially placed in the bulk fluid region of a drop surrounded by ambient air (Fig. 7). The domain is $X \times Y \times Z = 100 \times 100 \times 200$ in lattice units. The drop radius, $R_d = 50$, while the particle radius, $R_p = 5$, also in lattice units. The density ratio between the liquid and gas is $\rho_l/\rho_g = 842$. Symmetry conditions are applied at the domain boundaries. The drop is initially located at coordinates $(0, 0, 100)$, utilizing the symmetry boundaries. The particle center is initially located at $X_0 = (15, 15, 100)$. The simulation is initiated from quiescent conditions. Here a viscous timescale is defined as $t_v = \nu / \sigma = 500$ lattice time units. The maximum velocity found in the system, a measure of the magnitude of spurious currents, continuously decreases with time (Fig. 8), indicating that the presence of the particle in
the bulk does not interfere with the suppression of spurious currents. The radial position of the particle (Fig. 9) is affected by the spurious currents initially, but the motion decreases towards zero as the spurious currents decrease. The particle has moved less than 5% of one lattice unit after nearly 2000 viscous time units indicating that the spurious currents do not have a strong impact on the particle motion in the bulk regions.

3.2. Capillary tube

We claimed in Section 2.2.2 that, most likely due to our normal gradient free boundary conditions, our implementation of ME does not recover the surface tension force on a solid object. In this section, we verify this statement through a ME force calculation for a simple case with fixed boundaries, where solid motion is not coupled to the force calculation.

A capillary tube (Fig. 10) draws liquid up/down its interior by surface tension effects [69]. As a result, a column of fluid rises/falls inside the tube until the surface tension force pulling the liquid up/down is balanced by the gravitationally induced pressure difference across the interface. For a 2D geometry between parallel plates at equilibrium, the balance of forces (per unit length) is written:

\[ 2\sigma \cos \theta = 2w \rho g H, \]  

where \( \sigma \) is the surface tension, \( \theta \) is the contact angle, \( \rho \) is the density of the liquid, \( w \) is half the separation between plates, and \( H \) is the height of the fluid column. Eq. (30) is only valid when the spacing between plates is small compared to the capillary length; \( w \ll \sqrt{\sigma / \rho g} \). The column rises or falls depending on whether the contact angle is hydrophilic or hydrophobic, respectively. We take the case of a hydrophobic contact angle throughout the rest of this section, and continue to use the label "capillary tube" where appropriate, despite the two-dimensional geometry, for expediency.
Since there is no motion at steady state, there are no viscous stresses acting on the walls. Furthermore, since the surface tension force is exerted by the wall on the fluid to pull it down, there must be an equal and opposite force exerted by the fluid on the wall. The horizontal components of the surface tension force and pressure force each vanish due to symmetry. However, a vertical component of the surface tension force remains, as stated in Eq. (30).

We performed an LBM simulation of a capillary tube using the techniques of Section 2. This is a good case to test the force calculation because the boundaries are fixed, effectively simplifying the interaction to a one-way coupling. The simulation box was $80 \times 380$, with the vertical direction being the longest. The density ratio between fluids was $\rho_l/\rho_g = 842$ with $\rho_l = 1.0$ in lattice units, $\sigma = 0.008$ in lattice units, the contact angle $\theta = 100^\circ$, and the tube was 300 lattice units tall starting at $y = 40$, with a half-width $w = 12.5$ lattice units. The values of density and viscosity in our test simulations were chosen to be similar to those of a water–air system, representative of any system with large property contrasts. For this simulation, $\sigma$ was chosen to achieve the desired height of our liquid column, given the other geometrical parameters and the chosen density. Throughout all these test simulations, it may be useful to visualize a water–air system, though it should be kept in mind that the properties do not correspond exactly. We match dimensionless variables for the validation case of Section 3.4 as we wanted to correspond to the values used in the experiment directly.

The original liquid level was $H/4$ the height of the tube, and we specified that the desired height of the column was $H = 200$, and determined what value for $g$ should recover it. As fluid fell between the plates, it rose outside the plates as a result of mass conservation. Half the domain was simulated, with symmetry boundary conditions on the side boundaries. The bottom boundary was a wall, and the top boundary was also a symmetric boundary condition. The vertical component of the ME force was computed on the inner wall surface using Eq. (23). We did not want to include end effects, so the force was calculated on the entire inner surface of the plate, except for one interface thickness away from the top and bottom of the plate, respectively. At equilibrium, the only force contribution should come from the interface region, which was sufficiently far from the tube ends.

At steady state (Fig. 11), the theoretical capillary descent was reached (Fig. 12(a)), but the vertical component of force on the tube walls computed by ME dropped to zero (Fig. 12(b)). The non-negligible force computed during the transient part of the simulation is the viscous drag as the column descended; once the motion disappeared, so did the viscous drag. Therefore we conclude that our implementation of ME captures forces that arise from pressure and viscous stress, but not surface tension.

We augmented the ME algorithm by computing the surface tension force on the tube using the method of Section 2.2.2. The computed force (Fig. 13) oscillated about the expected value before the motion stopped. The oscillation results from a change in configuration of lattice nodes used in the force calculation as the interface slid down the wall. The inset compares the second iteration of the algorithm to the first, where the original algorithm possessed significantly more oscillation. This demonstrates that our algorithm recovers the expected surface tension force on fixed objects. In the next section, we employ this algorithm on a freely moving object, and show that the coupling of force and motion produce the expected equilibrium state.

3.3. Single particle immersed in an interface

In this section, we validate our model for the case of a particle immersed in a fluid–fluid interface. In this case, unlike the capillary tube, the solid body motion was coupled to the force calculation algorithm. We will show that not only did the surface tension force allow us to reach a steady state, but that the appropriate configuration was attained at equilibrium.
Consider the case of a particle immersed in the interface of a drop, without gravity, at equilibrium (Fig. 14). Since an interface is a region of minimum energy [6–8], a particle that is initially immersed in an interface should remain there. Moreover, it is possible to predict the depth of immersion into the drop as a function of the relevant parameters.

The problem is defined by the particle radius $R$, the drop radius $R_0$, the contact angle $\theta$, and the fluid–fluid surface tension $\sigma$. The difference in pressure on the inside and outside of the drop is prescribed by $P_i - P_o = 2\sigma/R_0$. We assume the particle is at its equilibrium immersion depth $d$, which fixes the contact line radius $a$ and the angle $\alpha$ the surface tension force makes with the horizontal $\alpha$. We also assume that the drop is spherical, i.e. there is no indentation due to the particle.

At equilibrium, all forces on the particle are balanced. Lateral forces balance due to symmetry. Pressure forces in the outward radial direction are balanced by surface tension forces. In particular, the radial force balance is written

$$\pi a^2(P_i - P_o) = 2\pi a \sigma \sin \alpha.$$  \hfill (31)

The pressure force acts over a projected circular area of radius $a$, the contact line radius, not the particle radius $R$ [9]. Also, the radial component of surface tension force depends on $a$, not merely the contact angle $\theta$. In fact, $a$ is not uniquely determined by the contact angle, which is specified a priori, but by the combination of the contact angle and the depth of immersion. Since $d$ also determines $a$, it becomes a convenient variable to specify the equilibrium state, as there is a unique depth where the surface tension force balances the pressure force.

If the particle is displaced slightly into the drop (relative to the equilibrium position), the pressure force (pushing out) will be larger than the surface tension force (pulling in), and the particle will move back to its equilibrium position. In the same manner, a displacement slightly out of the drop will lead to an unbalanced force pushing the particle back inside until it reaches the equilibrium position. Therefore, the equilibrium position is stable, and any particle immersed in the interface should eventually reach this depth of immersion.

Given that $a$ is determined by $d$, $\alpha$ is determined by $d$ and $\theta$, and the pressure difference can be written in terms of the surface tension and drop radius, Eq. (31) can be rearranged to yield an equation for the depth of immersion as a function of purely geometric parameters:

$$\left( \frac{d}{R} \right) = 1 - \frac{\cos \theta - \left( \frac{R}{R_0} \right)}{\sqrt{1 - 2\left( \frac{R}{R_0} \right) \cos \theta + \left( \frac{R}{R_0} \right)^2}}.$$  \hfill (32)

A flat interface is defined by $R_0 \rightarrow \infty$, where $R/R_0 \rightarrow 0$. For this case, Eq. (32) can be simplified:

$$\left( \frac{d}{R} \right) = 1 - \cos \theta.$$  \hfill (33)

For a flat interface, a particle with $90^\circ$ contact angle will sit exactly half way into the interface. This makes sense as there is no pressure difference driving a particle out, and the surface tension force has no vertical component. For a particle with $\theta = 90^\circ$ in a drop, $d/R$ must be greater than unity; i.e. it is impossible to have a particle sitting exactly half in the interface, as this would produce no radial component of surface tension force, and the pressure force would be unbalanced.
Fig. 12. Normalized height and vertical force as a function of lattice time. The difference in height between the liquid column and the free surface was determined by finding the height of the liquid level in the center of the column and out in the free surface, which was then normalized by the theoretical value based on the input parameters. The vertical component of the force computed by ME was normalized by the expected result based on the input parameters. The theoretical height was reached, indicating that the fluid behaved as expected, but the computed force decayed to zero at steady state. The force became negligible when the column stopped descending, indicating that the computed vertical force component arises from viscous drag, and does not capture the surface tension force.

Fig. 13. The normalized vertical component of force computed from the surface tension force algorithm of Section 2.2.2 as a function of lattice time. The computed force was normalized by the expected value based on the input parameters. This demonstrates that the algorithm captures the surface tension force. The inset compares the second iteration of the algorithm (red color, small amplitude), as in the full plot, to the first iteration (green color, large amplitude). While both results oscillated about approximately the same value, the original algorithm’s result oscillated with a much greater amplitude.
We performed several simulations where a drop was wetted to a substrate, and a particle was brought into contact with the interface (Fig. 15). Then the particle was released, and an equilibrium state was reached where the depth of immersion was compared to the theoretical result (Fig. 16). The drop was wetted to a substrate because it must be anchored, otherwise the coupled drop-particle motion will cause the system to drift. The presence of the substrate does not invalidate Eq. (32), as the difference between internal and external drop pressure only depends on the surface tension and curvature of the interface. We brought the particle into contact with the drop instead of initializing its position in the interface so that the mass of the drop remained the same for different sized particles.

At equilibrium, the depth of immersion matched the theory best for $\theta = 90^\circ$. This is because the contact line has the highest resolution, i.e. $a \approx R$. We cannot claim that our surface tension force, combined with the entire algorithm, is validated for the entire transient part of the simulation. This is just as we cannot claim that our multi-component algorithm always resolves the exact dynamic contact angle. However, for this simple case, we can claim that our algorithm is validated at equilibrium.

A major concern in diffuse interface methods is the variability of simulations when different sized interfaces are used, i.e. convergence. We require a minimum of 4–5 lattice units to represent the interface, below which gradients would not be calculated accurately. On the other hand, we want the diffuse interface to approximate a sharp interface, so we do not want to increase it any further. Since we are in essence compelled to use $\xi \approx 5$, we can approach the “sharp interface limit” [10] by increasing all other dimensions of the problem, while keeping the interface fixed. This is quantified by a Cahn Number $Cn = \xi / L$, where $L$ is a representative length scale of the problem; in any particle dynamics simulation accounting for effects of the interface on the particle, the particle radius should be used as this length scale. As $Cn$ becomes sufficiently small, the effects of the diffuseness of the interface should vanish, and results should remain the same for any $Cn$ smaller than a critical value. There is no universal rule for determining the critical value, as it is highly dependent on the situation. We have therefore run a set of simulations where we have doubled all dimensions (low-$R = 15$ lattice units, high-$R = 30$ lattice
units), keeping the interface fixed (5 lattice units). The results mostly achieve the same values for this higher resolution, lower Cn case, which demonstrates convergence.

Finally, the system was not able to eliminate the spurious currents completely when the particle was free to move (Fig. 17). There was always a small discrepancy between the pressure and surface tension forces that caused a persistent oscillatory motion, inhibiting the spurious currents from decreasing further. The oscillations in particle position were very small compared to the particle size though (Fig. 18), so it can be said that the system was able to reach an approximate “equilibrium” where the lingering motion was negligible. The case where the particle is held fixed after the approximate equilibrium state was reached is also plotted in the same figures. It can be seen that spurious currents are eliminated when the geometry is held fixed (Fig. 17) and that the position of the free particle oscillates about the fixed particle position with a small amplitude (Fig. 18). We can conclude that spurious currents are eliminated for our method when particles are relegated to bulk fluid regions, and that they are suppressed to the point of irrelevance when particles are in the interface. Simply holding the “equilibrium” position fixed allows the spurious currents to be eliminated.

3.4. Particle approach on a flat interface due to capillary interactions: comparison with experiment

In this section, we compare the full transient dynamics of our simulations to the experiment of Dalbe et al. [56] There are many experimental studies that measure the capillary force between two stationary objects, and consequently, several LBM studies have used a similar setup to validate their code. While this is important, we seek to differentiate our work by comparison with a dynamic experiment. There was only one experiment, to the best of our knowledge, that measured the transient response to a capillary force. The experiment is straightforward (though the physics involved is much more complicated), well-controlled, and employed a large density ratio (a glycerol–water/air system).
Our results are not in perfect agreement with the experiment, most likely because the experiment was performed under conditions that were convenient for matching a theoretical description (certain parameters are small), but were inconvenient to implement as a numerical experiment, as will be discussed below. Though our results do not quantitatively match the experiment exactly, an imperfect comparison is still worthwhile as it raises awareness of numerical issues such as the resolution of the contact line, and a separation of scales between the diffuse interface and particle (sharp interface limit [10]). We feel that there is certainly room for improvement in terms of transient validation, but that simulations using our algorithm can be considered accurate under conditions that avoid the inherent numerical limitations we hope to elucidate in this section.

Consider the case of two non-neutrally buoyant polyethylene particles immersed in a flat interface between a glycerol–water mixture and air, contained in a large tank (Fig. 19). The particles have a radius $R_p = 1.5875$ mm, density $\rho_p = 950$ kg/m$^3$, and a contact angle $\theta_p = 20^\circ$. The density of the mixture and air at 23°C are $\rho_l = 1233$ kg/m$^3$ and $\rho_g = 1.1925$ kg/m$^3$, respectively; the viscosities are $\mu_l = 0.175$ Pa s and $\mu_g = 1.835 \times 10^{-5}$ Pa s; and the fluid–air surface tension is $\sigma = 0.064$ N m$^{-1}$. This yielded a capillary length $l_c = \sqrt{\sigma/(\rho g)} = 2.30$ mm. Due to their buoyancy and wetting properties, the particles deformed (pushed up) the interface locally. If the particles are sufficiently close to each other, the individual ranges of capillary effects interfere in the region between the particles, which creates an unbalanced capillary force that drives the particles toward each other [9].

In the experiment, the particles were submerged, held at a fixed separation while they attained their vertical equilibrium at the interface (buoyant particles), and then released. The particles moved toward each other, while both their separation and relative velocity were recorded, until they contacted. One conclusion of Dalbe et al.’s study is that the relative approach velocity does not depend on the initial separation, only on the current separation, after some initial transient is passed after they are released. This allowed us to limit our study to dynamics relatively near contact, and therefore use a large spatial resolution.

The dimensionless numbers used in the simulation must match those used in the experiment. The parameters that characterize the problem are the densities of the liquid ($\rho_l$), gas ($\rho_g$), and particle ($\rho_p$); the viscosities of the liquid ($\mu_l$) and gas ($\mu_g$); the fluid–fluid surface tension ($\sigma$); the particle radius ($R_p$); and gravity ($g$). This required us to match
Table 1
The parameters of the experiment and simulation. In the experiment, the liquid was a mixture of glycerol–water and the gas was air at 23°C. There was a discrepancy in the size of the tank in physical/lattice units, where the simulation box was significantly smaller to allow the simulation to finish in a reasonable time. All sides of the box were about a capillary length away (at least) from the particles, so the effects of using a smaller tank should be minimal. The length of the box in physical units, or the number of lattice units, is listed for the experiment and simulation, respectively.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Physical units</th>
<th>Lattice units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimensional parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Liquid density</td>
<td>( \rho )</td>
<td>1233 kg/m(^3)</td>
<td>1.0</td>
</tr>
<tr>
<td>Gas density</td>
<td>( \rho_g )</td>
<td>1.1925 kg/m(^3)</td>
<td>9.67 \times 10^{-4}</td>
</tr>
<tr>
<td>Particle density</td>
<td>( \rho_p )</td>
<td>950 kg/m(^3)</td>
<td>0.77</td>
</tr>
<tr>
<td>Liquid dynamic viscosity</td>
<td>( \eta_l )</td>
<td>0.175 Pa s</td>
<td>0.35</td>
</tr>
<tr>
<td>Gas dynamic viscosity</td>
<td>( \eta_g )</td>
<td>1.835 \times 10^{-5} Pa s</td>
<td>3.67 \times 10^{-5}</td>
</tr>
<tr>
<td>Surface tension</td>
<td>( \sigma )</td>
<td>0.064 N/m</td>
<td>0.01</td>
</tr>
<tr>
<td>Gravity</td>
<td>( g )</td>
<td>9.81 m/s(^2)</td>
<td>1.92 \times 10^{-6}</td>
</tr>
<tr>
<td>Particle radius</td>
<td>( R_p )</td>
<td>1.5875 mm</td>
<td>50</td>
</tr>
<tr>
<td>Tank length</td>
<td>( L_x(N_x) )</td>
<td>20.3 cm</td>
<td>600</td>
</tr>
<tr>
<td>Tank height</td>
<td>( L_y(N_y) )</td>
<td>2.54 cm</td>
<td>200</td>
</tr>
<tr>
<td>Tank width</td>
<td>( L_z(N_z) )</td>
<td>20.3 cm</td>
<td>200</td>
</tr>
<tr>
<td>Initial separation</td>
<td>( \ell_0 )</td>
<td>5 mm</td>
<td>158</td>
</tr>
<tr>
<td>Capillary length</td>
<td>( L_c = \sqrt{\sigma/(\rho_g \ell_0^3)} )</td>
<td>2.30 mm</td>
<td>72</td>
</tr>
<tr>
<td><strong>Dimensionless parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Particle contact angle</td>
<td>( \theta_p )</td>
<td>20°</td>
<td>20°</td>
</tr>
<tr>
<td>Liquid–gas density ratio</td>
<td>( \rho_l/\rho_g )</td>
<td>1034</td>
<td>1034</td>
</tr>
<tr>
<td>Particle–liquid density ratio</td>
<td>( \rho_p/\rho_l )</td>
<td>0.77</td>
<td>0.77</td>
</tr>
<tr>
<td>Viscosity ratio</td>
<td>( \eta_l/\eta_g )</td>
<td>9537</td>
<td>9537</td>
</tr>
<tr>
<td>Ohnesorge Number</td>
<td>( Oh = \eta_l/(\rho_l \sigma R_p) )</td>
<td>0.4944</td>
<td>0.4944</td>
</tr>
<tr>
<td>Bond Number</td>
<td>( Bo = \rho_l g R_p^2/\sigma )</td>
<td>0.4763</td>
<td>0.4763</td>
</tr>
<tr>
<td><strong>Numerical parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface Thickness</td>
<td>( \xi )</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Mobility</td>
<td>( M )</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Initial liquid height</td>
<td>( Y_0 )</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>Initial x-position</td>
<td>( X_1 )</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>Initial x-position</td>
<td>( X_2 )</td>
<td>379</td>
<td></td>
</tr>
<tr>
<td>Initial y-position</td>
<td>( Y_1 = Y_2 )</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>Initial z-position</td>
<td>( Z_1 = Z_2 )</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

five dimensionless numbers: the liquid–gas density ratio \( \rho_l/\rho_g = 1034 \), the particle–liquid density ratio \( \rho_p/\rho_l = 0.77 \), the liquid–gas viscosity ratio \( \mu_l/\mu_g = 9537 \), the Ohnesorge number \( Oh = \mu_l/(\sqrt{\rho_l \sigma R_p}) = 0.4944 \), and the Bond number \( Bo = \rho_l g R_p^2/\sigma = 0.4763 \). To specify the lattice parameters, we chose \( \rho_l = 1 \), \( R_p = 50 \), and \( \sigma = 0.01 \); then we used the dimensionless numbers to determine the remaining parameters, where the complete parameter set of the experiment and simulation is summarized in Table 1. The particle radius was represented by 50 lattice units, which was chosen to yield a large resolution of the particle surface. It should be emphasized that this resolution is impractically large, not intended to be used in simulations involving many particles, but was necessary for this special validation study.

Our grid was \( N_x \times N_y \times N_z = 600 \times 200 \times 200 \). The tank was filled 75% with liquid (interface was located at \( Y_0 = 150 \)), the \( y \)-direction being vertical. We used a combination of a completely wetting contact angle (5°) below the initial liquid height and a completely non-wetting contact angle (175°) above to effectively pin the contact line along the tank perimeter, as in the experiment. Two particles were placed with their centers at \( Y_1 = Y_2 = 94 \) (particle surface is barely submerged), \( Z_1 = Z_2 = 100 \) (symmetric), and \( X_1 = 221 \) and \( X_2 = 379 \) (initial center-to-center separation \( \ell_0 \) corresponding to 5 mm, or \( \ell_0/R_p = 3.16 \), symmetric about \( x = 300 \)), respectively. This box was much smaller than the one used in the experiment, but all side walls were far enough away from the particle surface, compared to the capillary length (\( L_c \approx 72 \) lattice units), such that the surface deformation was negligible along the tank contact line perimeter. The \( x \)-direction was chosen larger than the \( z \)-direction because forces in the latter cancel due to symmetry, and therefore should not be as important to the motion.

The particles were released vertically, computing all vertical forces but no lateral forces, for 200,000 time steps, until equilibrium was reached. The weight was added to the particle motion algorithm as an explicit force. Buoyancy arises by virtue of the gravitational forcing term applied to the fluid, which manifests itself in the ME algorithm; i.e. the gravitational forcing term causes a hydrostatic pressure gradient in the fluid, and ME captures any viscous forces. After vertical equilibrium was achieved, the particles were released horizontally, and the full forcing/motion algorithm was activated.

Unfortunately, the nature of the experiment was such that the particles did not disturb the interface much (Fig. 20). This allowed Dalbe et al. to develop a theoretical expression for the approach velocity, which was confirmed by their experiments, but it made for an inconvenient simulation. The problem was that the contact line looped over the very top of the particles, and even though the particle surface as a whole was highly resolved, there was still relatively insufficient resolution near the contact line. The result was that the lateral component of the capillary force, the force responsible for bringing the particles together, could not be computed with sufficient accuracy.
Specifically, there were two aspects that were not resolved properly. The first is that the line integral was allocated a poor discretization because the local contact radius was small. A contact line that loops over an “equator” of the particle would yield a better discretization. The second aspect is that the algorithm could not capture the subtle differences in height of the contact line (and therefore the difference in the local contact radius) in the regions between the particles and that between the particle and far wall, known as the inner and outer regions from here on (Fig. 21).

The unbalanced lateral force is accomplished by the contact line being asymmetric, or tilted [9]. The liquid level in the inner region is slightly higher on the particle surface than in the outer region for this buoyant particle case. This occurs because of the contact angle boundary conditions imposed simultaneously on both particles. If the second particle were not nearby, the interface on the inner region would sag almost exactly as in the outer region, and there would be no unbalanced lateral force. The presence of the second particle increases the curvature of the interface between particles, affecting pressure in the fluid. To relax this pressure, the contact line crawls up the inner surface of the particle so that the interface is less curved and the contact angle boundary condition is still met. The height difference is amplified as the particles get closer together, and is pictured as a schematic in Fig. 21 and a simulation result in Fig. 22. The implication of capturing this difference will be discussed below, but here we state that the tilt for this particular case was so subtle that this resolution could not capture the minute changes in height of the contact line.

Physically, the result of the asymmetry (tilt) is that the part of the contact line pulling towards the other particle is slightly greater than the part pulling towards the outer boundary. Numerically, this is effectively subtracting two large numbers to obtain a small number, which is prone to roundoff effects and is dependent on the configuration of the particle in the lattice. This same issue plagues the generic ME algorithm, resulting in an erratic force. However, as long as the resultant is not too subtle, ME produces an accurate trajectory despite the erratic force. There is a stark contrast between
the lateral component of the surface tension force and the vertical, which is in the same direction (negative y-direction) along the entire contact line, and therefore much smoother and more reliable. By visually comparing the vertical equilibrium as in Fig. 20, we see that the vertical component of the surface tension force was computed accurately.

As the particles get closer and the liquid level between them rises, there are two effects competing to pull the particles in opposite directions. First the local tangent to the interface in the inner region has a larger lateral component (Fig. 22) which acts to pull the particles together faster. On the other hand, the contact line height in the inner region rises, and the contact radius there becomes smaller. This effectively gives less weight to this region and acts to slow down the approach velocity, which explains the need to detect the difference in height properly.

In this particular case, the resultant lateral force was so subtle that the dynamics could not tolerate the insufficient resolution near the contact line. As a result, there were severe fluctuations in the lateral force, especially when the particle surface covered/uncovered lattice nodes, and our algorithm produced a much faster approach than was expected. What is really desirable is to have a grid with areas of enhanced mesh refinement. The region near the contact line needs to have an increased resolution, while the region away from the contact line could be computed accurately with a significantly coarser resolution. As this would require a major renovation of our code and it was impossible to increase the resolution further in a realistic way, it became necessary to replace the lateral force computation with an algorithm specifically tailored to this system; our original algorithm was still used to compute the vertical force.

As the calculation was very sensitive to the configuration of nodes used to compute contact line elements, we instead discretized the contact line using a quadrature of 360 points, separated by equal angles in the x-z plane. In this way, we guaranteed there were the same number of elements on each side of the x-midplane of the particle. In other words, there were the same number of elements contributing to the force pulling the particles together as there were pulling the particles toward the walls. Looking down at the particle emerging through the surface, we can see that the projection of the contact line is not circular (Fig. 23). Therefore, an equal angle between points does not imply an equal length of the line element (Fig. 24). For each angle, we located the contact line and computed the component of the surface tension force as in the algorithm. A difference here was that we could not ignore the size of the contact line element, as we did above.

Since we know the nature of this simulation, we can make some assumptions we could not make in the generic case. We know that in order for the particles to approach each other, the contact line must have a tilt. If we know the local height of the contact line for a particular line element, then we know the local contact radius, and we assume the line element...
is $d\ell = a d\theta$, where $a$ is the local contact line radius and $d\theta$ is the angle swept out by the contact line element $d\ell$, since the tilt is small. Allowing the algorithm to compute the height of the contact line element from the information inside the chosen cube was too erratic. Based on Fig. 20, the exact position of the contact line is a little ambiguous; so we determined a better system was to assume the contact line height varied linearly from the inner region to the outer. We computed the slope by measuring the height directly fore and aft of the particle (in the $z = 100$ plane), locations $F$ and $A$ in Fig. 21. To complete the algorithm, we assumed the highest point on the contact line (the $F$ location), was the same height the particle would have on a flat interface without gravity, as in Section 3.3, since the effect of gravity was practically negligible in this case. We calculated this once when the particles were released, and assumed the tilt stayed the same throughout the simulation.

The relative approach velocity (mm/s) as a function of center-to-center separation for the simulation and experiment are compared in Fig. 25. In this plot, time is interpreted as advancing from right to left, as contact occurs at $l/R_p = 2$. The approach was very slow when the separation between particles was large, picked up speed, and finally slowed down again before contact. Even though the surface tension force increased as the particles neared contact (Fig. 26), their approach velocity decreased due to viscous lubrication effects. As plotted in Fig. 26, the surface tension force pulling the particles together combined with pressure and viscous stresses to form a resultant force that was much smaller than the separate components, with significant fluctuations relative to the average magnitude. The key is to compute the resultant surface tension force (red line) properly, as this drives the motion. The viscous forces are only the response to this motion. As the contact line was slightly higher in the inner region, there was a small fore-aft hydrostatic pressure imbalance that acted to oppose the approach of the particles. One further consequence of the asymmetrical surface tension force was that the particles rotated as they translated on the surface (Fig. 27). This does not appear to have a large impact in this case, but could be important for cases where particles have a heterogeneous wettability, i.e. Janus Particles.

Our results agree with the experiment qualitatively, but differ significantly quantitatively. The large spike in the simulation result (Fig. 25) occurred when the particle’s vertical position crossed a lattice node as it slightly rose with the interface, changing its discrete representation on the lattice; this demonstrates the sensitivity of the result to the particle–lattice configuration. It should be pointed out that our results match better quantitatively at the early stages of our simulation,
most likely because our calculation of the fore-aft contact line height difference was accurate at that time, and only became inaccurate as the particles came closer together and the fore-aft height difference changed. The inset of Fig. 25 replots the simulation result along with the result of our original unmodified algorithm where there is roughly a factor of two difference between them, and significantly more oscillation for the original algorithm.

It was determined that the height difference detected to obtain the tilt slope was less than one lattice unit (about 0.5). We then performed a test where we assumed there was no tilt, i.e. the fore and aft heights were the same and the weight of each contact line element was equal, and the modified algorithm approximately matched the results of the original algorithm, with less oscillation. This means that the difference between obtaining the correct result and being off by nearly an order of magnitude depends on the ability to detect a contact line fore-aft height difference of about one lattice unit when $R_p = 50$; this is probably too much to expect for any algorithm solved on a uniform mesh. Although our approach velocity was significantly faster than in the experiment, it was still an incredibly slow process. As a frame of reference, it took the particle, of diameter 100 lattice units, about 250,000 lattice time steps to advance 1 lattice unit in the lateral direction. This is a staggeringly slow pace, where it is conceivable that any force fluctuations could cause a significant problem.

As mentioned at the beginning of this section, this was not an ideal case to validate the transient dynamics of our algorithm, but it is currently the only one. Unfortunately, since the result changed a great deal after a small change in the parameters of the algorithm, we doubt our numerical result would be easily reproducible for this experimental comparison. It can be concluded that the vertical component of the force was calculated correctly by the original algorithm, and we assume that the lateral components could be just as accurate under more favorable numerical conditions.

Fig. 26. The surface tension, viscous, and pressure forces on particle 1 are plotted as a function of time. The surface tension force comes from the algorithm detailed in the text while the viscous and pressure forces arise from the ME algorithm. These forces combined to produce a resultant whose magnitude is much smaller than either force. The fluctuations of this resultant force are large compared to their average.

Fig. 27. The rotational velocity of particle 1 is plotted as a function of time. The particles developed a z-direction rotation due to the fore-aft asymmetry of the surface tension force, which was not reported by Dalbe et al. [56].
Two aspects that would improve the accuracy of lateral movement along the interface would be: 1) To extend the discretization by having the contact line loop over an “equator” of the particle instead of such a small contact radius. This would be achieved by using a contact angle closer to 90° and would reduce severe spikes as a result of configuration changes. 2) To increase the tilt of the contact line. The fore-aft asymmetry of the contact line can be enhanced by using a particle with a larger density difference from the liquid. In other circumstances, a natural liquid deformation such as occurs during drop impact could provide such an asymmetry.

4. Conclusions

We developed an LBM algorithm to simulate fluid flows involving finite-sized particles in interfaces. Our algorithm combined previously developed algorithms to handle the multiple fluid components as well as the particle transport, where the multi-component algorithm suppressed spurious currents. It was found that the surface tension force associated with the interface acting on the particle was not captured by ME alone, most likely due to our implementation of boundary conditions, and a supplementary surface tension force algorithm was implemented.

We validated the incorporation of this new algorithm in three progressively rigorous steps. We demonstrated that the interfacial force on a fixed object was not captured by ME, but was captured by our algorithm. This test removed the complications of a two-way coupling between the fluid and a moving boundary, since we know the fluid behaves as expected in the presence of a fixed boundary for our algorithm. Therefore there was no question that the problem occurred in the force algorithm, as the motion algorithm was not invoked.

In our second test, we showed that a particle, free to move and immersed in an interface, will achieve its expected equilibrium depth penetrating into the interface. This important step showed that the fluid motion, force on a particle, and particle motion aspects of the code act in concert to achieve the correct equilibrium state. However, it says nothing about the approach to equilibrium. This conclusion is analogous to the dynamic contact angle aspect of our free energy-based algorithm. We know that a drop on a flat substrate will progress towards and eventually reach its equilibrium contact angle, but we do not necessarily know if the dynamic contact angle during the transient period is what would occur experimentally.

In our final test, we compared the dynamic results of a simulation to an experiment. The experiment tracked the approach velocity of two particles being drawn together by capillary effects on a flat interface. We found from our simulation that the expected equilibrium state was reached (particles contact), and that our approach velocity matched the result of the experiment qualitatively, but not quantitatively (our approach was faster).

The nature of the computational difficulties is not necessarily catastrophic, and could be overcome under circumstances where the resultant force on the particle was not so sensitive to lattice configuration effects. In this particular case, the particle barely protruded out of the interface. Therefore, the contact line looped over this part of the particle (small contact radius) and was not resolved enough to compute the surface tension force accurately. Comparatively, viscous drag and pressure forces were computed using the entire surface of the particle, which was well-resolved. The results of the algorithm would certainly improve with increased resolution, especially using an adaptive mesh refinement algorithm.

Regardless of the availability of increased computational power, there are two physical aspects that could improve the resolution of the contact line: 1) if the contact angle were closer to 90°, it would tend to loop over an “equator” of the particle, thereby increasing the number of contact line elements involved in the discretization, and 2) if there was a larger disparity between the particle and fluid densities, there would be a larger asymmetry, or tilt, of the contact line for particles that interact with each other.

There are two competing effects that combine to increase the magnitude of the lateral component of the surface tension force as the particles approach. The local tangent to the interface increases its lateral component in the region between particles when they are close to fulfilling contact angle boundary conditions, while the interface remains relatively unchanged in the region between the particle and the wall. This increases the lateral component of force. On the other hand, the tilt of the contact line, another fulfillment of contact angle boundary conditions, determines the local contact line radius, and therefore the weight of each given element in the numerical integral. The contact line creeps up the particle in the region between the particles which gives less weight to local forces in this region. This acts to decrease the total surface tension force. However, the resultant lateral force produced by these two competing effects is one that increases as particles approach.

In future implementations of our code, the behavior of particles with regard to motion normal to the interface can be considered reliable. Any motion tangent to the interface can be regarded as qualitatively accurate, leading to the achievement of the expected equilibrium state. However, the accuracy of the dynamics tangent to the interface must be considered questionable until validated, for each physical situation.

Acknowledgements

Funding for this research was provided by the CCNY-UCHicago Partnership on the Dynamics of Heterogeneous and Particulate Materials at the City College of New York Levich Institute, an NSF PREM (Award Number DMR-0934206). This research was also supported, in part, by a grant of computer time from the City University of New York High Performance Computing Center under NSF Grants CNS-0855217, CNS-0958379, and ACI-1126113.


