

ULRR

Comparison of finite volume and lattice Boltzmann methods for multicomponent flow simulations

Item Type	Article
Authors	Shardt, Orest
Citation	Canadian Journal of Chemical Engineering;98 (1), pp. 44-53
Publisher	John Wiley & Sons, Inc.
Download date	2026-06-11 08:08:10
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/8506

Comparison of finite volume and lattice Boltzmann methods for multicomponent flow simulations

Orest Shardt*

Bernal Institute and School of Engineering, University of Limerick,
Castletroy, Limerick, V94 T9PX, Ireland

Abstract

In pseudopotential lattice Boltzmann (LB) models for simulating multicomponent flows, interaction forces between the components of a mixture lead to phase separation and interfacial tension. At the macroscopic scale, such LB models solve an advection-diffusion equation for each component and the Navier-Stokes equations for the fluid mixture. In this paper, the computational efficiency of the LB method is compared with a finite volume (FV) solver for the same macroscopic-scale equations for a binary system in a two dimensional domain. The FV implementation replicates the phase separation of the LB model. Differences in the interfacial tension are due to truncation of the Taylor series expansion of the LB interaction force in the FV version. While the computations required to update the domain for each timestep can be completed faster with the FV approach, a smaller timestep is required to achieve stability, which negates the improvement in processing speed. The FV implementation, however, allows independent variation of model parameters, which is not possible in LB. For example, the viscosity can be changed without affecting interfacial tension or the extent of phase separation. Furthermore, it is possible to obtain low interfacial tensions without suppressing phase separation with the FV formulation. The significance of changing the diffusion rate of components on the deformation of a droplet in shear is also demonstrated. For three-dimensional simulations, the finite volume approach is expected to be faster than LB and would benefit from the demonstrated flexibility in specifying model parameters.

KEYWORDS: computational fluid dynamics, finite volume, lattice Boltzmann method, multiphase flow

*orest.shardt@ul.ie

This article has been accepted for publication and undergone full peer review but has not been through the copyediting, typesetting, pagination and proofreading process which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1002/cjce.23634

1 INTRODUCTION

Over the last several decades, lattice Boltzmann methods (LBMs) have been popular for simulations of flows with multiple phases and components. Among other LBMs for simulating multiphase flows, the pseudopotential method^[1,2] has been applied broadly. This method has been used to study, for example, flows in porous media,^[3,4] droplet deformation and breakup,^[5,6] droplet formation in microfluidic devices,^[7] droplets in turbulence,^[8,9] and other systems.^[10] In the pseudopotential LBM for multicomponent mixtures, separation of the mixture into distinct phases (with interfacial tension between them) is a consequence of an interaction force model. This automatic phase separation and implicit capturing of interface shapes obviates the need for moving meshes to describe interfaces between droplets or bubbles and the surrounding fluid. Such diffuse interface models are therefore useful for simulations of multiphase flows where topological changes, ie breakup and coalescence, occur as in emulsion flows and bubbly flows.

Simulations of droplet coalescence require high resolutions to capture the evolution of the thin films between colliding droplets.^[11,12] Simulations of such phenomena demand algorithms that minimize the need for arithmetic operations and memory transfers. In general, LB methods for computational fluid dynamics (CFD) solve the mass and momentum conservation equations by describing the translation and collisions of fictitious particles that move within a lattice.^[13] The state of the fluid is described via the distribution of the amount of these particles (called populations) that move with each of a finite set of velocities. Macroscopic quantities, such as density and momentum, follow from moments of the populations. Pressure is obtained from the density through an equation of state.

A two-dimensional pseudopotential LBM simulation of a two-component system (such as an oil-water dispersion) with a standard lattice (D2Q9: two-dimensional with nine discrete velocities) requires computing the evolution of two sets of populations, which amounts to 18 variables per lattice node. Additional variables, namely the densities and velocity components would also need to be computed. In contrast, a finite volume method (FVM) for the same continuum-level equations would require computing only four variables: two densities and two velocity components. Given that LBM simulations are often limited by the memory bandwidth offered by computing hardware (rather than the speed of performing floating point operations),^[13] a finite volume solver for the same mass and momentum transport equations would be an appealing alternative to an LBM solver due to its lower memory requirements. In three dimensions, the difference is more significant: an LBM implementation requires a minimum of 38 variables per node (D3Q19 lattice) whereas a finite volume solver requires five.

This paper presents a finite volume solver for the same macro-scale transport equations that are effectively solved by the pseudopotential LB with two ideal components.^[1,2] The purpose is

to assess whether the LB or FV approach is more efficient at solving these equations. The motivation is thus similar in spirit to that of Scarbolo et al,^[14] who derived a phase field equivalent of the pseudopotential model and compared a pseudo-spectral implementation of the resulting phase field model with pseudopotential and free-energy LBM. The authors concluded that the pseudo-spectral method was more accurate than LBM but more computationally expensive. To the author's knowledge, a finite volume implementation of the macroscopic equations of the pseudopotential LBM has not been previously reported.

After describing the numerical methods (LB and FV), this paper presents a comparison of the phase composition and interfacial tension in the FV implementation and LBM. Due to limitations in the pseudopotential LBM, several model parameters cannot be varied independently; these parameters may be freely specified in the FV implementation. The next section of the paper shows the ability to vary the viscosity (without affecting phase separation) in the FV implementation, decouple phase separation and interfacial tension, as well as adjusting the timescale of component diffusion. The last section compares the computational speed of the FV and LBM implementations.

2 NUMERICAL METHODS

Since the purpose of the paper is a comparison of equivalent (at the macroscopic level) FV and LBM codes, an introduction to LBM and the pseudopotential model for multicomponent systems is needed. In the pseudopotential LBM for multiple ideal components, each component (denoted by σ) is described by a distribution of populations f_i^σ . These populations represent amounts of fictitious molecules, and they evolve according to

$$f_i^\sigma(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) = \left(1 - \frac{1}{\tau_\sigma}\right) f_i^\sigma(\vec{x}, t) + \frac{1}{\tau_\sigma} f_i^{\sigma,eq}(\vec{x}, t), \quad (1)$$

which describes the movement of the populations between the adjacent nodes of a regular lattice. Collisions are modelled through relaxation towards an equilibrium distribution $f_i^{\sigma,eq}$. The parameter τ_σ specifies the rate of relaxation to the equilibrium distribution and as a result the viscosity of the fluid. As will be discussed later, this parameter also affects the diffusion of the components and therefore the rate of phase separation. The vectors \vec{c}_i are the discrete lattice speeds. This work considers the standard two dimensional lattice with nine directions (D2Q9). Attention is restricted to a system with two components $\sigma \in \{A, B\}$, and both components have equal relaxation times, ie $\tau^\sigma \equiv \tau$.

The equilibrium distribution for each component is^[13]

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

where the weights w_i are $w_0 = 4/9$, $w_{1-4} = 1/9$, and $w_{5-8} = 1/36$; and $c_s = 1/\sqrt{3}$ is the (isothermal) speed of sound. The density used in the equilibrium distribution is each component's density, and the velocity is a common fluid velocity with a shift to include the effect of the interaction force. This equilibrium velocity is

$$\vec{u}^{\sigma,eq} = \vec{u}' + \frac{\tau^\sigma \vec{F}^\sigma}{\rho^\sigma}, \quad (3)$$

where \vec{u}' is the common velocity

$$\vec{u}' = \frac{\rho^A \vec{u}^A + \rho^B \vec{u}^B}{\rho^A + \rho^B} \quad (4)$$

for the case of equal relaxation times for both components. The component densities ρ^σ and velocities \vec{u}^σ are calculated as

$$\rho^\sigma = \sum_i f_i^\sigma, \quad \rho^\sigma \vec{u}^\sigma = \sum_i f_i^\sigma \vec{c}_i. \quad (5)$$

While other methods for including forces in LBM can be considered,^[13,15,16] we use this form as an example. The choice of forcing scheme affects the continuum-level equations that the LBM solves.

To model the intermolecular forces that cause phase separation, each component σ experiences a force \vec{F}^σ due to the presence of the other component $\bar{\sigma}$. This force is calculated as

$$\vec{F}^\sigma = -G_{\sigma\bar{\sigma}} \rho^\sigma \sum_i w_i \rho^{\bar{\sigma}}(\vec{x} + \vec{c}_i \Delta t, t) \vec{c}_i \Delta t, \quad (6)$$

where the parameter $G_{\sigma\bar{\sigma}}$ specifies the magnitude of the repulsive interaction ($G < 0$ implies attraction). Since interactions are symmetric, only one parameter is needed for a two component system, and we define $G \equiv G_{AB} = G_{BA}$. The weights in Equation (6) are the same as those in the equilibrium distribution, although this is not required.

For a single component, LBM solves continuity and momentum conservation (Navier-Stokes) equations for weakly compressible flows.^[13] The kinematic viscosity of the fluid is given by $\nu = c_s^2(\tau - \Delta t/2)$, and its pressure follows from an ideal equation of state $p = c_s^2 \rho$. In the two component system, the interaction force influences the evolution of the component densities, causing phase separation. In the long wavelength limit, the preceding system of equations effectively solves mass and momentum conservation equations as approximated by^[2,14]:

$$\frac{\partial \rho^\sigma}{\partial t} + \nabla \cdot (\rho^\sigma \vec{u}) = \nabla \cdot \vec{J}^\sigma \quad (7)$$

$$\frac{\partial (\rho^\sigma \vec{u})}{\partial t} + \nabla \cdot (\rho^\sigma \vec{u} \vec{u}) = -\nabla p^\sigma + \nabla \cdot \left(\rho^\sigma \nu^\sigma (\nabla \vec{u} + \nabla \vec{u}^\top) \right) + \vec{F}^\sigma, \quad (8)$$

where \vec{u} is the common fluid velocity. The pressure and body force are often combined to form a modified pressure tensor. Here, they are left separate to reflect the implementation of the FV code. The component flux \vec{J} is

$$\vec{J}^A = -\vec{J}^B = D \left(x^B \nabla \rho^A - x^A \nabla \rho^B \right) - \tau \left(x^B \vec{F}^A - x^A \vec{F}^B \right), \quad (9)$$

where x^σ denotes the local mass fraction of component σ , eg $x^A = \rho^A / (\rho^A + \rho^B)$. The diffusion coefficient $D = c_s^2 (\tau - \Delta t / 2) = \nu$ in the LBM scheme, however, it is allowed to vary (and differ from the kinematic viscosity) for generality in the FV implementation. Similarly, τ may be varied freely in the FV implementation without affecting the fluid viscosity. In the FV implementation, τ is only a parameter in Equation (9) and has no effect on D or ν . In some simulations that are presented, the values of D and ν match those of the LB model and are therefore determined by the value of τ . Adding the equations for the two components, we obtain

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \quad (10)$$

$$\frac{\partial (\rho \vec{u})}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) = -\nabla p + \nabla \cdot (\rho \nu (\nabla \vec{u} + \nabla \vec{u}^T)) + \vec{F}, \quad (11)$$

where $\rho = \rho^A + \rho^B$; $p = p^A + p^B$; $\nu = (\rho^A \nu^A + \rho^B \nu^B) / \rho$; and $\vec{F} = \vec{F}^A + \vec{F}^B$. Taylor expansion of the intercomponent force, Equation (6), yields^[17]:

$$\vec{F}^\sigma = -G \rho^\sigma \left(c_s^2 \Delta t^2 \nabla \rho^\sigma + \frac{1}{2} c_s^4 \Delta t^4 \nabla (\nabla^2 \rho^\sigma) + \text{h.o.t.} \right). \quad (12)$$

Truncating the higher order terms and adding a parameter to allow the magnitudes of the two terms to be specified independently yields the form of the force term used in the FV implementation:

$$\vec{F}^\sigma = -\rho^\sigma (G_1 \nabla \rho^\sigma + G_2 \nabla (\nabla^2 \rho^\sigma)). \quad (13)$$

It follows that the total force is:

$$\vec{F} = -G_1 \left(\rho^A \nabla \rho^B + \rho^B \nabla \rho^A \right) - G_2 \left(\rho^A \nabla (\nabla^2 \rho^B) + \rho^B \nabla (\nabla^2 \rho^A) \right). \quad (14)$$

Since all parameters may be varied independently in the FV implementation, grid refinement studies are possible, in contrast to LBM where interfacial tension is an artifact of the discretization and several parameters are tied to the value of τ .

The finite volume code solves one mass conservation equation for each component (Equation (7)) and one momentum equation (Equation (11)) for the (common) fluid velocity. A fully-staggered uniform grid is used, with velocity control volumes centred on the faces of density

control volumes. This choice prevents checkerboarding in the solution, which was found with an implementation using a collocated grid. The cell size is taken to be unity, matching the lattice spacing of LBM. Interaction forces are evaluated at the centre of each velocity control volume. Convective and diffusive fluxes are evaluated using the central difference method. Where required, densities and velocity components are interpolated linearly from the values in adjacent control volumes. Integration over time is implemented with the explicit Euler method. The Laplacian of the density is evaluated at the centre of each density control volume using the nine point Mehrstellen stencil, which has isotropic discretization error.^[18] The gradient of the Laplacian over a velocity control volume (in Equation (14)) is computed by subtraction of the values of the Laplacian at the adjacent density control volumes.

The steps of the FV implementation are:

1. Initialize velocity and density fields.
2. Compute Laplacian of both density fields.
3. Compute interaction forces on both components.
4. Compute mass and momentum fluxes. In the staggered grid, the centres of the velocity control volumes (where the body force is applied) coincide with the edges of the density control volumes (where the force enters the flux calculation, Equation (9)), and therefore interpolation of the force components is not needed.
5. Update densities and fluid momentum.
6. Update fluid velocity from new momentum and interpolated densities.
7. Return to step 2.

This FV method is explicit, involving no iterations to solve the momentum or continuity equations. Like LB, it is also trivially parallel and requires data only from adjacent cells to update each cell.

The simulation domains are either fully periodic or, in the case of simple shear, periodic only in the shear direction (x) and have no-slip walls with specified velocities at both ends of the domain in the velocity gradient (y) direction. The initial conditions are zero velocity, and the density distributions of the two components depend on the case being simulated. In the simulations with sheared walls, a neutral wetting boundary condition is implemented by assuming zero normal gradients for both component densities when evaluating the Laplacians at a position adjacent to a wall. Other wetting boundary conditions, as used in the LBM literature, could be readily implemented.

3 RESULTS AND DISCUSSION

3.1 Phase composition and interfacial tension

We first assess the extent to which the FV implementation reproduces the behaviour of the LB model in two ways. The first comparison involves the compositions of the co-existing phases due to the repulsive interaction between the two components. For this test, the FV and LB computations were initialized with random 1% variations in the density of both components around an average density ρ_0 . The domains were 32×16 and fully periodic. In the LBM simulations, $\tau = 1$ was used, and the FV simulation parameters were chosen to match the LB model ($c_s^2 = 1/3$, $D = \nu = 1/6$, $G_1 = c_s^2 G$, $G_2 = \frac{1}{2} c_s^4 G$). Two values of the average density ($\rho_0 = 0.5$ and 1) and a range of G values were used. For the FV calculations time increments of $\Delta t = 1/3$ were used to ensure stability, while $\Delta t = 1$ in LBM. The simulations were run until $t = 50000$, after which negligible changes in the density profiles were observed. For each case, random initializations were chosen such that only two flat interfaces (normal parallel to the longer dimension of the domain) were present at the end of each simulation.

Figure 1 presents the compositions of the two phases as a function of the interaction parameter G . For $\rho_0 G > \rho_0 G^* = 0.5$, the extent of phase separation increases with increasing G . The curves for both $\rho_0 = 0.5$ and 1 are indistinguishable when plotting ρ/ρ_0 as a function of $\rho_0 G$. The densities computed using FV are in good agreement with those computed by LB. With increasing G , the extent of phase separation increases slightly faster for FV than LB. For $\rho_0 G > 0.85$, the finite volume calculations are unstable. By refining the grid and reducing the step size, it is possible to obtain stable results for higher G . The trends of the minimum and maximum densities continue, with the minimum density crossing zero. When negative minimum densities occur, the density profiles are not monotonic (they overshoot the bulk phase densities), and such cases are not considered further. LBM calculations are stable for $\rho_0 G \leq 1$.

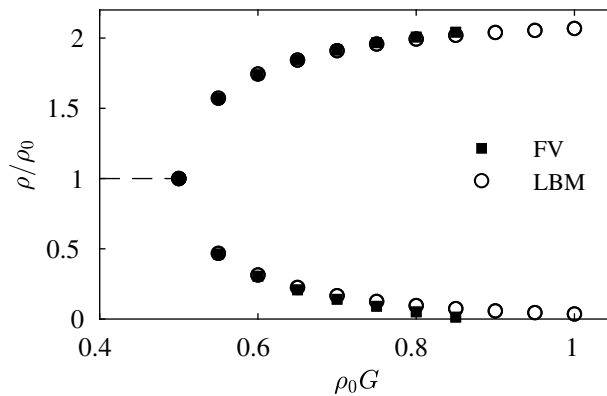


FIGURE 1: Compositions of the two co-existing phases as a function of the repulsive interaction strength in the FV and LBM implementations. The symbols for $\rho_0 = 0.5$ and $\rho_0 = 1$ coincide

The second comparison of the FV and LBM implementations involves the interfacial tension between the two fluid phases. For these simulations, the domain size was 64×64 , and circular droplets were initialized using the phase compositions determined from the previous flat interface simulations. These compositions (for $\tau = 1$ and $\rho_0 = 0.5$) were used to initialize the densities inside and outside of a droplet with radius $R = 20$. From such a simulation, the interfacial tension γ can be determined from the pressure difference $\Delta p = \gamma/R$ across the interface of a stationary droplet. Writing the body force as the divergence of a tensor,^[13,19] the pressure is

$$p = c_s^2(\rho^A + \rho^B) + G_1 \rho^A \rho^B \quad (15)$$

sufficiently far from any interfaces that terms involving density gradients vanish. Furthermore, from the difference in the normal and tangential components of the stress across a planar interface, the interfacial tension is

$$\gamma = -2G_2 \int \left(\frac{\partial \rho^A}{\partial x} \right) \left(\frac{\partial \rho^B}{\partial x} \right) dx. \quad (16)$$

The interfacial tensions determined by integration of the previously-obtained flat interfaces were compared with those computed with the Young-Laplace equation applied to 2D simulations of droplets. The interfacial tensions for the FV and LBM simulations are summarized in Figure 2. The units of the interfacial tension are lattice units with lattice spacing (cell size in FV), timestep, and reference mass all taken to be unity. All 2D simulations were run to $t = 50000$, long enough for the density profiles inside and outside the droplets to be steady.

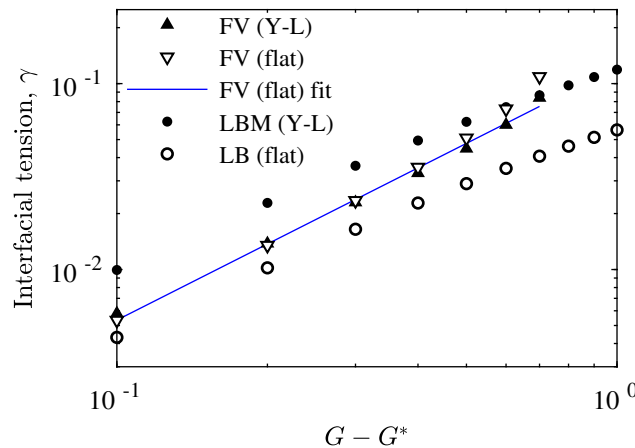


FIGURE 2: Dependence of the interfacial tension on the interaction strength between the two components. Symbols are shown for both the Young-Laplace method (Y-L) and integration of the stress difference across a flat interface (flat), Equation (16). The solid line shows a fit of the FV flat interface tensions, Equation (17)

The interfacial tension in the FV calculations, as determined by integration of the density

profiles of flat interfaces, is well-described by the empirical power law:

$$\frac{\gamma}{\rho_0} = 0.63[(G - G^*)\rho_0]^{1.4}. \quad (17)$$

With increasing G , the discrepancy between the interfacial tensions calculated from the droplet simulations and flat interfaces increases. The flat interface tension also deviates from the $(G - G^*)^{1.4}$ scaling for large G . Both effects are attributed to the decrease in interface thickness with increasing G , which causes density profiles in the interfacial region to be resolved poorly.

In contrast to FV, the interfacial tension in the LBM calculations increases with a smaller exponent (fitted value ≈ 1.2), in agreement with previous reports of a linear trend.^[20] Most notably, the interfacial tensions computed with the flat interface and Young-Laplace methods disagree by a factor of about two throughout the range of G values. The reason is the omission of τ -dependent terms in the macro-scale pressure tensor and Equation (16).^[21] Lower values of τ (approaching $1/2$) lead to closer agreement between the two calculations. With the Guo^[15] force scheme for LBM, interfacial tensions computed with the Young-Laplace equation are consistent with Equation (16). Use of this force scheme, however, also changes the composition of the phases. We do not consider here in further detail the reasons for the differences in behaviour of LB forcing schemes that are due to differences in higher order terms that affect the mass and momentum equations.^[13,22,23] Higher order terms, which were omitted in the FV implementation, are also the reason for the difference in interfacial tension between the LBM and FV results.

For both LBM and FV, the relationship between pressure difference and droplet radius is linear. The differences in interfacial tension between droplets with radii of 10 and 20 were 1.5% for FV and 0.6% for LBM, both with $G = 1.2$.

3.2 Parameter variation

The FV implementation allows greater flexibility in the specification of the model parameters. For example in LBM, the relaxation time τ determines both the kinematic viscosity of the fluid and the component flux (Equation (9)). Furthermore, the form of the interaction force is specified by a single parameter G in LBM, whereas one is free to use arbitrary factors in FV (Equation (13)). Modified LBM schemes that include multiple layers of adjacent lattice nodes have been developed to allow more flexibility in the context of non-ideal single component systems.^[24–26] This section demonstrates some consequences of the ability to freely vary parameters in the FV implementation.

3.2.1 Viscosity

One advantage of the FV implementation is the ability to vary the viscosity without affecting the diffusion of the components (and consequently the composition of the phases and the interfacial tension). While other authors have noted the τ -dependency of the interfacial tension,^[21] a second issue is that τ also affects the dynamics of the interface (through Equation (7)).

To show the effects of varying the viscosity, the deformation of a droplet (initial radius $R = 10$) was computed in a 80×40 domain with walls moving in opposite directions at a speed u_w . The compositions of the phases inside and outside the droplet were as determined previously (Figure 1). Parameters were chosen to obtain a droplet Reynolds number $\text{Re} = \frac{\dot{\gamma}R^2}{\nu} = 1$ and several capillary numbers $\text{Ca} = \frac{\rho\nu\dot{\gamma}R}{\gamma}$, where $\rho = \rho^A + \rho^B \approx 1$ is the outer phase density, $\dot{\gamma} = 2u_w/H$ is the shear rate, and $H = 40$ is the domain height (in the velocity gradient direction). To obtain capillary numbers of 0.2, 0.4, and 0.6, the viscosities were 0.167, 0.237, and 0.29, and the wall speeds were 0.0335, 0.0473, and 0.0580, respectively. For all cases, the other parameters were $D = 1/6$, $\tau = 1$, $G = 1.2$ (for which $\gamma = 0.014$), and $\Delta t = 1/3$.

The simulations were run to $t = 30000$, sufficient to obtain steady droplet shapes. Figure 3 shows that all three droplet shapes pass through two points, consistent with previous results.^[27,28] The extent of deformation qualitatively matches the earlier free-energy LBM calculations,^[28] though showing slightly less deformation at the highest Ca.

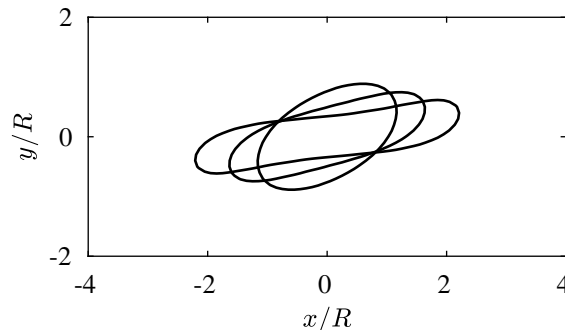


FIGURE 3: Steady droplet shapes at $\text{Re} = 1$ and $\text{Ca} = 0.2, 0.4, 0.6$ (in order of increasing deformation). Contours are computed as the positions where $\rho^A = \rho^B$

3.2.2 Interfacial tension

Due to the coupling of phase separation and interfacial tension, the LB model does not allow simulations with arbitrarily low interfacial tensions. With decreasing G , the interfacial tension decreases, and the interface thickness increases. To avoid impractically thick interfaces, one must keep the value of G (and therefore γ) above a minimum threshold. With the minimum interfacial tension thus fixed, high capillary numbers must be achieved by using a high viscosity (τ). In the FV implementation, one may independently control how the forcing terms affect the

mass and momentum equations.

Considering Equations (9) and (13), one could compensate for a reduction in G_1 and G_2 by increasing τ . For example, with $G = 0.1$ and $\tau = 11$ (together with $\rho_0 = 0.5$, $D = 1/6$), the component densities in both phases are 0.242 and 0.759 and the interfacial tension is 0.00052 from the Young-Laplace test as described earlier. Comparing with $G = 1.1$ and $\tau = 1$, the phase compositions are similar (compare with 0.232 and 0.785) while the interfacial tension is 11 times lower. Simulations with $\tau = 11$ are used in the next section to reach capillary numbers where droplet breakup occurs.

3.2.3 Diffusion

The deformation and eventual breakup of a droplet in shear are sensitive to the diffusion rates of the components, a result that is not evident from pseudopotential LB simulations (where the parameters are specified by the choice of τ). As can be seen from Equation (9), multiplying D and τ by the same factor only alters the timescale over which the densities evolve. Figure 4 presents the evolution over time of the perimeter of a sheared droplet for all parameters held constant except D and τ , which are varied while keeping their ratio constant. The perimeter S is computed as the total length of all $\rho^A = \rho^B$ contours (including potentially multiple droplets) and normalized by the initial undeformed droplet circumference $2\pi R$. The droplet initially had a stair-stepped interface with the densities inside and outside specified as the coexisting phase compositions for a flat interface. In these simulations, the droplet radius R was 20 in a $16R \times 4R$ domain. The parameters in the flux were specified as $\tau = 11\beta$ and $D = \beta/6$, where β is a free parameter that sets the diffusion timescale. The other parameters were $RT = 1/3$, $\nu = 0.107$, $u_w = 0.0107$, $G = 0.1$, $G_1 = GRT$, and $G_2 = \frac{1}{2}G_1RT$. The simulations used $\Delta t = 1/3$. With these parameters, $\gamma = 0.00052$, and it follows that $Re = 1$ and $Ca = 1.1$.

The computed behaviour of the sheared droplet depends on the value of β as evidenced by the changes in the evolution of the perimeter shown in Figure 4. For the highest β (2), the droplet deforms the least and does not break. Decreasing the rate of diffusion (lower β) induces breakup (shown as a sudden decrease in perimeter) and increases the extent of deformation. The maximum perimeter occurs for $\beta \approx 0.05$. Further decreasing β decreases the extent of deformation and delays breakup. At the lowest β considered here (0.005), the droplet breaks long after reaching a maximum perimeter ($\dot{\gamma}t/(2Ca) \approx 15$). Though not shown, the number and relative sizes of the daughter droplets is also affected by β . Figure 5 shows the droplet shape for the simulation with $\beta = 0.02$.

The evolution of the perimeter obtained with $\beta = 0.02$ matches previous 2D calculations with a free-energy LBM fairly well.^[28] In this case, the maximum perimeter $S/(2\pi R)$ is four and breakup occurs at $\dot{\gamma}t/(2Ca) = 10$, showing slightly less deformation (compare with 4.3) and earlier breakup (13) than previously reported. The difference may be due to the interface

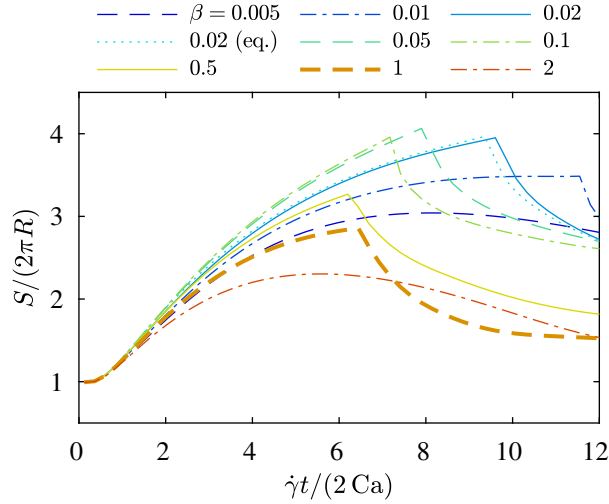


FIGURE 4: Effect of the diffusion rate (described by β) on the evolution of the normalized droplet perimeter for $Ca = 1.1$ and $Re = 1$. The case designated as (eq.) includes an initial equilibration time before application of shear

thickness, since a thicker interface in the free-energy method led to earlier breakup and less deformation.^[28] It is noteworthy that only the results with $\beta = 1$ can be obtained with the pseudopotential LBM scheme.

Figure 4 includes one additional case, labelled (eq.), in which a 100 000 time step delay was included to allow equilibration of the droplet interface before application of shear. In this case ($\beta = 0.02$), the droplet breaks slightly earlier than in the corresponding case without an equilibration delay, but the difference is small.

Further variations of the parameters in the FV model could be considered. In addition to the effect on deformation and breakup, the selection of D and τ are significant because they affect the rate of dissolution of small droplets and coarsening of simulated emulsions via Ostwald ripening. One would also be able to adjust the interface thickness and interfacial tension by changing G_2 and the relative magnitude of D with respect to τ . While the 2D computations presented here demonstrate the effect of β , further analysis of the effects of the various parameters – in addition to β – is deferred to future studies in 3D since droplet breakup is an inherently three-dimensional phenomenon.

3.3 Simulation speed

When domain sizes are small, as for the simulations reported so far, the memory requirements are modest and much of the domain can fit into the caches of a typical processor. In this case, memory access is fast and memory-intensive algorithms, such as LBM simulations, proceed quickly, limited by the speed of arithmetic operations. Larger domains, however, exceed the capacity of these caches that are between a processor and the main system RAM (random

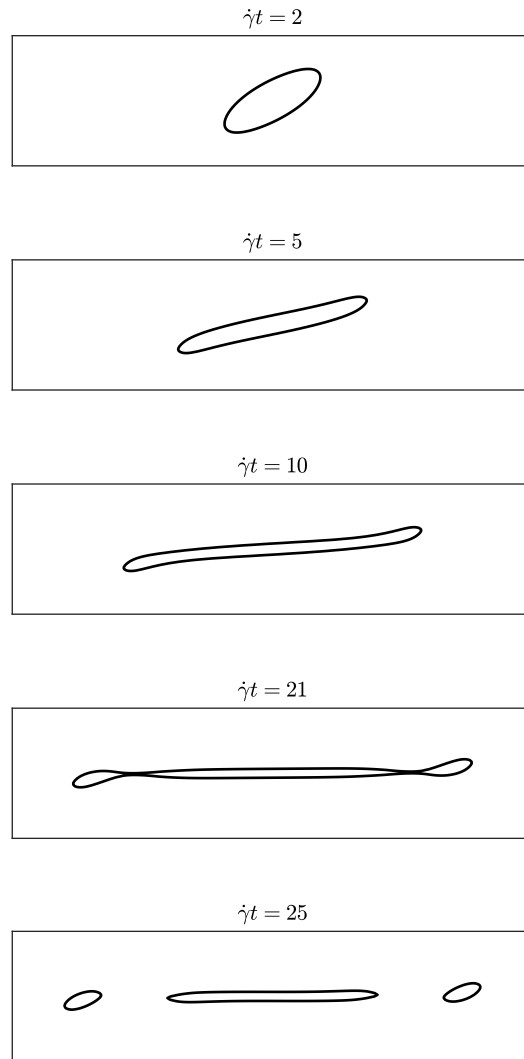


FIGURE 5: Deformation and breakup of a droplet computed with the FV implementation at $Ca = 1.1$, $Re = 1$, and $\beta = 0.02$. The contours show where $\rho^A = \rho^B$

access memory), and the available RAM bandwidth determines the speed of a simulation. This section considers the performance of the LB and FV codes in the limit of a large domain size, as would be relevant to practical applications of these models to simulate physical systems.

An open source code was used for the LB simulations,^[13] with the forcing scheme modified to implement the Shan-Chen rather than Guo method. A custom code was written for the FV implementation. All computations are performed using double precision, and both codes are serial. The C++ codes were compiled and run on a laptop with an Intel i5-6200U (Skylake, 2.3 GHz, 3 MiB cache) CPU and 8 GiB of DDR4-2133 RAM in one channel (maximum bandwidth 15.9 GiB/s). Both codes were compiled with the GNU C++ compiler with the optimization options `-Ofast` and `-march=native`. The first of these options allows optimizations that may alter the propagation of round-off error (in contrast to `-O3`, which selects high-level standards-compliant optimizations that cannot change error propagation). The second option enables instructions (such as fused multiply-add operations) based on the architecture of the compiling system rather than assuming a generic processor. The addition of the option `-march=native` provides a significant increase in speed ($1.7\times$ faster than only `-O3` for LB; $2.4\times$ for FV), while use of `-Ofast` rather than `-O3` provides a small increase in speed ($1.06\times$ faster than `-O3` with `-march=native` for LB; negligible difference for FV).

To test the speed of the codes for domain sizes that greatly exceed the capacity of the CPU's caches, simulations were run with a 3072×3072 domain. Initialization and data output were excluded from the timing. For this domain size, the speed of the LBM code was 7.4 million lattice node updates per second.

The FV implementation was 1.4 times faster, updating 10.6 million computational cells per second. The FV method, however, is unstable with $\Delta t = 1$. The size of the timesteps was therefore reduced until the computations were stable, which was found to be when $\Delta t = 0.67$ for simulations of phase separation with $G = 1.1$ and $G = 1.2$ (and all other parameters matching the LB model with $\tau = 1$). (The simulations in the preceding sections of the manuscript were run using a timestep of half this size, $\Delta t = 1/3$, which ensured that stability issues were not encountered.) The effective simulation speed of the FV code therefore decreases to 7.1 million cell updates per second, slightly (4%) slower than the LB scheme.

The FV code uses 16 arrays of double precision values for the simulation state (densities, momenta, Laplacians of the density fields, and force components), including temporary variables. In contrast, the LB code allocates 45 such arrays. For a 3072×3072 domain, the memory requirements are 1.1 vs. 3.2 GiB. The FV code is not faster than the LB implementation by the same factor as the ratio of the memory requirements (2.9) for several reasons. Table 1 presents the results of profiling both codes to determine the number of arithmetic operations performed and the amounts of data transferred between the processor and RAM.^[29,30] For the FV code, the values are for one timestep, which is smaller in size than the LB timestep by a factor of

	LB	FV
Total instructions executed	677	457
Double precision floating point operations	327	253
Bytes read	1403	576
Bytes written	391	112
Bytes transferred	1795	688
Computational intensity (operations/transfer)	1.5	2.9

TABLE 1: Performance characteristics of the two codes. The operation and memory transfer counts are per computational node (or cell) per timestep, and they are averaged over three runs. The computational intensity is the average number of double precision floating point operations per value (8 bytes each) transferred

0.67. The LB computations require 2.6 times the memory transfers of the FV approach. Furthermore, the FV approach requires 23% fewer floating point operations to update a cell than the LB method uses to update a node. Due to the force evaluation step in which two force components are computed from nine density values, significantly more data is read than written by both codes. Overall, the FV code performs twice as many floating point operations per byte of memory read or written. Neither code, however, could be considered dominantly limited by memory bandwidth or processing speed. Taking the theoretical speed of arithmetic operations as two per clock cycle (packed vector operations account for a negligible fraction of the floating point operations), the theoretical throughput is 5.6×10^9 operations per second. A memory bandwidth of 15.9 GiB/s corresponds to 2.1×10^9 double precision floating point values per second. Thus, 2.6 arithmetic operations take as much time as one transfer of a 64 bit floating point value. It follows that the theoretical ratios of compute time to memory transfer time are 1:2 (LB) and 1:1 (FV). Neglecting integer and other instructions, the theoretical speed of the FV code would be 1.9 times faster than the LB code; in practice it is 1.4 times faster.

At this stage, neither the LB nor FV codes were fully optimized to minimize memory use (total size and access rate). One option to reduce the total amount of memory required by the LBM code is to use an in-place streaming algorithm, in which the streaming step is completed in a way that allows data to be written to the same array as it is read from. This optimization, however, cannot reduce the amount of data that must be read and written and only changes where it is read from or written to. The most important optimization, relevant to both algorithms, would be to divide and update the domain by blocks so that any local temporary data used to update a node/cell need not be written and read from main RAM and could be retained in a CPU cache. For example, one step in both codes involves computing the intercomponent force fields in the whole domain. This requires reading the densities, writing the force values, then re-reading the density and force fields for subsequent calculations. Ideally, the densities and forces would stay in cache from the time they are first read and computed (respectively) to when they are used in later steps of the algorithm, thus eliminating the need for comparatively slow access to RAM.

Such an optimized implementation would bring the RAM requirements closer to the theoretical minimum of accessing RAM only for the state variables (component density and momentum in FV; populations in LB) in the whole domain. The reduction in memory access requirements could also shift the balance so that computation speed rather than memory speed determines the overall speed of a simulation.

In 2D, the increase in speed when using FV is balanced by a reduction in the timestep that is needed to achieve stability. While a 3D LB algorithm would require 19 velocities (D3Q19 lattice) instead of nine in 2D (D2Q9), the FV method requires only one more momentum component (and additional force and flux components, but these are temporary variables and also used in LB). In this case, it is likely that the reduction in memory requirements outweighs the necessary decrease in the timestep for stability, making FV faster overall.

4 CONCLUSIONS

With the aim of a direct comparison of lattice Boltzmann and finite volume approaches for multicomponent flows, this paper presented a finite volume solver for the same macro-scale equations as a popular multicomponent LBM for a binary mixture. The extent of phase separation as a function of the strength of the repulsion between components was consistent in the two methods. Interfacial tension was weaker in FV and increased faster with increasing repulsion between the components. This difference is attributed to truncation of the series expansion of the LBM interaction force as implemented in FV. The agreement between the LB and FV solvers implicitly demonstrates that the LB approach indeed solves the expected macro-scale equations, at least for the range of parameters considered.

In the FV formulation, the inclusion of a force term in the momentum equation does not affect the continuity equations of the components. This flexibility allows the FV implementation to overcome two limitations of the LBM approach. By adjusting the magnitudes of the terms involving the intercomponent force that appear in the continuity and momentum equations, one may reduce the interfacial tension between the components without affecting phase separation (interface thickness) and the fluid viscosity. Simulations of droplets in shear over a wider range of capillary numbers are therefore possible. Secondly, the timescale of component diffusion (analogous to mobility in free-energy models) may be specified independently of other parameters. This timescale was shown to affect the extent of droplet deformation in shear and the time of breakup, and it must therefore be chosen correctly to replicate the behaviour of physical systems.

Though the FV implementation updated the simulation domain 40% faster, the timestep size had to be reduced by at least 33% (relative to the LBM timestep for the same parameters) to ensure stability. The speeds of the LB and FV codes are therefore practically equivalent in

2D. For three-dimensional simulations, FV is expected to be faster due to the need for twice as many populations for LBM in 3D (D3Q19 lattice versus D2Q9). Furthermore, the total memory allocated by the FV code is about one third of the LB code's requirements for 2D domains. The lower memory requirements of FV make it appealing for use on memory-constrained architectures such as general purpose graphics processing units (GPUs), which offer high memory bandwidths and many cores for fast parallel computations.

The speed and flexibility of FV suggest that a parallel 3D implementation (for conventional CPUs or GPUs) is worthwhile. Even if stability constraints limit the effective speed of FV, the ability to independently vary model parameters motivates further development of FV models. For example, different forms of the component flux and intercomponent force could be considered with the goal of increasing simulation speed (by using an expression for the flux that is simpler to evaluate or improves stability) and introducing longer-range interactions by including higher order derivatives of the density through the use of wider stencils. The latter option has the potential to enable efficient simulations (and therefore larger physical domains or higher resolutions) of complex interfacially-dominated systems such as stable foams and emulsions (see eg LB models with multi-range interactions^[17,24,25]).

REFERENCES

- [1] X. Shan, H. Chen, *Phys. Rev. E* **1993**, *47*, 1815.
- [2] X. Shan, G. Doolen, *J. Stat. Phys.* **1995**, *81*, 379.
- [3] N. Martys, H. Chen, *Phys. Rev. E* **1996**, *53*, 743.
- [4] M. Sukop, H. Huang, C. Lin, M. Deo, K. Oh, J. Miller, *Phys. Rev. E* **2008**, *77*, 026710.
- [5] B. Sehgal, R. Nourgaliev, T. Dinh, *Prog. Nucl. Energ.* **1998**, *34*, 471.
- [6] H. Xi, C. Duncan, *Phys. Rev. E* **1999**, *59*, 3022.
- [7] W. Wang, Z. Liu, J. Jin, Y. Cheng, *Chem. Eng. J.* **2011**, *173*, 828.
- [8] P. Perlekar, L. Biferale, M. Sbragaglia, S. Srivastava, F. Toschi, *Phys. Fluids* **2012**, *24*, 065101.
- [9] S. Mukherjee, A. Safdari, O. Shardt, S. Kenjeres, H. van den Akker, *arXiv* **2019**, 1902.09929.
- [10] L. Chen, Q. Kang, Y. Mu, Y.-L. He, W.-Q. Tao, *Int. J. Heat Mass Tran.* **2014**, *76*, 210.
- [11] O. Shardt, J. Derksen, S. Mitra, *Langmuir* **2013**, *29*, 6201.
- [12] O. Shardt, S. Mitra, J. Derksen, *Langmuir* **2014**, *30*, 14416.
- [13] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E. Viggien, *The Lattice Boltzmann Method: Principles and Practice*, Springer International Publishing Switzerland **2017**.
- [14] L. Scarbolo, D. Molin, P. Perlekar, M. Sbragaglia, A. Soldati, F. Toschi, *J. Comput. Phys.* **2013**, *234*, 263.
- [15] Z. Guo, C. Zheng, B. Shi, *Phys. Rev. E* **2002**, *65*, 046308.
- [16] A. Kupershtokh, D. Medvedev, D. Karpov, *Comput. Math. Appl.* **2009**, *58*, 965.
- [17] M. Sbragaglia, R. Benzi, L. Biferale, S. Succi, K. Sugiyama, F. Toschi, *Phys. Rev. E.* **2007**, *75*, 026702.
- [18] M. Patra, M. Karttunen, *Numer. Meth. Part. D. E.* **2006**, *22*, 1.
- [19] M. Sbragaglia, D. Belardinelli, *Phys. Rev. E* **2013**, *88*, 013306.

- [20] Z. Yang, T. Dinh, R. Nourgaliev, B. Sehgal, *Int. J. Heat Mass Tran.* **2001**, *44*, 195.
- [21] R. Benzi, M. Sbragaglia, S. Succi, M. Bernaschi, S. Chibbaro, *J. Chem. Phys.* **2009**, *131*, 104903.
- [22] D. Lycett-Brown, K. Luo, *Phys. Rev. E* **2015**, *91*, 023305.
- [23] A. Zarghami, N. Looije, H. van den Akker, *Phys. Rev. E* **2015**, *92*, 023307.
- [24] G. Falcucci, G. Bella, G. Chiatti, S. Chibbaro, M. Sbragaglia, S. Succi, *Commun. Comput. Phys.* **2007**, *2*, 1071.
- [25] S. Chibbaro, G. Falcucci, G. Chiatti, H. Chen, X. Shan, S. Succi, *Phys. Rev. E* **2008**, *77*, 036705.
- [26] A. Kuzmin, A. Mohamad, *Comput. Math. Appl.* **2010**, *1902.09929*, 2260.
- [27] H. Zhou, C. Pozrikidis, *Phys. Fluids A-Fluid* **1993**, *5*, 311.
- [28] R. van der Sman, S. van der Graaf, *Comput. Phys. Commun.* **2008**, *178*, 492.
- [29] J. Treibig, G. Hager, G. Wellein, *arXiv* **2010**, *1004.4431v3*.
- [30] GitHub - RRZE-HPC/likwid: Performance monitoring and benchmarking suite, <https://github.com/RRZE-HPC/likwid> (accessed: June 2019).