Lattice Boltzmann III: Multiphase flows

November 24, 2015

In this lecture we shall present the main extensions of the LB method to include the effects of potential energy interactions, leading to non-ideal fluid behavior and ensuing phase-transitions relevant to the physics of multiphase flows.
LB for non-ideal fluids

Sauro Succi

LB multiphase: droplet formation
Non-ideal equation of state

\[ p = \rho c_s^2 + p^*(\rho) \]

**Excess pressure** \( p^* \)

As a result of potential energy \( \rightarrow \) force

\[ \mathbf{F} = \text{div} \mathbf{P}^{(nid)} \]

\[ f_i(r; t) = f_i(r; t) + \Delta f_i(r; t) \]

Forcing terms: coupling to the environment

\[ \frac{df}{dt} = \frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \mathbf{F}(\mathbf{x}, t) \cdot \frac{\partial f}{\partial \mathbf{v}} \]

\[ \mathbf{F}(x; t) = \begin{cases} 
mg, & qE \\
-q\nabla \phi; & \Delta \phi = q\rho \\
-\frac{\delta G(\rho)}{\delta \rho(r)}; & G(\rho) = \int [g(\rho(r)) + \frac{\gamma}{2} (\nabla \rho)^2] dr \\
\gamma(\mathbf{u} - \mathbf{v}) 
\end{cases} \]
Forcing terms: projection route

\[ F_i = \beta [F^1 \gamma + F^2 \zeta_i + F^3 (\gamma \zeta_i - \zeta_i^2 \delta_{ab})] \]

\[ \sum F_i = \frac{\partial p}{\partial t} + \rho = R \]

\[ \sum F_i \delta \tilde{v}_i = \frac{\partial \rho \tilde{u}}{\partial t} = \rho \tilde{u} \]

\[ \sum F_i \delta \tilde{F}_i = \frac{\partial \rho}{\partial t} \delta_{col} = \rho (\tilde{u} + \tilde{a})/2 \]

LB for non-ideal fluids: pseudopotentials

Ising-like: with moving spins

\[ \tilde{F}(x) = G \sum \tilde{c}_i \Psi[\rho(x)] \Psi[\rho(x + \tilde{c}_i)] \]

\( \Psi \): Free-energy

\( G \): Coupling strength

Interfaces emerge from microdynamics

(Gustensen-Rothman 1992, Shan-Chen, Swift, Yeomans, 1996, Care, Halliday, 2005)
Diagrammatic expansion

\[ <c> \equiv \sum_{i=0}^{b} w_i c_i \]

\[ \psi <c> \quad \psi = 0 \]

\[ \psi <cc>_{ab} \nabla_b \psi \propto \nabla^2 \frac{\psi^2}{2} \Rightarrow \rho^* = G \frac{\psi^2}{2} \]

\[ \psi <ccc>_{abc} \propto \nabla_b \nabla_c \psi = 0 \]

\[ \psi <cccc>_{abcd} \propto \nabla_a \nabla_b \nabla_c \psi \]

Isotropic tensors of order 4 needed!

Shan-Chen EoS
Shan-Chen pseudo-thermodynamics

PseudoP:
\[ \psi[\rho] = \sqrt{\rho_0} (1 - e^{-\rho/\rho_0}) \]

EoS:
\[ P = \rho c_s^2 + \frac{G c_s^2}{2} \Psi^2[\rho] \]

Surftens:
\[ \gamma \sim \rho G \int_{-\infty}^{+\infty} (\partial_y \psi)^2 \, dy \]
Shan-Chen EoS

![Shan-Chen EoS Diagram](image1)

Shan-Chen EoS

![Shan-Chen EoS Diagram](image2)
Calibrating the density ratio

The parameter $G$ (inverse temperature) determines the surface tension $\gamma$ and the density ratio between gas and liquid phase

$$\rho_c = \rho_0 \ln 2 \quad G_c = -4$$
Laplace Test

\[ p_{in} - p_{out} = \frac{\gamma}{R} \]

Hint: initialize with density close to critical ...

LB phase separation
LB phase: droplet formation

Phase-separation: droplets
LB multi-phase code

Basically add 20 lines of code to implement the nearest-neighbor Force $\Psi(x) \Psi(x \_c \_i)$. 

Nothing else!

LB Code: force

\[ f_i(\vec{r} + \vec{c}_i \Delta t; t + \Delta t) - f_i(\vec{r}, t) = -\omega \Delta t \left( f_i - f_i^{eq} \right) + F_i \Delta t \]

where \[ F_i = \frac{\vec{F} \cdot \vec{c}_i}{c_i^2} \]

\| \vec{F} \| = F_{inv} = G \xi \psi \nabla \psi

- we can account for the force in the collision routine, or
- (better) with a shifted velocity in the equilibrium computation

\[ \vec{u} \rightarrow \vec{u}' = \vec{u} + \frac{\vec{F} \xi}{\rho} \]
LB Code Structure

```
* -------------------------------------------------------------
* program lb2b                                             *
* -------------------------------------------------------------
* implicit double precision(a-h,0-9)                       *
  include 'mpaww.par'                                       *
* -------------------------------------------------------------
  c ------- input parameters                                  *
    common dfl                                              *
    call input                                              *
  c ------- initialization                                  *
    call lshydol                                            *
    call lshhyg                                            *
* -------------------------------------------------------------
  c ------- MAIN LOOP                                       *
    do 10 iang = 1,range                                     *
      call schskol                                            *
      call phil                                              *
      call mrs                                              *
      call lhydol                                            *
      call lhydol                                            *
      if (ifiony) then                                        *
        call error                                            *
        call error                                            *
        call error                                            *
      endif                                                  *
  10 continue                                              *
```

LB Code: input

```c
#include <stdio.h>

#define MAX_ELEMENT 100
#define MAX_STEP 100

int main

  // Initialize variables
  int element, step, x, y;
  double density, velocity;

  // Loop over elements and steps
  for (element = 0; element < MAX_ELEMENT; element++)
    for (step = 0; step < MAX_STEP; step++)
      // Update density and velocity
      density += update_density(density, velocity);
      velocity += update_velocity(density, velocity);

  // Output results
  printf("Density: %.2f, Velocity: %.2f\n", density, velocity);
```

LB Code: inthydro

```c
#include <stdio.h>

#define MAX_ELEMENT 100
#define MAX_STEP 100

int main

  // Initialize variables
  int element, step, x, y;
  double density, velocity;

  // Loop over elements and steps
  for (element = 0; element < MAX_ELEMENT; element++)
    for (step = 0; step < MAX_STEP; step++)
      // Update density and velocity
      density += update_density(density, velocity);
      velocity += update_velocity(density, velocity);

  // Output results
  printf("Density: %.2f, Velocity: %.2f\n", density, velocity);
```
LB Code: move

```
subroutine move
![Code snippet]
end
```

LB Code: hydrovar (1 & 2)

```
subroutine hydrovar
![Code snippet]
end
```

15
LB code: force

```plaintext
import numpy as np

# Initial conditions
x0 = np.zeros((3, N))
v0 = np.zeros((3, N))
F0 = np.zeros((3, N))

# Time step
dt = 0.01

# Courant number
C = 0.5

# Density
rho = 1

# Viscosity
nu = 0.01

# Density of air
rho_a = 1.225

# Velocity of sound
c = np.sqrt(rho_a / rho)

# Function to calculate force
def calculate_force(r, u):
    # Implementation of the force calculation
    pass

# Function to calculate velocity
def calculate_velocity(x, v, t):
    # Implementation of the velocity calculation
    pass

# Main loop
for t in range(100):
    # Calculate force
    F = calculate_force(x, u)
    # Calculate new velocity
    v = v - nu * np.gradient(F, x) / c
    # Calculate new position
    x = x + v * dt
```

LB code: equilibrium

```plaintext
import numpy as np

# Initial conditions
w0 = np.zeros((3, N))

# Time step
dt = 0.01

# Density
rho = 1

# Viscosity
nu = 0.01

# Density of air
rho_a = 1.225

# Velocity of sound
c = np.sqrt(rho_a / rho)

# Function to calculate force
def calculate_force(r, u):
    # Implementation of the force calculation
    pass

# Function to calculate velocity
def calculate_velocity(x, v, t):
    # Implementation of the velocity calculation
    pass

# Main loop
for t in range(100):
    # Calculate force
    F = calculate_force(x, u)
    # Calculate new velocity
    u = u - nu * np.gradient(F, x) / c
    # Calculate new position
    x = x + u * dt"
subroutine collision

! implicit double precision(a-h, o-x)
include 'muphasa.par'

! as j = 1, nj
! do i = 1, nx
! k = 0, npop-1
! f(k,l,i) = f(k,i,l) * (0.0460 - omega)
! omega = omega + f(k,l,i)
! end do
! end do
return
end

subroutine media

! implicit double precision(a-h, o-x)
include 'muphasa.par'

! do j=1,nj
! do i=1,nx
! w(i,j)=w(i,j)*0.560
! end do
! end do
return
end

Phase-separation: show movie
Moving droplets, jets: still a challenge

Shan-Chen, pros and cons

✓ Simple and elegant
✓ Efficient

Spurious currents (isotropy)
✓ EoS and surfens, same G
✓ Thick interface (w=5 dx)
Beyond Shan-Chen

Standard vs. 2-belt

### Standard Shan-Chen model

\[
\begin{align*}
\mathbf{S}(t) &= \mathbf{S}(t-1) + \mathbf{F}(t) \\
\mathbf{w}(t) &= 1 - \mathbf{w}(t-1) \\
\mathbf{w}(t) &= 1 - 36
\end{align*}
\]

\[
\begin{align*}
\sum_{i=1}^{3} w_i^2 &= \sum_{i=1}^{3} p_i + \sum_{i=1}^{3} p_i' = 1 \\
\sum_{i=1}^{3} w_i^2 &= \sum_{i=1}^{3} p_i p_i' = \sum_{i=1}^{3} p_i p_i' = \sum_{i=1}^{3} p_i p_i' = \sum_{i=1}^{3} p_i p_i'
\end{align*}
\]

### 2-belt model

\[
\begin{align*}
\mathbf{F}(t) &= \\
\mathbf{p}_i, \mathbf{p}_i' &= 4, 43, \\
\mathbf{p}_i, \mathbf{p}_i' &= 4, 43, \\
\mathbf{p}_i, \mathbf{p}_i' &= 4, 4, \\
\mathbf{p}_i, \mathbf{p}_i' &= 5, 5, 5, \\
\mathbf{p}_i, \mathbf{p}_i' &= 10, 5
\end{align*}
\]

### Higher order lattices

(Sragaglia, Benzi, Biferale, SS, PRE 2007)
**A/R competition: Foams/Emulsions**

![Image of foam and emulsion simulations at t=2000 and t=5000]

**Fluid-wall interactions**

\[ \vec{F}_{fw}(x) = G_{fw} \Psi(x) \sum_i \vec{c}_i \Psi(\rho(x + c_i)) \]

We assign at the points belonging to the wall a density \( \rho_w \).

Tuning \( \rho_w \) we are able to modify and control the contact angle \( \theta \).

![Graph showing the relationship between contact angle and \( \rho_w \)]
Assignments

*Insert the Shan-Chen forces in the 2d LB code and run the Laplace test (single bubble) at various coupling strengths G.*
End of the Lecture