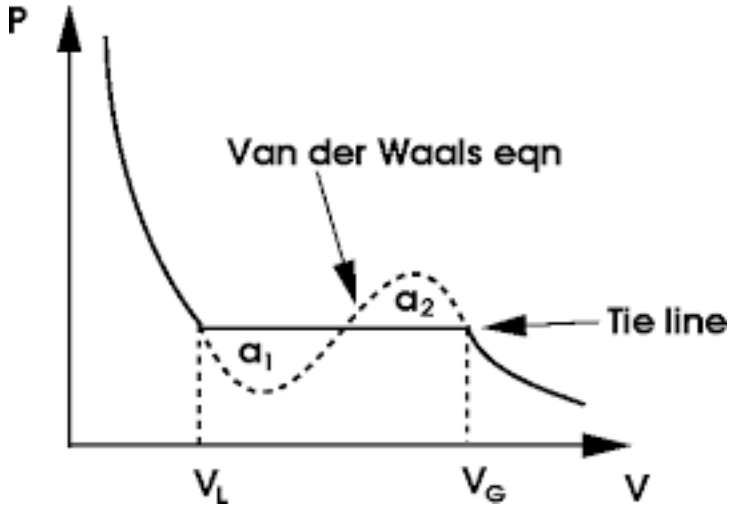
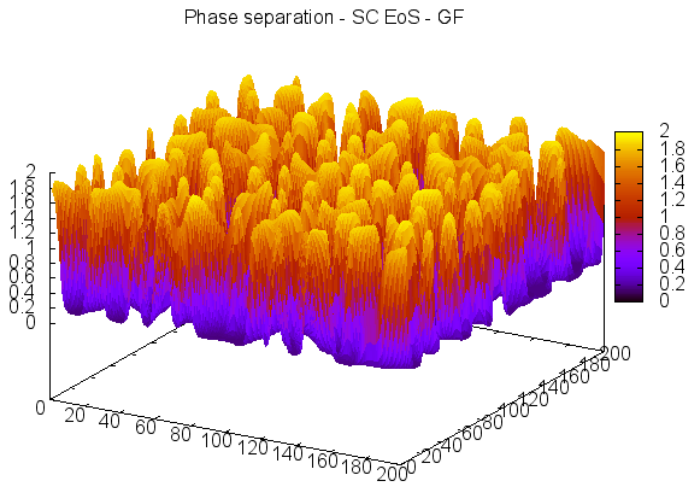
LB for non-ideal fluids

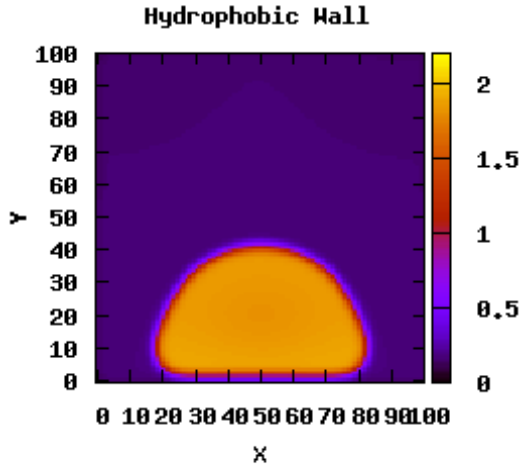
Goal: Include Potential Energy

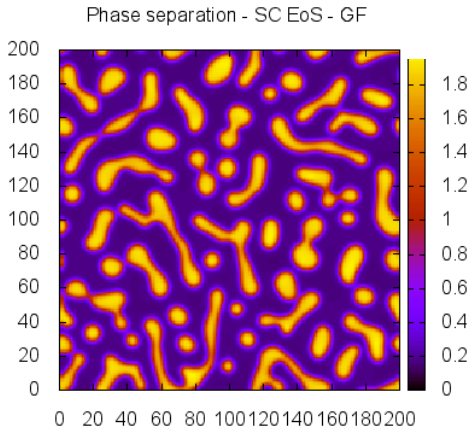
Non-Ideal EoS



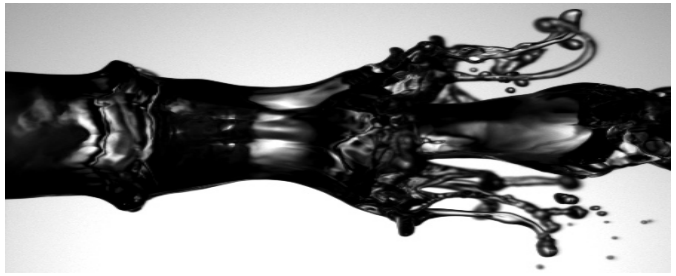
Multiphase Flows: Phase Transitions

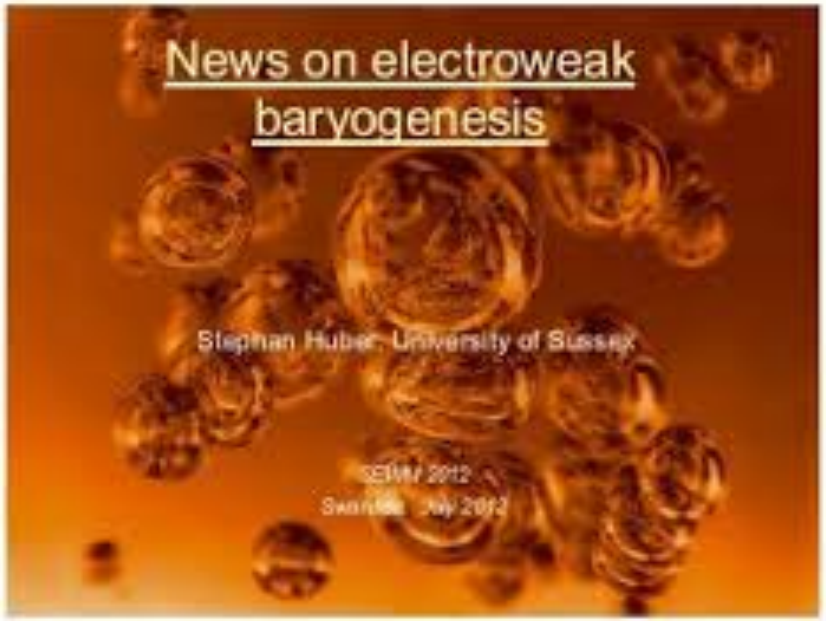






Applications





Non-ideal equation of state: potential energy

$$p = \rho c_s^2 + p^*(\rho)$$

Ideal pV=nRT +
Excess pressure p*

Short-range repulsion + long-range attraction

$$V_{LJ}(r) = 4\varepsilon * [(r_0 / r)^{12} - (r_0 / r)^6]$$

$$(p + a / V^2)(V - b) = nRT$$

Non-ideal kinetic theory

$$\frac{df}{dt} \equiv \frac{\partial f}{\partial t} + \vec{v} \bullet \frac{\partial f}{\partial \vec{r}} + \boxed{\frac{\vec{F}(\vec{x},t)}{m}} \bullet \frac{\partial f}{\partial \vec{v}}$$

Many types of forces:

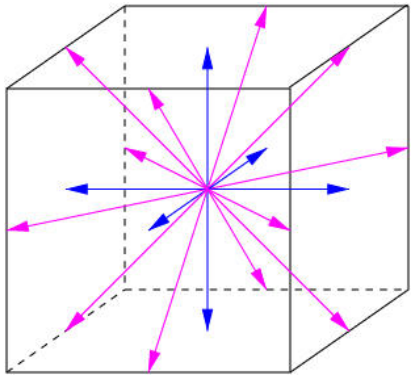
$$\vec{F}(x;t) = \begin{cases} m\vec{g}, \quad q\vec{E} \\ -q\nabla\phi; \ \Delta\phi = q\rho \\ -\dfrac{\delta G[\rho]}{\delta\rho(r)}; \quad G[\rho] = \int [g(\rho(r)) + \dfrac{\chi}{2}(\nabla\rho)^2]dr \\ \gamma(\vec{u} - \vec{v}) \end{cases}$$

Lattice non-ideal equation of state

$$p = \rho c_s^2 + p^*(\rho)$$

Ideal pV=nRT +
Excess pressure p*

As a result of potential energy → force



$$\vec{F} = div\,\vec{\vec{P}}^{(nid)}$$

$$f_i(r;t) = f_i(r;t) + \Delta f_i(r;t)$$

$$\Delta f_i(r;t) = F_i \Delta t$$

Forcing terms: projection route

$$F_i = w_i[F^0 1_i + F^1_a c_{ia} + F^2_{ab}(c_{ia}c_{ib} - c_s^2 \delta_{ab})]$$

$$S_v \equiv \frac{\vec{F}(\vec{x},t)}{m} \bullet \frac{\partial f}{\partial \vec{v}}$$

$$\vec{a} \equiv \frac{\vec{F}(\vec{x},t)}{m}$$

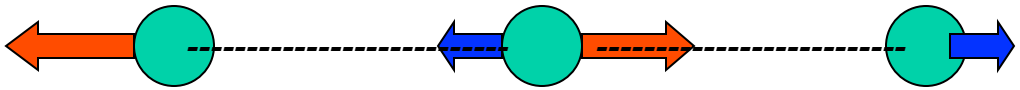
$$\sum_i F_i = \frac{\delta \rho}{\delta t}\Big|_{coll} = R$$

$$\sum_i F_i \vec{c}_i = \frac{\delta \rho \vec{u}}{\delta t}\Big|_{coll} = \rho \vec{a}$$

$$\sum_i F_i \vec{c}_i \vec{c}_i = \frac{\delta \vec{\vec{P}}}{\delta t}\Big|_{coll} = \rho(\vec{a}\vec{u} + \vec{u}\vec{a})/2$$

LB for non ideal fluids: Pseudo-potentials

Ising-like: with moving spins



$$\vec{F}(x) = G \sum_i \vec{c}_i \Psi[\rho(x)]\Psi[\rho(x+\vec{c}_i)]$$

Ψ : *Density functional*

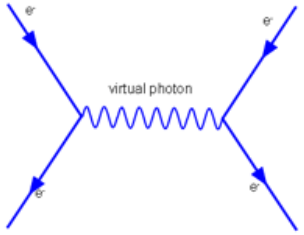
G : *Coupling strength*

Interfaces **emerge** from microdynamics

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

Diagrammatic expansion

$$< c >_a \equiv \sum_{i=0}^{b} w_i c_{ia}$$



$$\psi < c >_a \psi = 0$$

$$\psi < cc >_{ab} \nabla_b \psi \propto \nabla \frac{\psi^2}{2} \Rightarrow p^* = G \frac{\psi^2}{2}$$

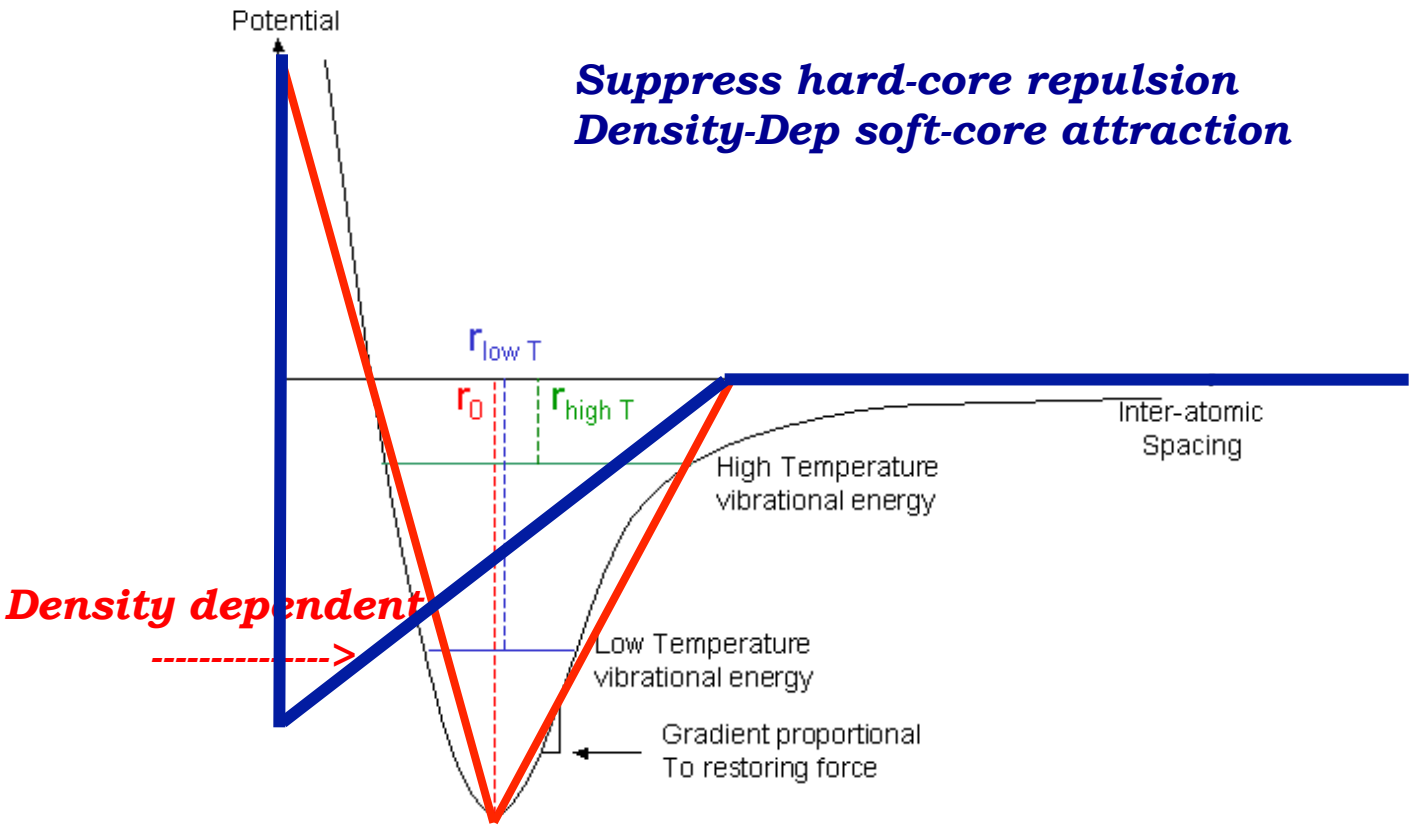
$$\psi < ccc >_{abc} \propto \nabla_b \nabla_c \psi = 0$$

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

$$\gamma \approx G \int (\nabla \psi)^2 dx$$

Isotropic tensors of order 4 needed!

Lattice potential: Shan-Chen vs Lennard-Jones



Shan-Chen pseudo-thermodynamics

PseudoP

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

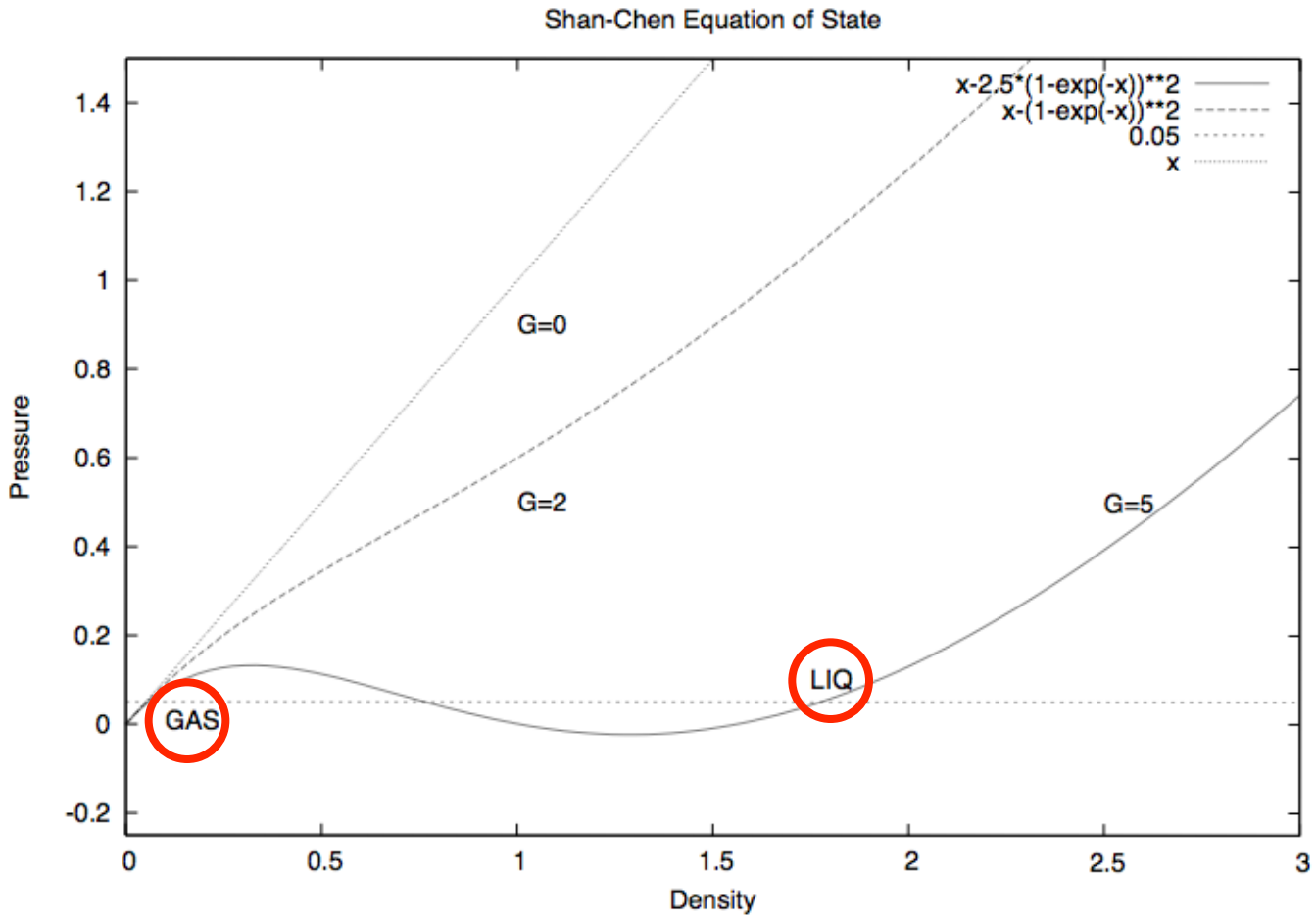
EoS:

$$P = \rho c_s^2 + \frac{G}{2} c_s^2 \, \Psi^2[\rho]$$

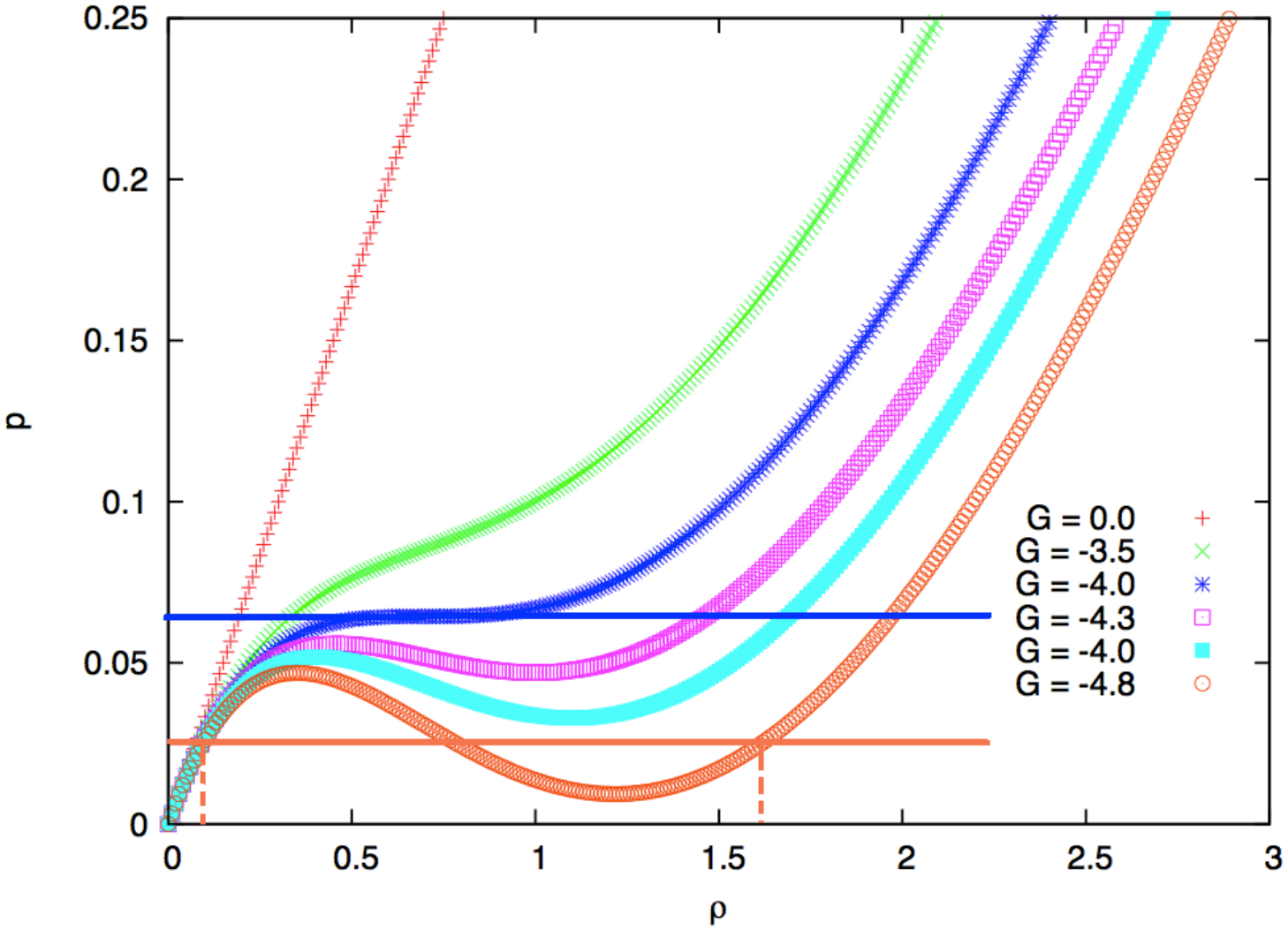
Surftens:

$$\gamma \sim \rho G \int_{-\infty}^{+\infty} (\partial_y \psi)^2 \, dy$$

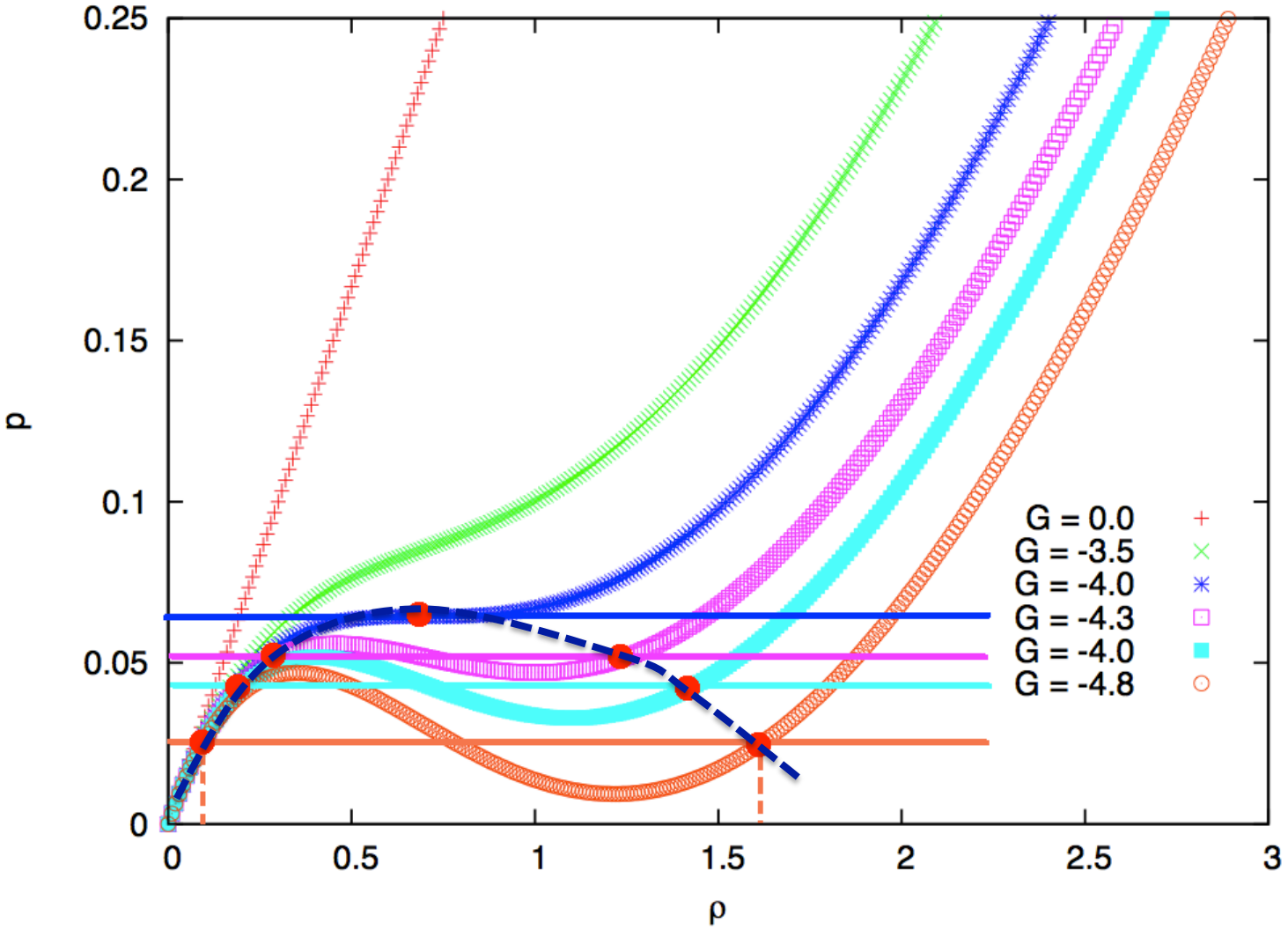
Shan-Chen EoS



Shan-Chen EoS



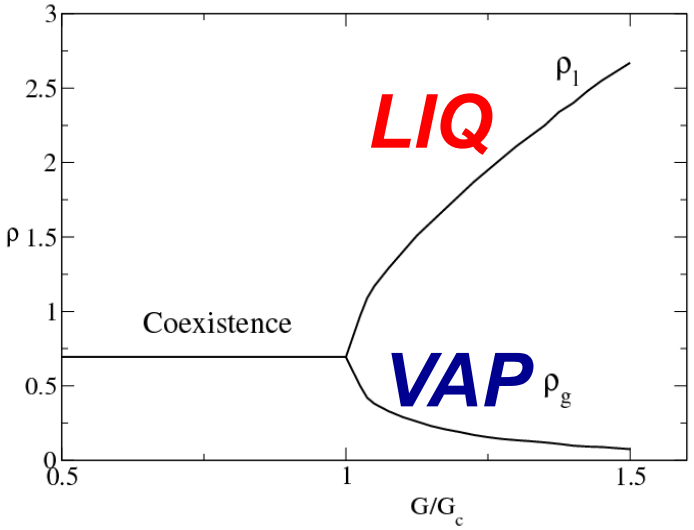
Shan-Chen EoS

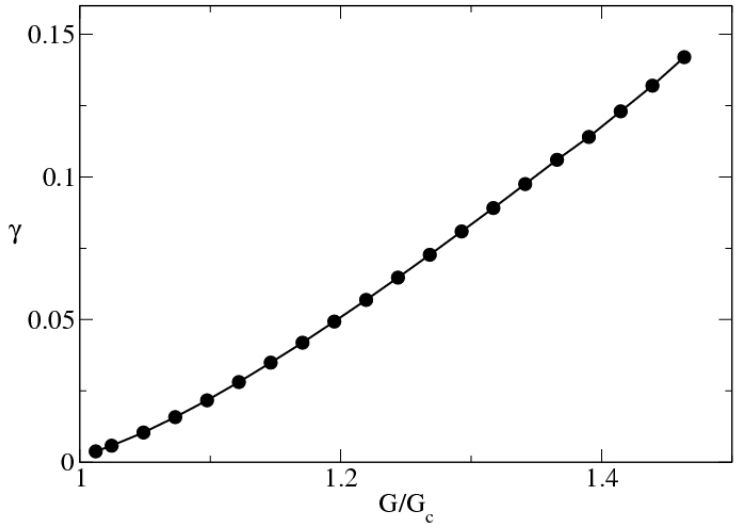


Calibrating the density ratio

The parameter G (*inverse temperature*) determines the surface tension γ and the density ratio between gas and liquid phase

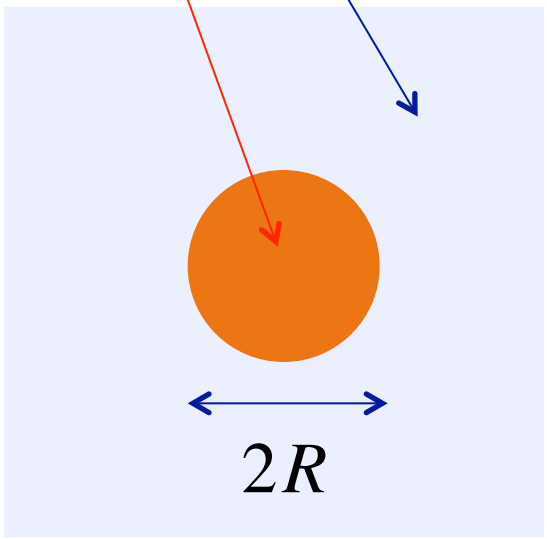
$$\rho_c = \rho_0 \ln 2 \qquad G_c = -4$$





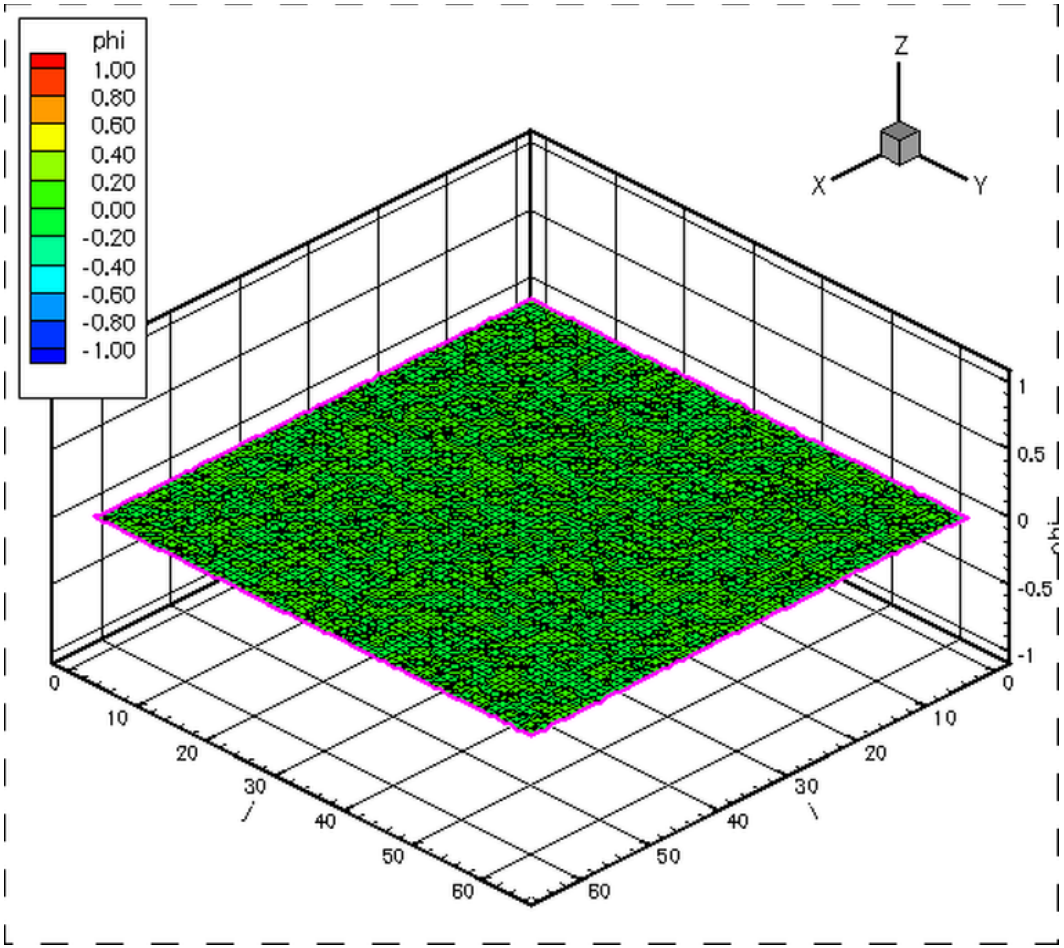
Laplace Test

$$p_{in} - p_{out} = \frac{\gamma}{R}$$

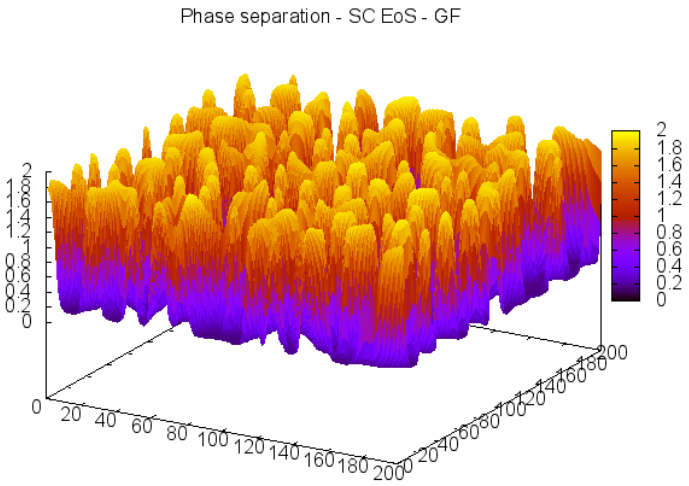


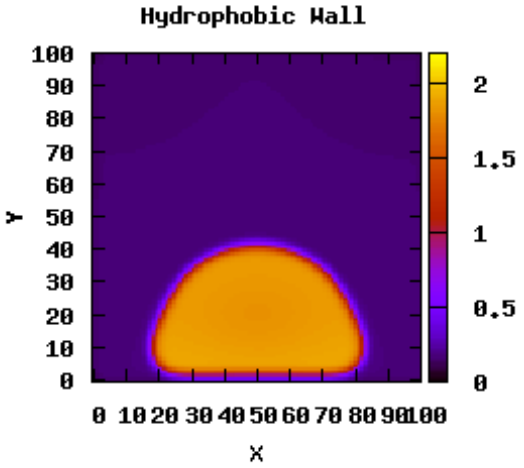
Hint: initialize with density close to critical ...

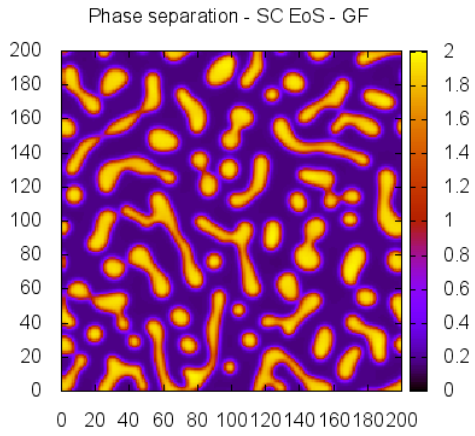
Phase-separation: show movie



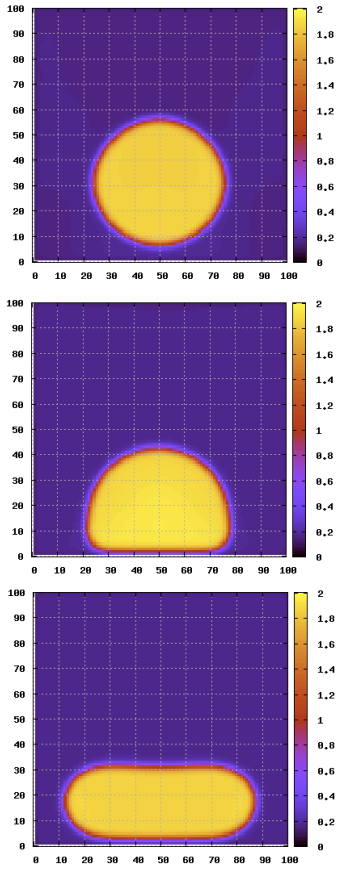
LB *muphase*: droplet formation

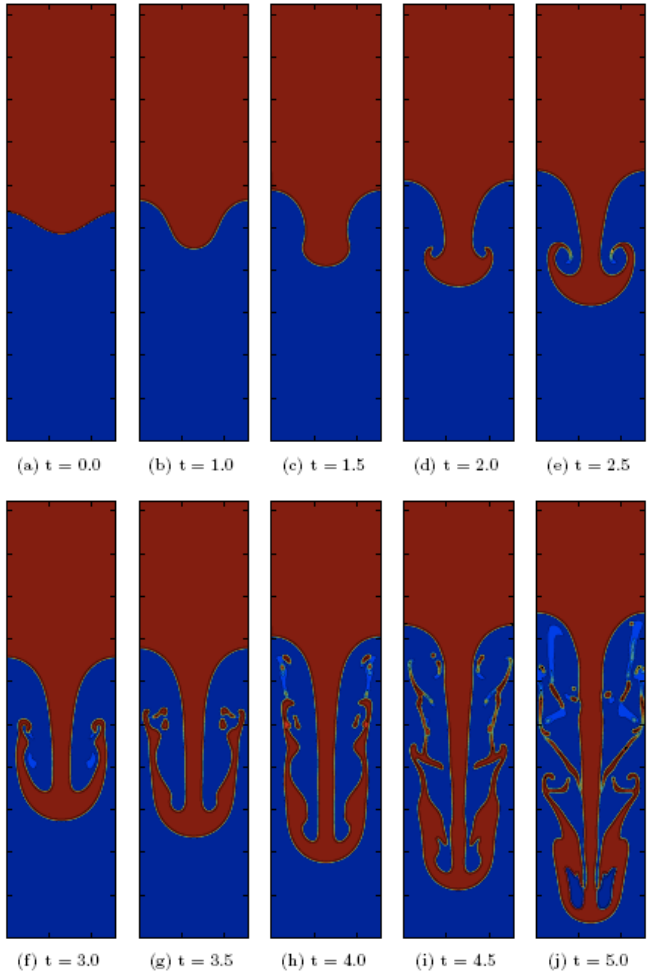


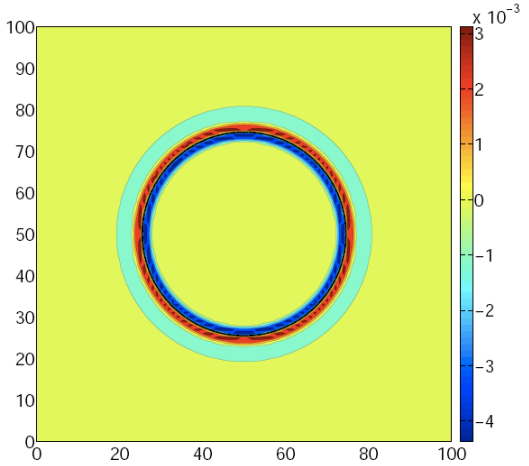




Moving droplets, jets: still a challenge







LB *multi-phase* code

Basically add 30 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 $$F_i = w_i \frac{\vec{F}\bullet\vec{c}_i}{c_s^2}$$

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

$$\vec{u} \rightarrow \vec{u}' = \vec{u} + \frac{\vec{F}\tau}{\rho}$$

LB Code Structure


```
c ================================
        program bgk2D
c ================================
        implicit double precision(a-h,o-z)
        include 'muphase.par'

c ------- input parameters

        omega=1.d0

        call input

c -------initialisation

        call inithydro1
        call initpop

c ------- MAIN LOOP

        do 10 istep = 1,nsteps
           call pbcdens
           call pbc
           call move
           call hydrovar1
           call hydrovar2
           if (iforce) then
              call force
           endif
           call equili
           call collis
           call hydrovar1
           call media
```

```
 if (mod(istep,ndiag).eq.0) then
              call out0d
           endif

           if (mod(istep,nout).eq.0) then
              call out1d(frce)
              call out2d(frce)
           endif

           if(mod(istep,100).eq.0) then
            write(6,*) 'completed time step ',istep
           endif


           if(mod(istep,nsteps).eq.0) then
             call savepop
           endif

           if(mod(istep,nsteps).eq.0) then
             call laplace
                endif

c-------- end of main loop

10      continue

        end
```

LB Code Structure

```
c ================================
      program bgk2D
c ================================
      implicit double precision(a-h,o-z)
      include 'muphase.par'

c ------- input parameters

      omega=1.d0

      call input

c -------initialisation

      call inithydro1
      call initpop

c ------- MAIN LOOP

      do 10 istep = 1,nsteps
         call pbcdens
         call pbc
         call move
         call hydrovar1
         call hydrovar2
         if (iforce) then
            call force
         endif
         call equili
         call collis
         call hydrovar1
         call media
```

```
      if (mod(istep,ndiag).eq.0) then
            call out0d
         endif

         if (mod(istep,nout).eq.0) then
            call out1d(frce)
            call out2d(frce)
         endif

         if(mod(istep,100).eq.0) then
          write(6,*) 'completed time step ',istep
         endif


         if(mod(istep,nsteps).eq.0) then
           call savepop
         endif

         if(mod(istep,nsteps).eq.0) then
           call laplace
                endif

c-------- end of main loop

10       continue

         end
```

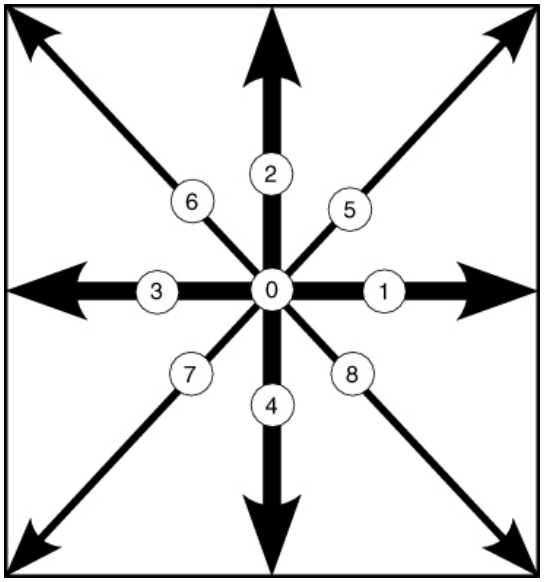
LB Code: input


```
c-------------------------------------------------
      subroutine input

      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-------------------------------------------------
      open(5,file='muphase.inp')

      print*,' Number of steps'
      read(5,*)nsteps

      print*,' Number of steps between printing
profile'
      read(5,*)nout

      print*,' Number of steps between performing
diagnostics'
      read(5,*)ndiag

      print*,'dt'
      read(5,*)dt

      print*,'viscosity'
      read(5,*)visc

      print*,' Coupling gx'
      read(5,*)gx

      print*,' Coupling gy'
      read(5,*)gy

      print*,' Coupling gxy'
      read(5,*)gxy

      print*,' Coupling gyx'
      read(5,*)gyx

      print*,'Insert rhopsi'
      read(5,*)rhopsi

 print*,'Applied force  (.TRUE. or .FALSE.) ?'
      read(5,*)iforce

      print*,' Initial density and velocity for the
Poiseuille force'
      read(5,*)rhoin,u0

      print*,' Final velocity for the Poise force'
      read(5,*) uf

      print*,' Linear obstacle ?'
      read(5,*) iobst

      if (iobst) then
         print*,' Length of the obstacle (multiple of 2)'
         read(5,*) nobst
      endif

      print*,' File for output: 5 chars'
      read(5,'(A)')fileout

      print*,' read populations dump (0 or 1)'
      read(5,*)dump

      close(5)

      open(10,file=fileout//'.ruvy')
      open(11,file=fileout//'.ruvy2')
      open(14,file=fileout//'.ruvx')
      open(15,file=fileout//'.ruvx2')
      open(30,file=fileout//'.probe')
      open(51,file=fileout//'.ruv2d')
      open(55,file='sez_ny_m.dat',status='unknown')

      print*,'****************************************'
      print*,' Lattice BGK model, 2D with 9
velocities'
      print*,'****************************************'
      print*,'Number of cells :',nx,'*',ny
      print*,'Nsteps :',nsteps

 print*,'Relaxation frequency :',omega
      print*,'Coupling gnn :',gnn
      print*,'Coupling gnnn :',gnnn
      print*,'Applied force :',iforce
      print*,'Initial velocity for this Poiseuille
force :',u0
      if (iobst) then
         print*,' Linear Obstacle with length :',
      endif
      write(6,*)'Output file :',fileout
c constants
      cs2  = 1.0d0 / 3.0d0
      cs22 = 2.0d0 * cs2
      cssq = 2.0d0 / 9.0d0
! input weights and storage in w(o:npop-1) arr
      w0 = 4.0d0 / 9.0d0
      w1 = 1.0d0 / 9.0d0
      w2 = 1.0d0 / 36.0d0
      w(0) = w0
      do i = 1, 4
        w(i) = w1
        w(i+4) = w2
      end do
      den = rhoin/float(npop)
! scaling
      dx = dt
      omega = 1.d0/(3.*visc*(dt*dt)/(dx*dx) + 0.5
 print*,' Viscosity :',visc,omega

! calculation of the constant applied force
      fpois = 8.0d0 * visc * uf / dfloat(ny) / dflo
      fpois = rhoin*fpois/6.  ! # of biased popul
      print*,' Intensity of the applied force ',fp

      return
      end
```

LB Code: inithydro


```
c-----------------------------------------------
      subroutine inithydro1
      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-----------------------------------------------
! modified to take into account the possibility of localized peturbation
      do j = 1, ny
        do i = 1, nx
          u1(i,j) = u0
          v1(i,j) = 0.d0
        enddo
      enddo
! Random Perturbation in all the Computational Domain
      do j = 1, ny
        do i = 1, nx
           rhod1(i,j) = rhoin*(1.d0 + 0.01d0 * (rand(0) - 0.5d0) * 2.d0)
!!!GF! Circular Droplet in the middle of the domain
!!!GF
!!!GF      do j = 0,ny+1
!!!GF       do i =0,nx+1
!!!GF
!!!GF       if(((i-nx/2)**2+(j-ny/2)**2).lt.324) then
!!!GF       rhod1(i,j)=1.875d0*(1.d0 + 0.01d0 * (rand(0) - 0.5d0) * 2.d0)
!!!GF       else
!!!GF        rhod1(i,j)=0.16d0
!!!GF       endif
      enddo
     enddo
      return
      end
c -----------------------------------------------
```

LB Code: initpop




```
c ---------------------------------------------------
      subroutine initpop

      implicit double precision(a-h,o-z)
      include 'muphase.par'
c---------------------------------------------------
      if(dump.eq.1)then
       call resume
      else

        do j = 1, ny
         do i = 1, nx
          do k = 0, npop-1
            f(k,i,j) = rhod1(i,j)*w(k)
          enddo
         enddo
        enddo
      endif

      rhoaver = 0.d0

        do j = 1, ny
          do i = 1, nx
            do k = 0, npop-1
              rhoaver = rhoaver + f(k,i,j)
            enddo
          enddo
        enddo

      rhoaver = rhoaver / dfloat(nx*ny)
      dinvrho = 1.d0 / rhoaver
      print*,'average density at t=0:', rhoaver

      return
      end
```

LB code: pbcdens and pbc

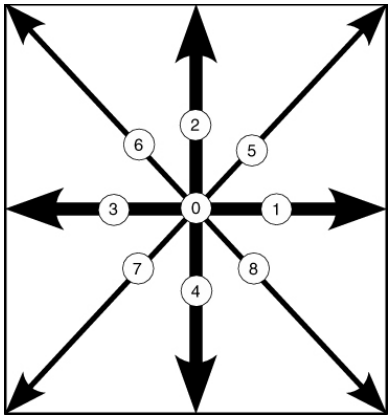
```
c ========================
      subroutine pbcdens
c ========================
      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-------------------------------------------------------------

      do j = 1, ny
        rhod1(0,j) = rhod1(nx,j)
        rhod1(nx+1,j) = rhod1(1,j)
      enddo

      do i = 1, nx
        rhod1(i,0) = rhod1(i,ny)
        rhod1(i,ny+1) = rhod1(i,1)
      enddo

        rhod1(0,0) = rhod1(nx,0)
        rhod1(nx+1,0) = rhod1(nx+1,ny)
        rhod1(nx+1,ny+1) = rhod1(1,ny+1)
        rhod1(0,ny+1) = rhod1(0,1)


      return
      end
```

```
c ========================
      subroutine pbc
c ========================
      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-------------------------------------------------------------

      do j = 1, ny
        do k = 0, npop-1
          f(k,0,j) = f(k,nx,j)
          f(k,nx+1,j) = f(k,1,j)
        enddo
      enddo

      do i = 1, nx
        do k = 0, npop-1
          f(k,i,0) = f(k,i,ny)
          f(k,i,ny+1) = f(k,i,1)
        enddo
      enddo

      do k = 0, npop-1
        f(k,0,0) = f(k,nx,0)
        f(k,nx+1,0) = f(k,nx+1,ny)
        f(k,nx+1,ny+1) = f(k,1,ny+1)
        f(k,0,ny+1) = f(k,0,1)
      enddo

      return
      end
```

LB Code: move

```
c-------------------------------------------
      subroutine move
c-------------------------------------------
      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-------------------------------------------
      do j = 0,ny+1
        do i = 0, nx+1
          do k=0,npop
            fp(k,i,j)=f(k,i,j)
          enddo
        enddo
      enddo

      do j = 1,ny
        do i = 1, nx
          f(2,i,j) = fp(2,i,j-1)
          f(6,i,j) = fp(6,i+1,j-1)
          f(1,i,j) = fp(1,i-1,j)
          f(5,i,j) = fp(5,i-1,j-1)
          f(4,i,j) = fp(4,i,j+1)
          f(8,i,j) = fp(8,i-1,j+1)
          f(3,i,j) = fp(3,i+1,j)
          f(7,i,j) = fp(7,i+1,j+1)
          f(0,i,j) = fp(0,i,j)
        enddo
      enddo
      return
      end
```

Two memory levels (lazy programmer)



LB Code: hydrovar (1 & 2)


```
 subroutine hydrovar1

       implicit double precision(a-h,o-z)
       include 'muphase.par'
c--------------------------------------

       do j = 1, ny
         do i = 1, nx
           rhod1(i,j) = 0.d0
           do k = 0, npop-1
             rhod1(i,j) = rhod1(i,j) + f(k,i,j)
           enddo
         enddo
       enddo
c Calculation of velocities
       do j = 1, ny
         do i = 1, nx
          rho1 = 1.d0 / rhod1(i,j)
          u1(i,j) = ( f(1,i,j) - f(3,i,j) + f(5,i,j) -
     &              f(6,i,j) - f(7,i,j) + f(8,i,j) ) * rho1

          v1(i,j) = ( f(5,i,j) + f(2,i,j) + f(6,i,j)
     &              - f(7,i,j) - f(4,i,j) - f(8,i,j) ) * rho1
         enddo
       enddo

       return
       end
```

LB code: force

```
c ==========================================
      subroutine force
c ==========================================
      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-------------------------------------------------------------
      frce = fpois   ! constant external force
      do i=-1,nx+2
       do j=-1,ny+2
         psi(i,j) =rhopsi * ( 1.d0 - exp(- rhod1(i,j) / rhopsi))
       enddo
      enddo

      wnn=w1/cs2
      wnnn=w2/cs2

      ww_n  = ww1/cs2
      ww_nn = ww2/cs2



      do j = 1, ny
       do i = 1, nx
c non local Lennard-Jones (may put a density functional, not density
itself)

       fnnx  = psi(i,j)*(psi(i+1,j)-psi(i-1,j))
       fnny  = psi(i,j)*(psi(i,j+1)-psi(i,j-1))
       fnnnx = psi(i,j)*((psi(i+1,j-1)+psi(i+1,j+1)
     &                -psi(i-1,j-1) - psi(i-1,j+1)))
       fnnny = psi(i,j)*((psi(i+1,j+1) + psi(i-1,j+1)
     &                - psi(i-1,j-1) - psi(i+1,j-1)))
```

```
! interaction forces

      f2x =-(gnn * fnnx*1./9.+ gnn * fnnnx *1./36.)
      f2y =-(gnn * fnny*1./9.+ gnn * fnnny *1./36.)

      force_x(i,j)=f2x
      force_y(i,j)=f2y

      f_x(i,j) =f2x

      f_y(i,j) =f2y

     end do
     end do


! Velocity  Shifting

     do i=1,nx
       do j=1,ny

        u1(i,j)=u1(i,j)+force_x(i,j)/(omega*rhod1(i,j))

        v1(i,j)=v1(i,j)+force_y(i,j)/(omega*rhod1(i,j))

       end do
     end do


      return
      end
```

LB Code: equilibrium

```
c-------------------------------------------------
      subroutine equili
      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-------------------------------------------------
      do j = 1, ny
        do i = 1, nx
          usq = u1(i,j) * u1(i,j)
          vsq = v1(i,j) * v1(i,j)
          sumsq = (usq + vsq) / cs22
          sumsq2 = sumsq * (1.0d0 - cs2) / cs2
          u22 = usq / cssq
          v22 = vsq / cssq
          ui = u1(i,j) / cs2
          vi = v1(i,j) / cs2
          uv = ui * vi
          rhoij = rhod1(i,j)

          feq(0,i,j) = (4.d0/9.d0)*rhoij*(1.0d0 - sumsq)

          feq(1,i,j) = (1.d0/9.d0)*rhoij*(1.0d0 - sumsq + u22 + ui)
          feq(2,i,j) = (1.d0/9.d0)*rhoij*(1.0d0 - sumsq + v22 + vi)
          feq(3,i,j) = (1.d0/9.d0)*rhoij*(1.0d0 - sumsq + u22 - ui)
          feq(4,i,j) = (1.d0/9.d0)*rhoij*(1.0d0 - sumsq + v22 – vi)
          feq(5,i,j) = (1.d0/36.d0)*rhoij*(1.0d0 + sumsq2 + ui + vi  + uv)
          feq(6,i,j) = (1.d0/36.d0)*rhoij*(1.0d0 + sumsq2 - ui + vi   - uv)
          feq(7,i,j) = (1.d0/36.d0)*rhoij*(1.0d0 + sumsq2 - ui – vi   + uv)
          feq(8,i,j) = (1.d0/36.d0)*rhoij*(1.0d0 + sumsq2 + ui - vi   - uv)
        enddo
 enddo

      return
      end
```

$$f_i^{eq}(\vec{r};t) = w_i \rho \left(1 + \frac{\vec{u}\bullet\vec{c}_i}{c_s^2} + \frac{(\vec{u}\bullet\vec{c}_i)^2}{2c_s^4} - \frac{\vec{u}\bullet\vec{u}}{2c_s^2} \right)$$

LB Code: collision and media

```
c-------------------------------------------------------
      subroutine collis

      implicit double precision(a-h,o-z)
      include 'muphase.par'

c-------------------------------------------------------
       do j = 1, ny
         do i = 1, nx
           do k = 0, npop-1
            f(k,i,j) = f(k,i,j) * (1.0d0 - omega)
     &                + omega * feq(k,i,j)
           end do
         end do
       end do

      return
      end
```

```
c=========================================
      subroutine media
c =========================================
      implicit double precision(a-h,o-z)
      include 'muphase.par'
c-------------------------------------------------------

        do i=1,nx
          do j=1,ny

            u1(i,j)=(u1(i,j)+u2(i,j))*0.5d0
            v1(i,j)=(v1(i,j)+v2(i,j))*0.5d0

          end do
        end do
      return
      end
```

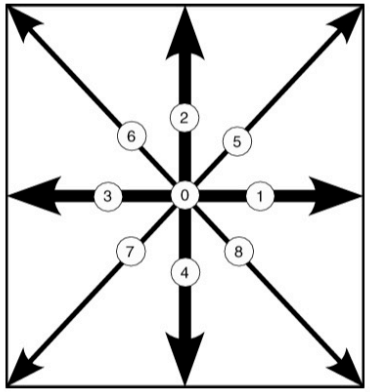
Shan-Chen, pros and cons:

✓ ***Simple, elegant, efficient***

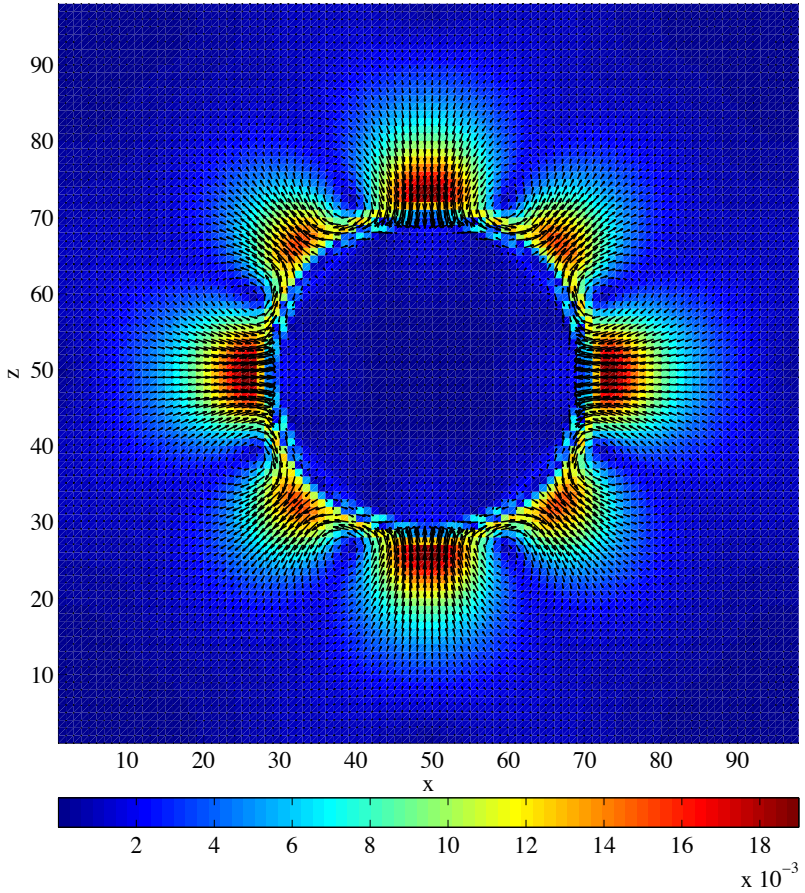
- ***Spurious currents (isotropy)***
- ***EoS and surftens, same G***
- ***Narrow surftens range***
- ***Thick interface (w~10 dx)***
- ***Sound speed gas > liquid***

Spurious Currents

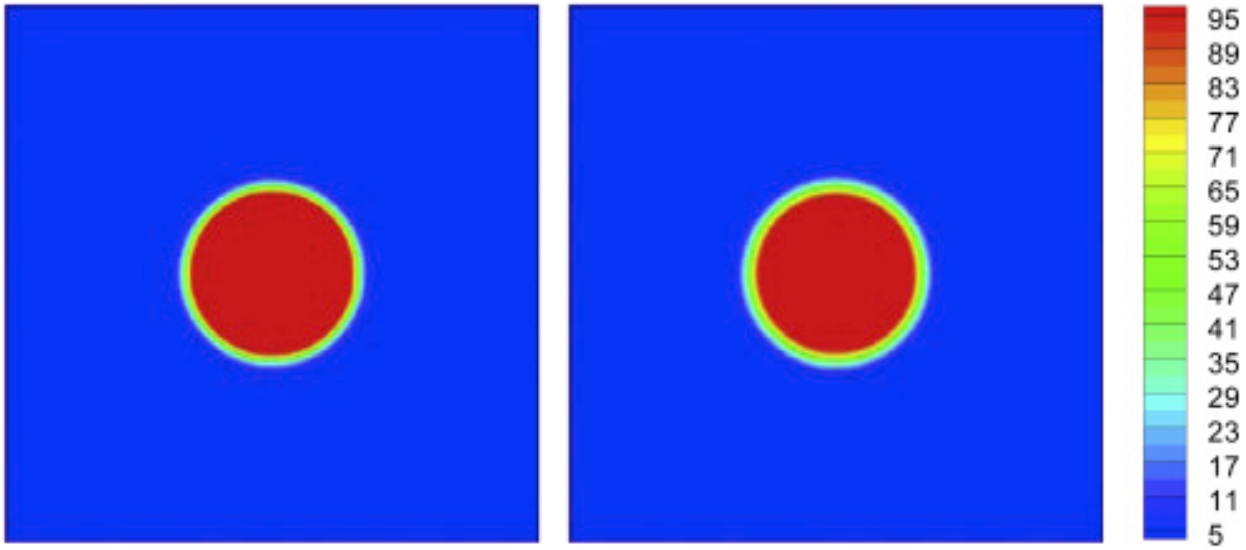
Easy access to major configurational complexity, comes at a price...



These modes have no citizenship in the Continuum!
Near-grid physics always UV exposed

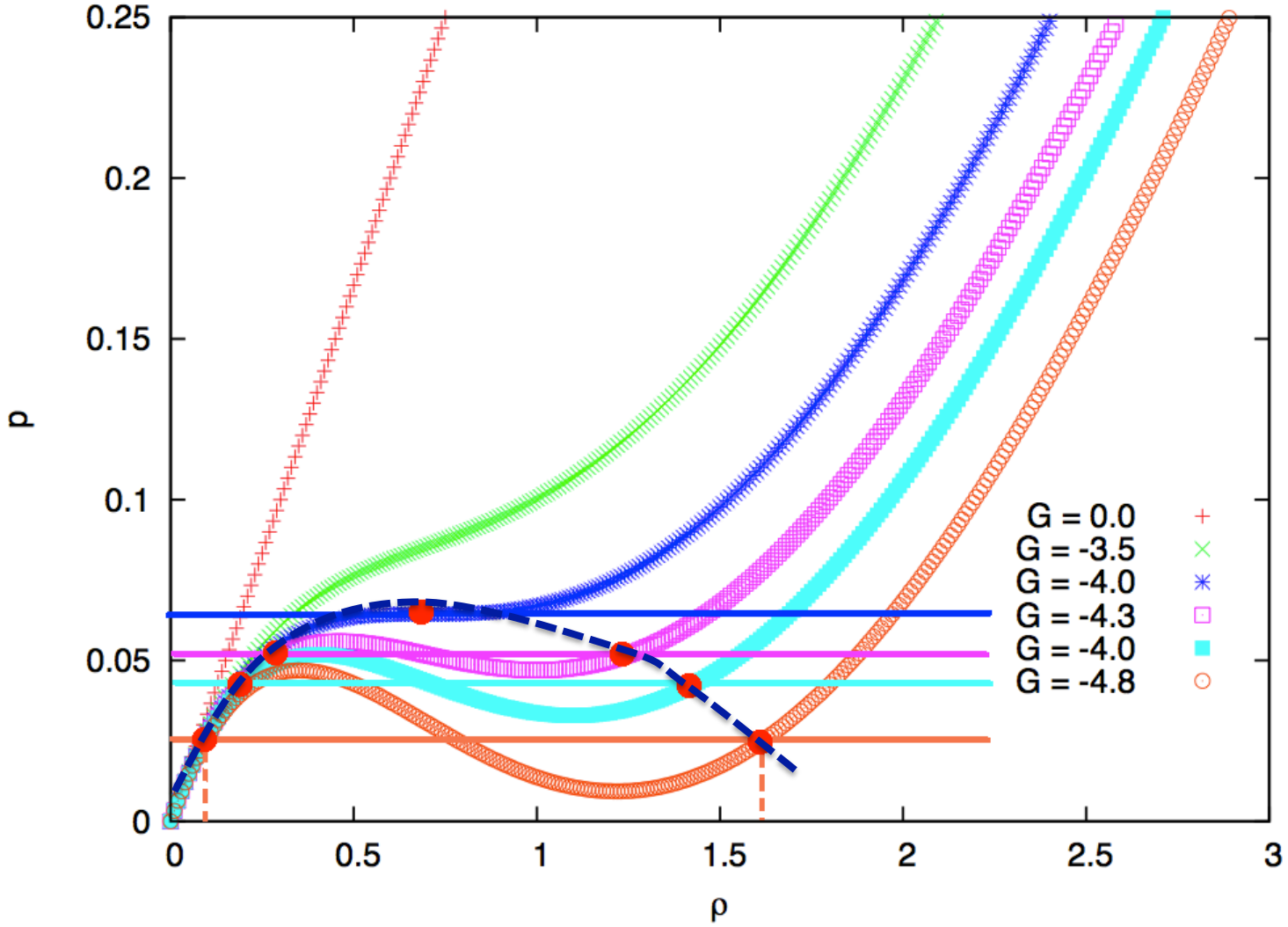


The interface pops out on its own.
But it is a diffuse interface, w=5-10 lattice spacings
Waste of resolution



Sound speed

Gas: 1/3, Liquid <1/3: unphysical



Beyond Shan-Chen

Two-parameters equations of state

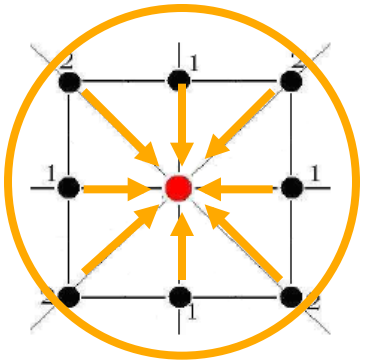
Multi-range pseudo-potentials

Rule-Driven Chromodynamics

.....

Standard vs. 2-belt

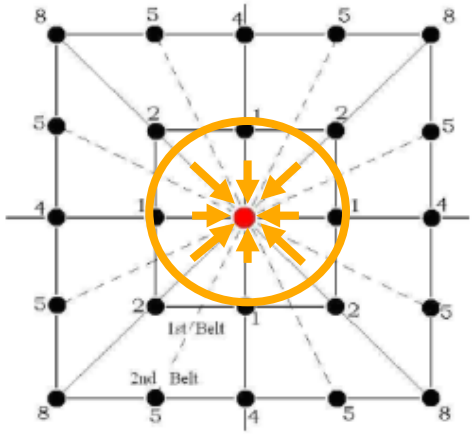
Standard Shan-Chen model



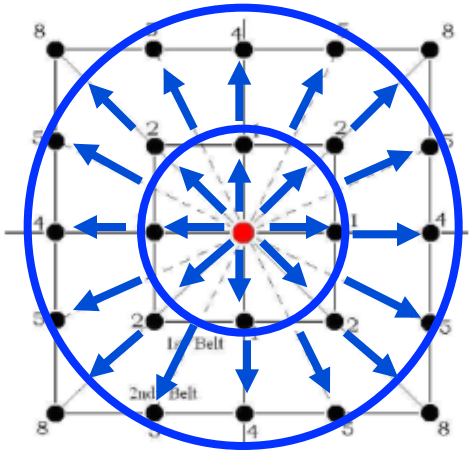
E(4)
w(1) = 1 / 9
w(2) = 1 / 36

$$\begin{cases} \sum_{i=0}^{b_1} w_i = \sum_{i=0}^{b_1} p_{si} + \sum_{i=1}^{b_2} p_{mi} = 1 \\ \sum_{i=1}^{b_1} w_i c_i^2 = \sum_{i=1}^{b_1} p_{si} c_{si}^2 + \sum_{i=1}^{b_2} p_{mi} c_{mi}^2 = c_s^2 \end{cases}$$

2-belt model

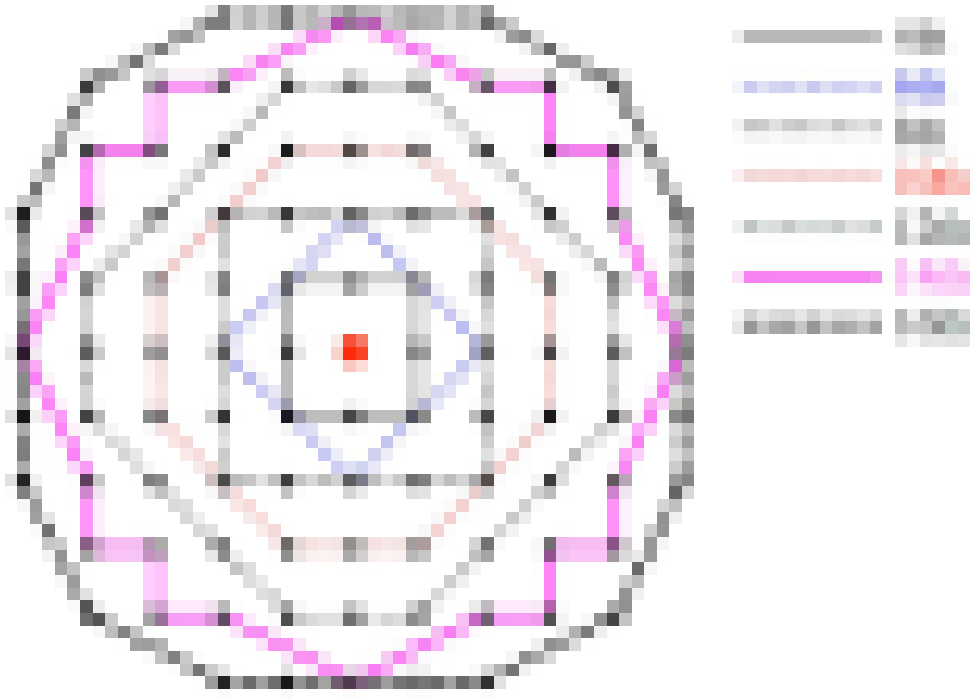


+



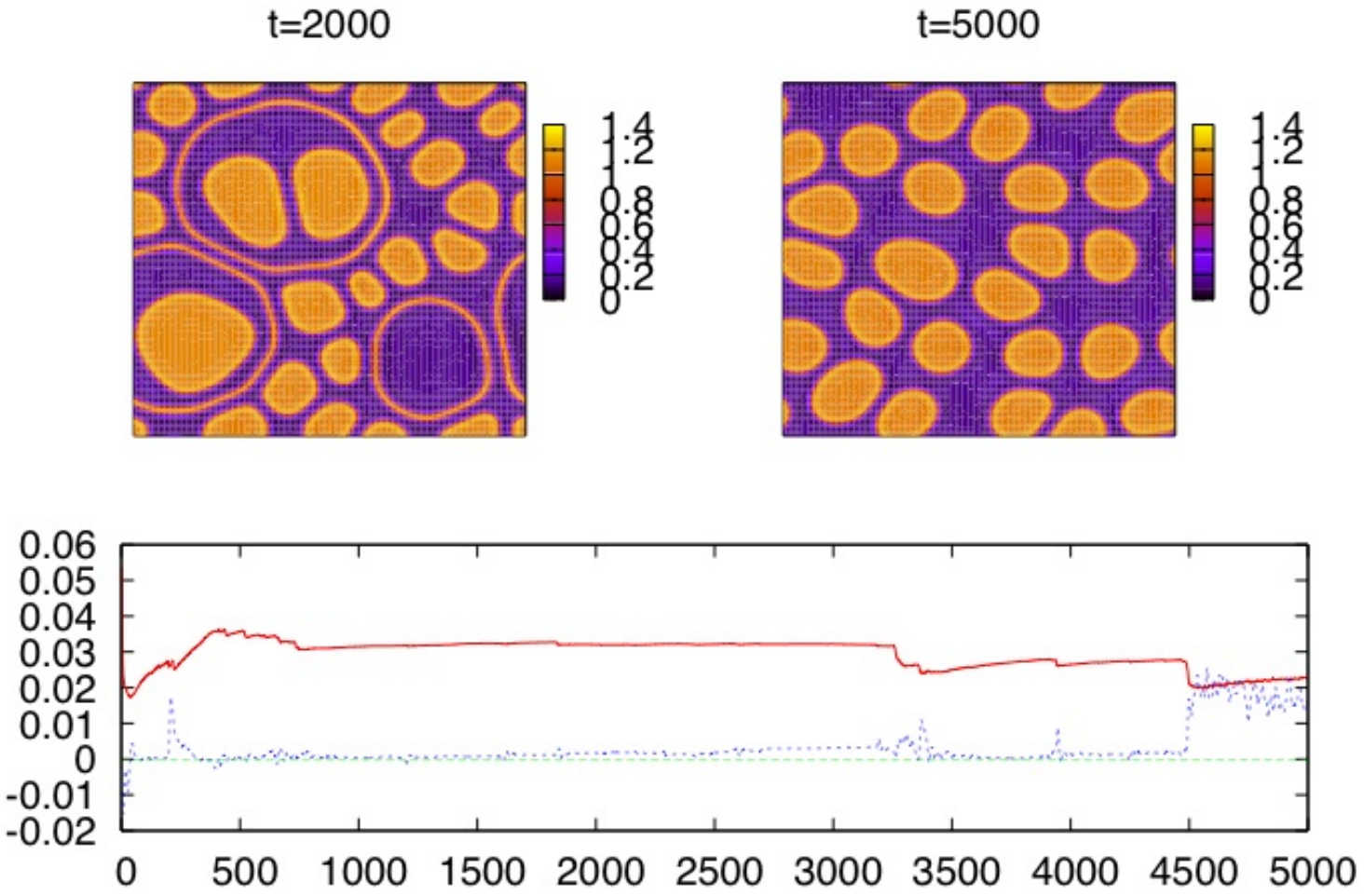
$E(8)$	
$p_{si}=p(1)=4/63,$	$i=1,4$
$p_{si}=p(2)=4/135,$	$i=5,8$
$p_{mi}=p(4)=1/180,$	$i=1,4$
$p_{mi}=p(5)=2/945,$	$i=5,12$
$p_{mi}=p(8)=1/15120,$	$i=13,16$

Higher order lattices



(Sbragaglia, Benzi, Biferale, SS, PRE 2007)

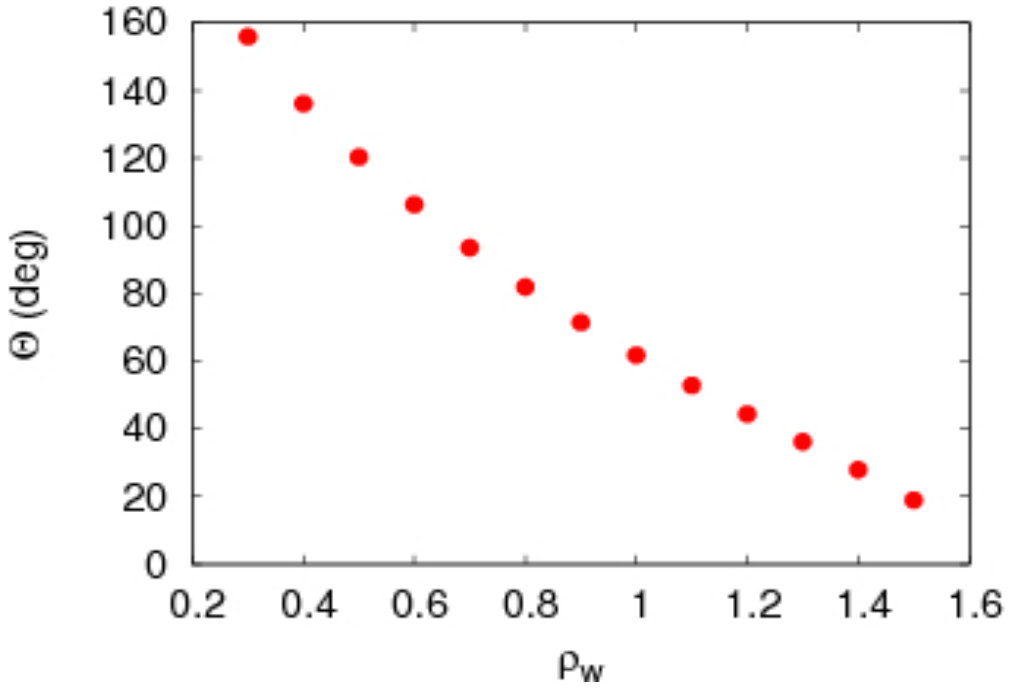
A/R competition: Foams/Emulsions



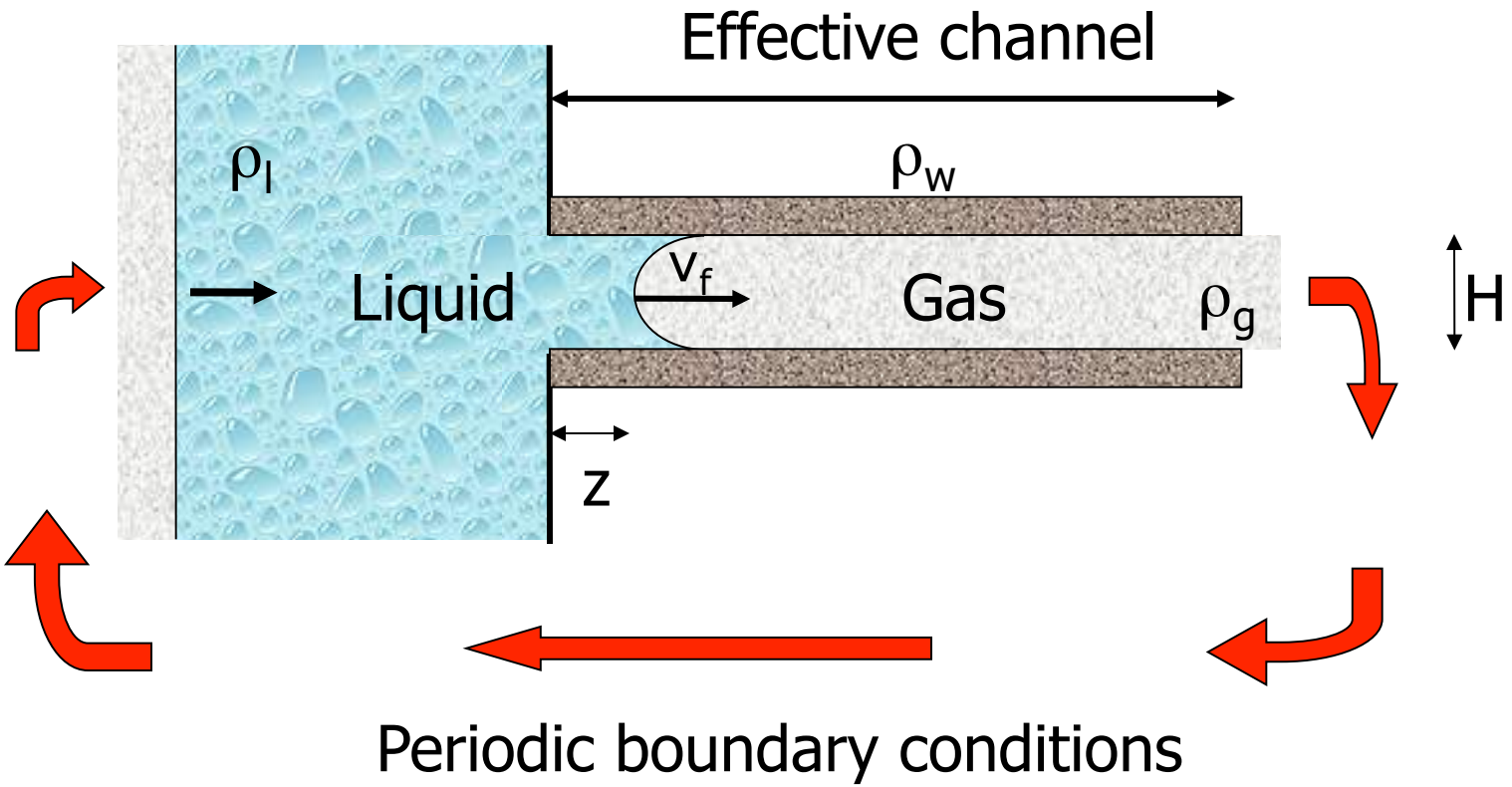
Fluid-wall interactions

$$\vec{F}_{FW}(x) = G_{FW}\Psi(x)\sum_i \vec{c}_i \Psi(\rho(x + c_i))$$

We assign at the points belonging to the wall a density ρ_w
Tuning ρ_w we are able to modify and control the contact angle θ



Microfluidics



Summary

Pseudo-potential LB is very handy, easy to learn&program
Qualitative results come by very swiftly.

"LB cannot be too wrong" (A. Ladd)

Many weaknesses, most can be mitigated

MUCH simpler than grid-based solution of non-ideal Navier-Stokes equations,
Especially for flows with high S/V ratios

Very popular, major LB mainstream

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

Forcing terms: projection route

$$F_i = w_i[F^0 1_i + \vec{F}^1 \bullet \vec{c}_i + \overleftrightarrow{F}^2 : (\vec{c}_i\vec{c}_i - c_s{}^2\overleftrightarrow{I})]$$

$$F^0 = R$$

$$\vec{F}^1 = \rho\vec{a}/c_s{}^2$$

$$\overleftrightarrow{F}^2 = (\rho\vec{a}\vec{u} + \rho\vec{u}\vec{a})/2c_s{}^4$$

$$c_s{}^2 = (\sum_i w_i c_i{}^2)/D$$

Shifted equils: the Fokker-Planck connection

$$u \cdot c_i \rightarrow (u + a\tau) \cdot c_i$$

$$uuc_i c_i \rightarrow (u + a\tau)(u + a\tau)c_i c_i = uuc_i c_i + 2ua\tau c_i + aa\tau\tau$$

$$\tau^2/_2 (\vec{a}\vec{a} : \nabla_v \nabla_v) f$$

Lattice Fokker-Planck: diffusion in v-space

External/Internal Forces: Shifted equilibria

$$C(f) \approx -\frac{\vec{F}}{m} \bullet \vec{\nabla}_v f^{eq} - \omega(f^{eq} - f)$$

$$-\frac{\vec{F}}{m} \bullet \vec{\nabla}_v f^{eq} + \omega(f^{eq} - f) = +\omega[f^{eq}(v - a\tau) - f] + O(\tau^2)$$

$$(v - a\tau) - u = v - (u + a\tau) = v - \tilde{u}$$

$$\tilde{u} = u + a\tau$$

Discrete speeds untouched: free-flight!!!

A nice property of Hermite projection

$$f(x,v) = W(v)[1 + \beta u(x) H_1(v) + \frac{\beta^2}{2} P(x) H_2(v)]$$

$$f^{eq}(x,v) = W(v)[1 + \beta u(x) H_1(v) + \frac{\beta^2}{2} P^{eq}(x) H_2(v)]$$

$$W(v) = (2\pi)^{-1/2} \, e^{-v^2/2}$$

$$\partial_v f - \partial_v f^{eq} = -W(v) P^{neq} [-v + \partial_v H_2(v)] = 0 + O(H_3(v))$$

$$\partial_v W = -vW; \;\; \partial_v H_2 = H_1$$

$$H_1(v) = v; \;\; vH_n = H_{n+1} + nH_{n-1}$$

Coding up Shan-Chen (a new world in a few lines …)

```
For (ix=1;ix<Nx;ix++{
  For (iy=1;iy<Ny;iy++{
    Fx=Fy=0;                    /*  SC  Force */
    for(k=0;k<Npop;k++ {
        PsiC  =  1.-exp(-rho[ix][iy]);
        PsiE  =  wnn*(1.-exp(-rho[ix+1][iy]));
        PsiW =  wnn*(1.-exp(-rho[ix-1][iy]));
        PsiNE = wnnn*(1.-exp(-rho[ix+1][iy+1]))
        ……
        Fx =+ G*PsiC*(PsiE-PsiW+PsiNE-PsiNW+PsiSE-PsiSW);
        Fy =+ G*PsiC*(PsiN-PsiS+PsiNE-PsiSE+PsiNW-PsiSW);
     }

/*      Update populations  */

        fE[ix][iy]   =  fE[ix][iy]  + wnn*Fx*tau/rho[ix][iy];
        fS[ix][iy]   =  fS[ix][iy]   - wnn*Fx*tau/rho[ix][iy];
        fNE[ix][iy] = fNE[ix][iy] + wnnn*(Fx+Fy)*tau/rho[ix][iy];
        …….
  }
}
```