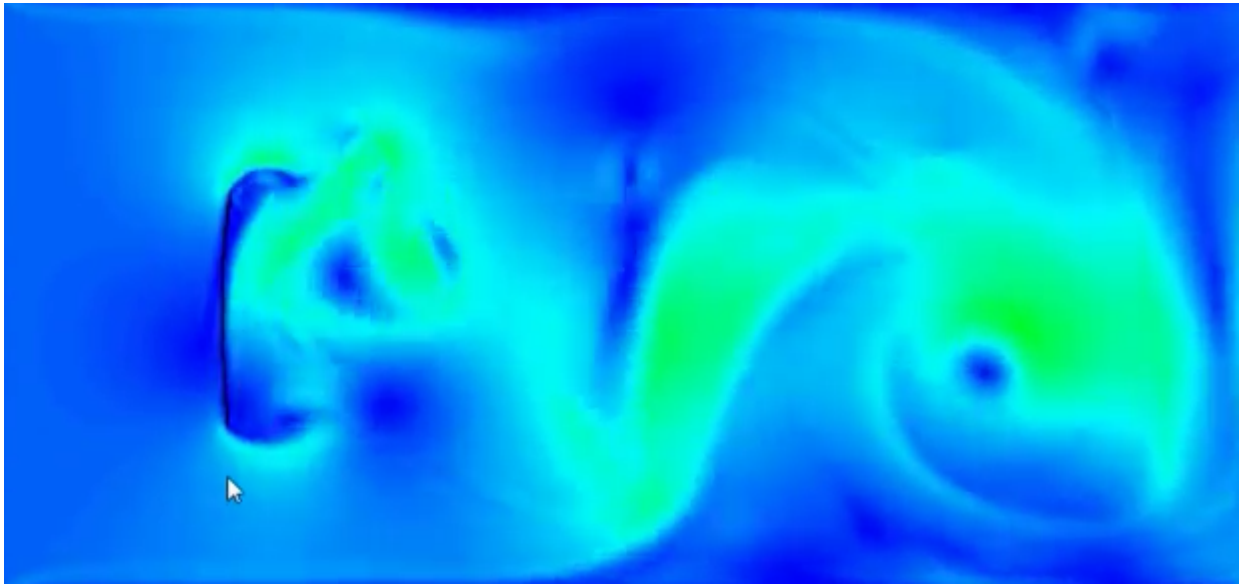
Computer simulations of fluid dynamics

Lecture 11 – LBM: Algorithm for BGK

Maciej Matyka



https://www.youtube.com/watch?v=VLUyP_ydfJc

https://youtu.be/CJ52ZpggKa

Use of the Boltzmann Equation to Simulate Lattice-Gas Automata

Guy R. McNamara and Gianluigi Zanetti[(a)]

The Research Institutes, The University of Chicago, 5640 South Ellis Avenue, Chicago, Illinois 60637

(Received 29 July 1988)

We discuss an alternative technique to the lattice-gas automata for the study of hydrodynamic properties, namely, we propose to model the lattice gas with a Boltzmann equation. This approach completely eliminates the statistical noise that plagues the usual lattice-gas simulations and therefore permits simulations that demand much less computer time. It is estimated to be more efficient than the lattice-gas automata for intermediate to low Reynolds number $R \lesssim 100$.

PACS numbers: 47.10.+g

In this Letter we describe a simulation technique that can be used in some situations as an alternative to the lattice-gas automaton. The latter has been proposed as a new technique[1] for the numerical study of the Navier-

equivalent to the LGA since, by factorizing the LGA collision operator on the one-particle distribution functions, we completely neglected all the effects due to the correlations between the particles. Nevertheless, the

(789 citations)

Lecture goal

- Give a complete introduction to LBM so that you my start writing your own solver **today**.

- You know nothing about LBM, after this lecture you will think its **so simple**... :-)

Suggestion

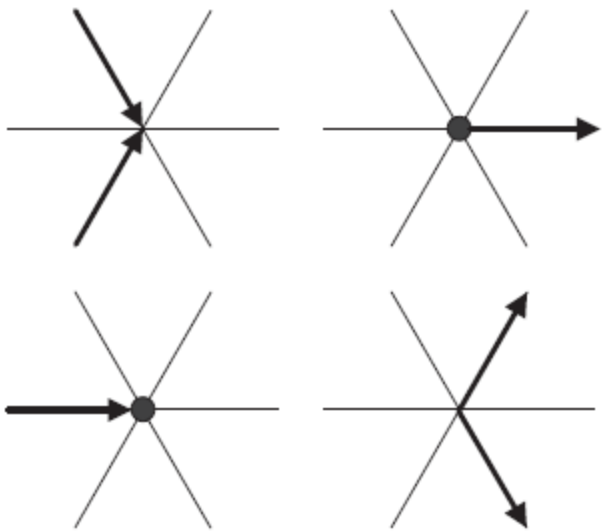
- <u>After the lecture</u> you get some material from me.
- <u>You come to the lab</u> and start writing the solver.
- <u>After 2h</u> you get it almost done.
- <u>Next week</u> your 2nd project is done.

CFD Methods

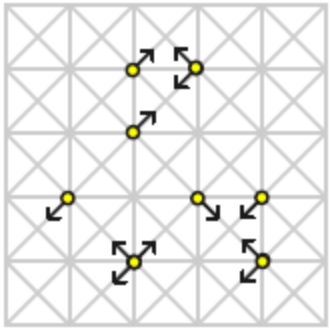
- Navier–Stokes Equations (NSE)
  - Finite Difference Method (FDM)
  - Finite Volume Method (FVM)
  - Finite Element Method (FEM)
  - Spectral Element Methods (SEM)
  - Boundary Element Methods (BEM)
  - High-res discretization (HRES)
  - Vorticity-Stream function (VOR)
- Particle in Cell (PIC)
- Material Point Method (MPM)
- **Smoothed Particle Hydrodynamics** (SPH)
- **Dissipative Particle Dynamics (DPD)**
- Vortex-in-Cell (VIC)
- **The Lattice Boltzmann Method (LBM)**
- Lattice Gas Automata, HPP, FHP (LGA)
- **Multiparticle Collision Dynamics, Stochastic Rotation Dynamics (MPC, SRD)**
- Molecular Dynamics (MD)

LGA gas

- U. Frish, B. Hasslacher and Y. Pomeau (FHP)



Collisions



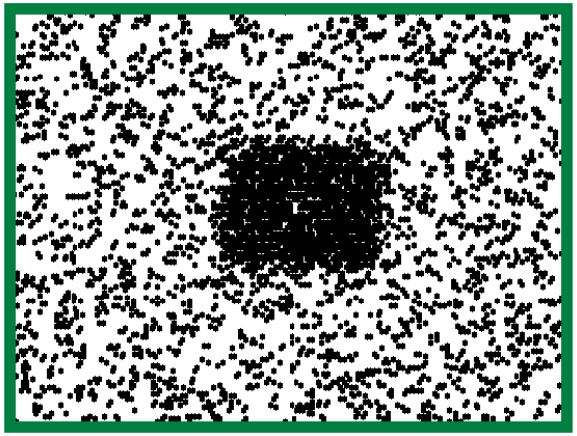
1) Transport
2) Collisions

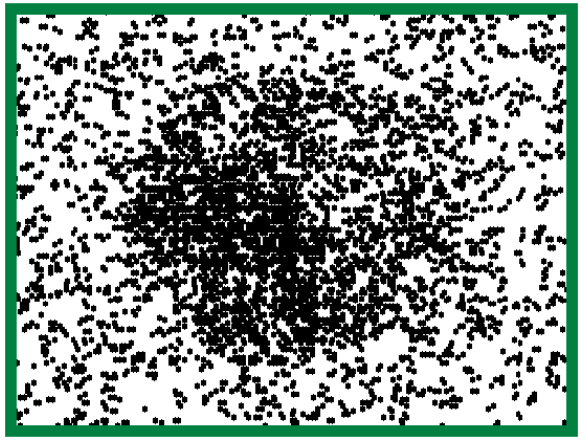
U. Frish, B. Hasslacher, Y. Pomeau 1986 *Lattice-Gas Automata for the Navier-Stokes Equation*, Phys. Rev. Lett. 56, 1505–1508.

Matyka, M. and Koza, Z., *Spreading of a density front in the Kuentz-Lavallee model of porous media* J. Phys. D: Appl. Phys. 40, 4078-4083 (2007)

LGA gas automata

- LGA is **far from perfect**:
  - Statistical noise
  - Requires averaging in <u>time / space</u>
  - Requires <u>large lattices</u>

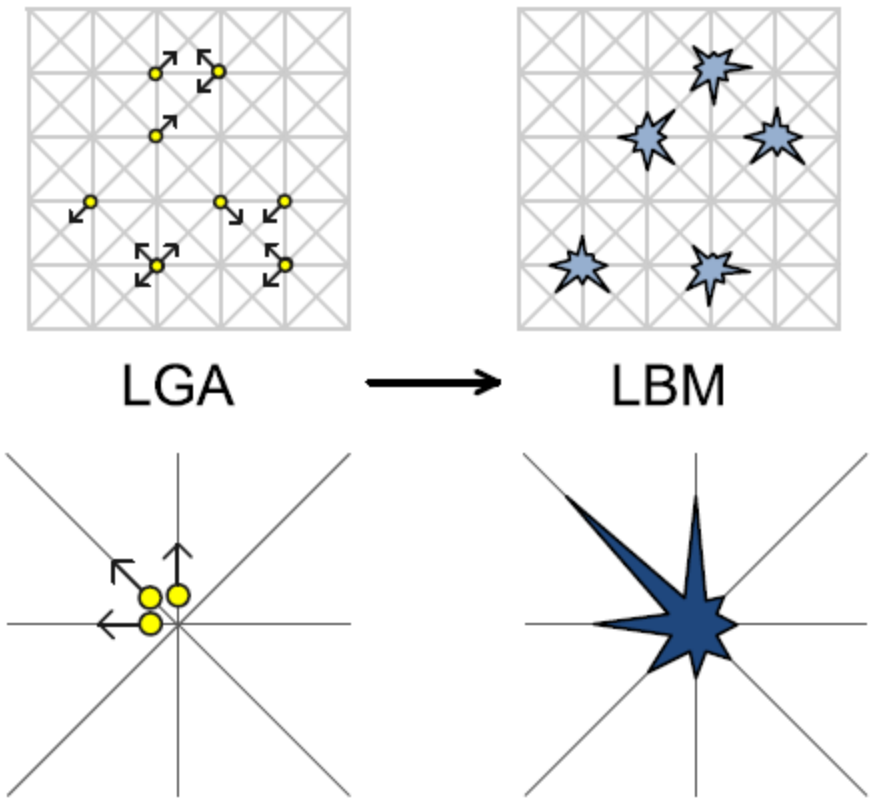




W. Saramak, FHP

The Lattice Boltzmann Method

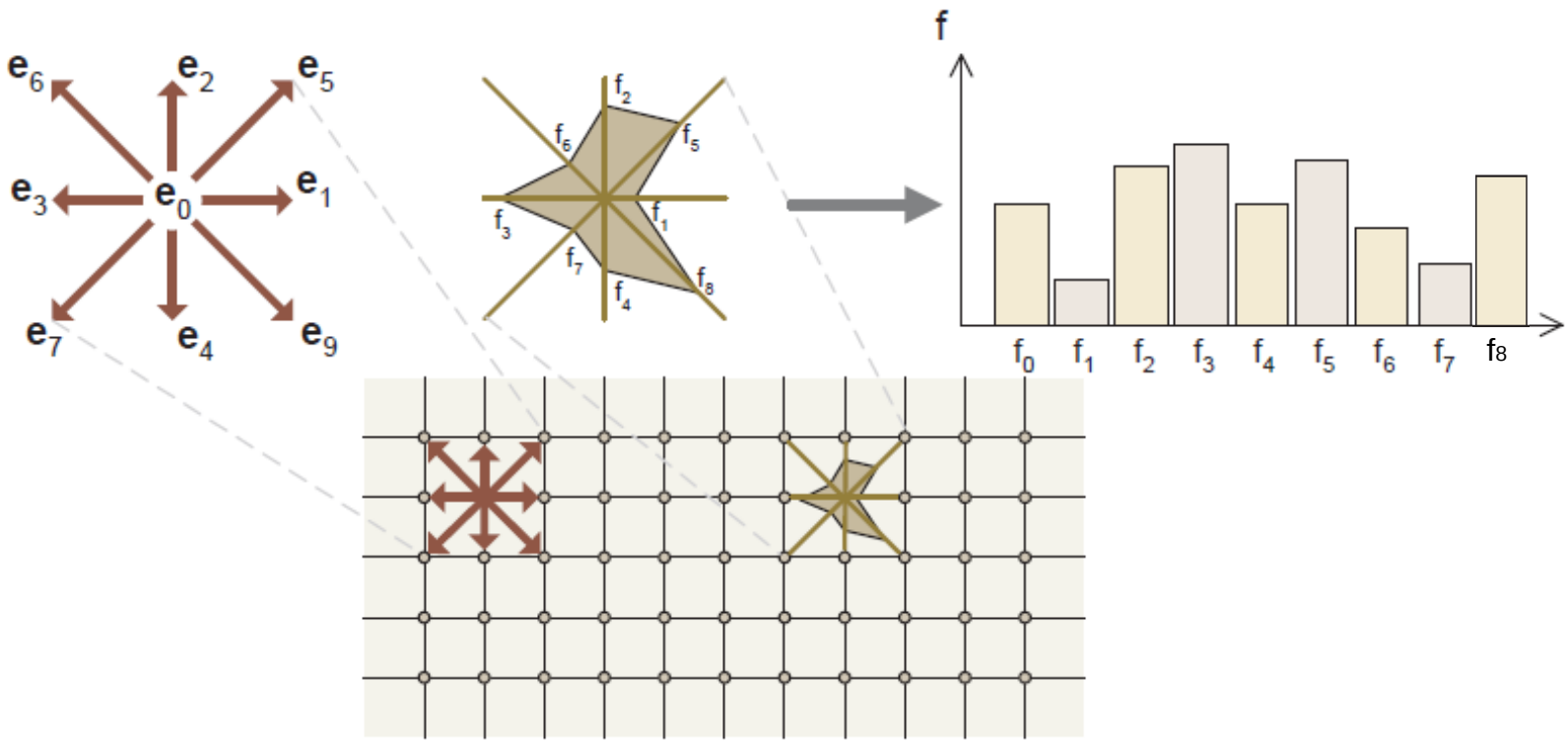
- Historical development of LGA



- Boolean n_i (0,1) variables replaced with $f_i \in [0, 1]$

D2Q9

- Discrete LBM model in 2D
- Distribution function on a lattice (9 directions)



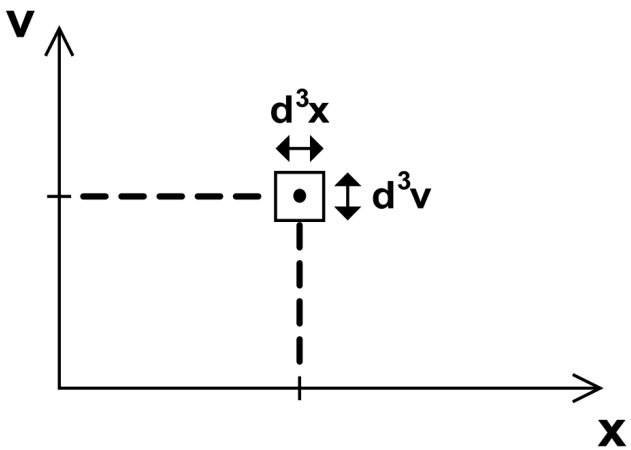
Thorne, D., Sukop, M., *Lattice Boltzmann Method for the Elder Problem*, FIU, CMWR(2004)

$$f(\mathbf{r},\mathbf{v},t)$$

distribution function

Distribution function f

- Number of particles in finite element of momentum / position space:



$$f(\mathbf{x}, \mathbf{v}, t)\, d^3x\, d^3v$$

- The task of kinetic theory is to find distribution function for a given model of interactions
- (i.e. gas, fluids)

The Boltzmann Equation

- **Time evolution** of distribution function:

$$(\partial_t + \mathbf{v} \cdot \nabla) f(\mathbf{r}, \mathbf{v}, t) = (\partial_t f)_{coll}$$

- Assumptions:
  - Molecular chaos (no velocity – position correlations)
  - Two-body collisions in microscale

Collision term: BGK

- Bhatnagar, Gross, Krook (1954):

$$(\partial f)_{\text{coll}} \approx -\frac{(f - f^{eq})}{\tau},$$

- Linear relaxation to equilibrium function f_{eq}

d'Humieres, D., Ginzburg, I., Krafczyk, M. Lallemand, P. and Luo, L.-S., *Multiple-relaxation-time lattice Boltzmann models in three dimensions*, Phil. Trans. R. Soc. Lond. A 360, 437-451 (2002)

Collisions: BGK approximations

- Bhatnagar, Gross, Krook (1954):

$$(\partial f)_{\text{coll}} \approx -\frac{(f - f^{eq})}{\tau},$$

- Linear relaxation to equilibrium function f_{eq}

- f_{eq}, i.e. from **Maxwell-Boltzmann distribution**

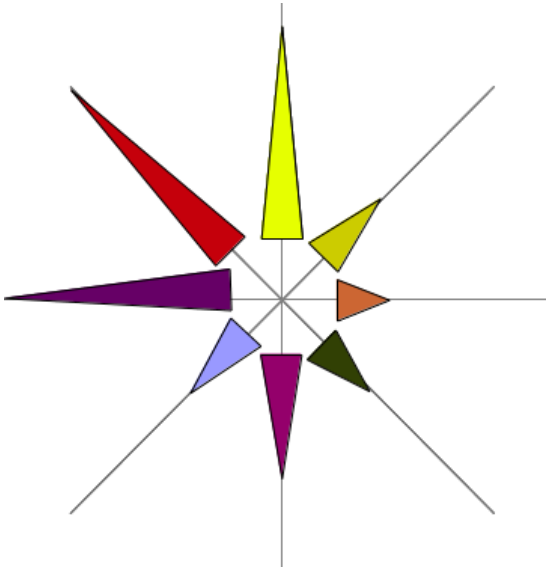
$$f^{eq} = \frac{\varrho}{(2\pi k_b T)^{d/2}} \exp\left[-\frac{\mathbf{v}^2}{2k_b T}\right]$$

d'Humieres, D., Ginzburg, I., Krafczyk, M. Lallemand, P. and Luo, L.-S., *Multiple-relaxation-time lattice Boltzmann models in three dimensions*, Phil. Trans. R. Soc. Lond. A 360, 437-451 (2002)

Macroscopic variables

- Density:

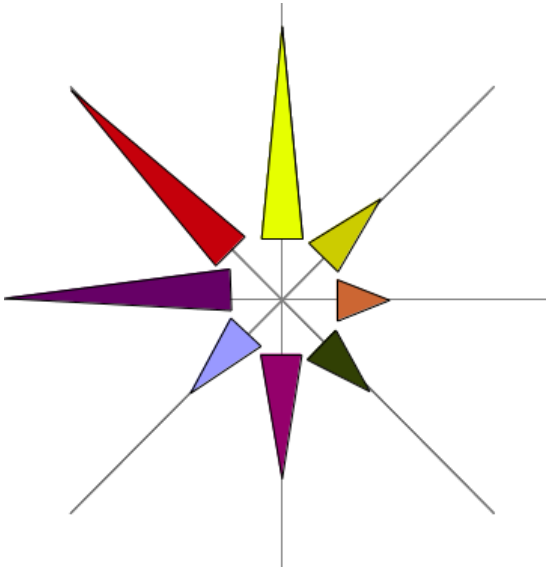
$$\varrho = \sum_{i=1}^{9} f_i$$



Macroscopic variables

- Density:

$$\varrho = \sum_{i=1}^{9} f_i$$



- Velocity:

$$\varrho \mathbf{u} = \sum_{i=1}^{9} f_i \mathbf{e}_i$$

Discrete form of transport equation

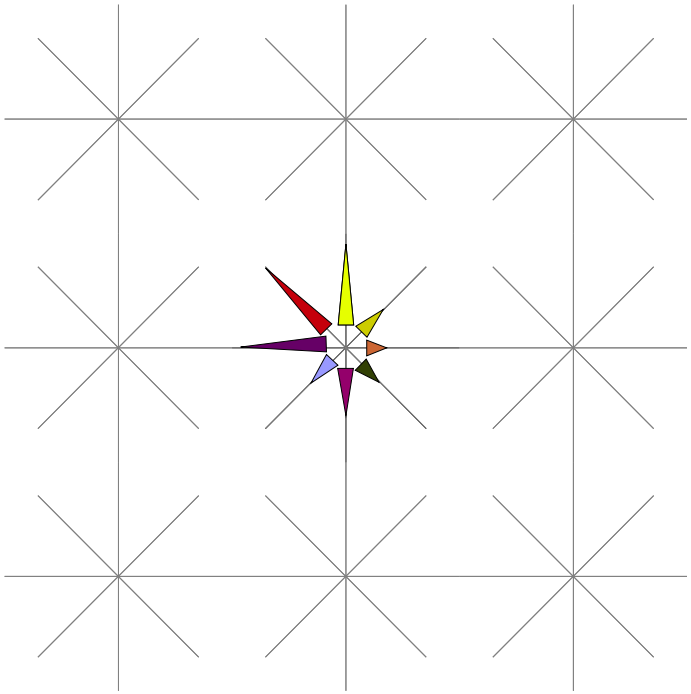
- Transport equation (BGK):

$$\underbrace{f_i(\mathbf{x}+\mathbf{c}_i, t+1) - f_i(\mathbf{x}, t)}_{\text{Transport}} = \underbrace{-\frac{1}{\tau}(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t))}_{\text{Collision}},$$

- where *i* goes over all lattice directions.

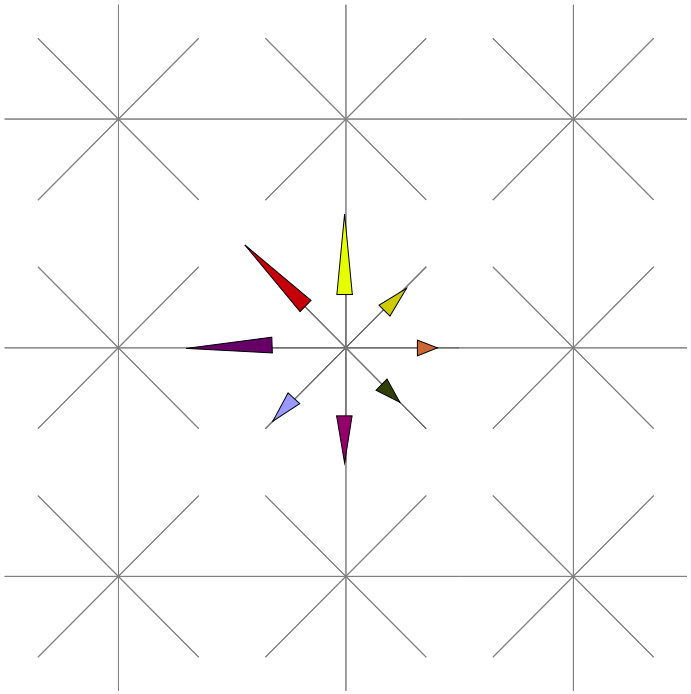
Transport step

- $t=0$



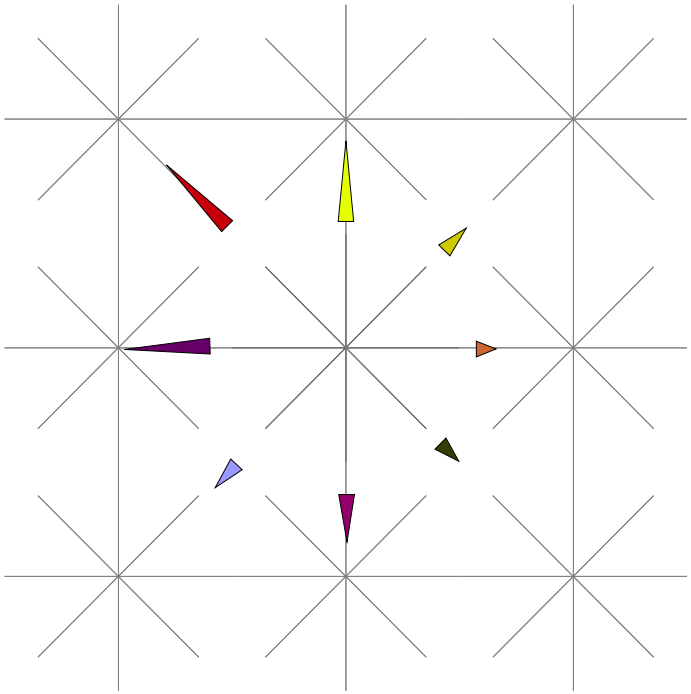
Transport step

- *t=1/4*



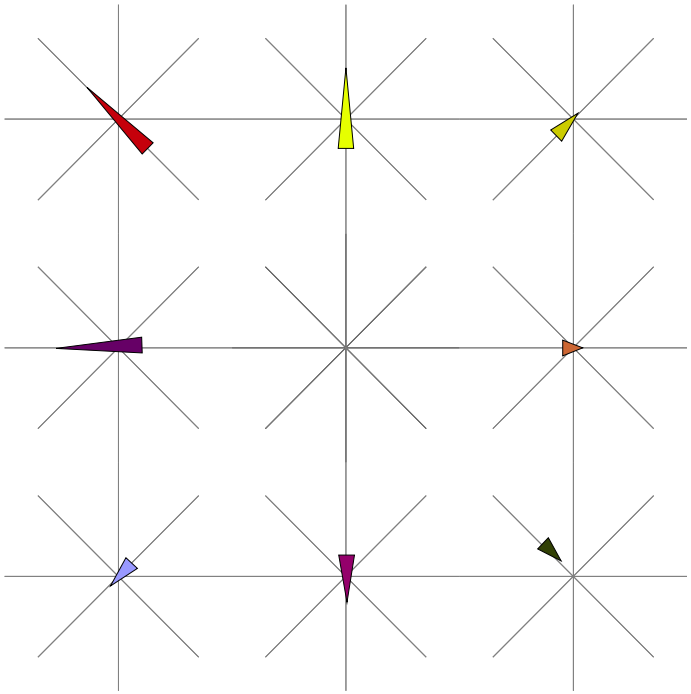
Transport step

- *t=1/2*



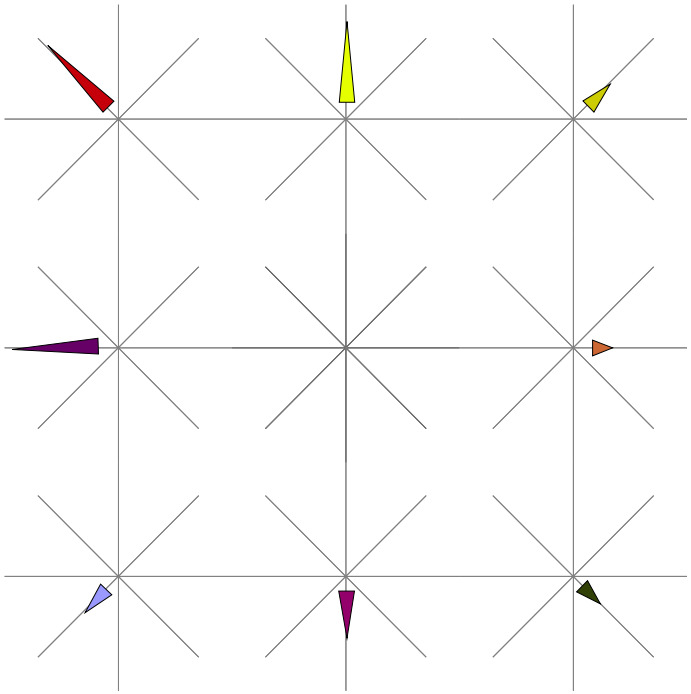
Transport step

- *t=3/4*



Transport step

- *t=1*



Simulation model

- **Collision** (Relaxation towards equilibrium)
- **Transport** (The Boltzmann Equation)

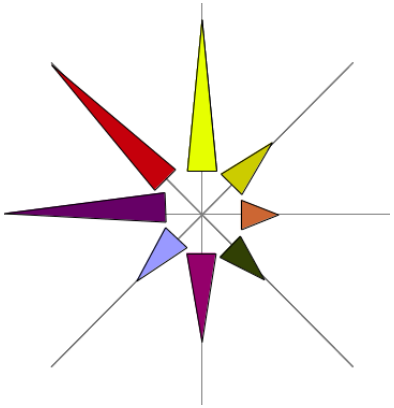
Implementation

Preliminary code

Data structures.
Computation of density and velocity.

Example Implementation

- Density function

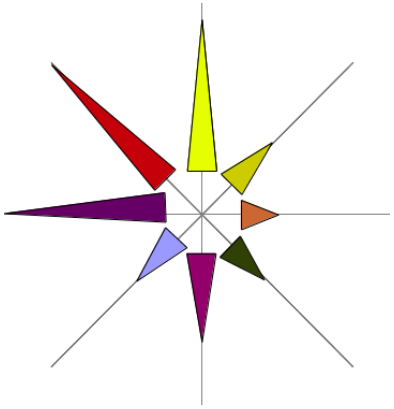


Example Implementation

- Density function

```
float  df[2][L*L][9];
```

Grid 2x
copy



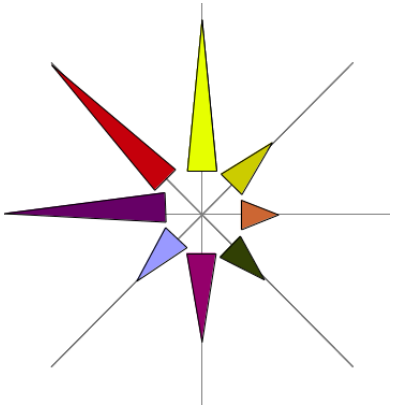
Example Implementation

- Density function

```
float  df[2][L*L][9];
```

Grid 2x copy

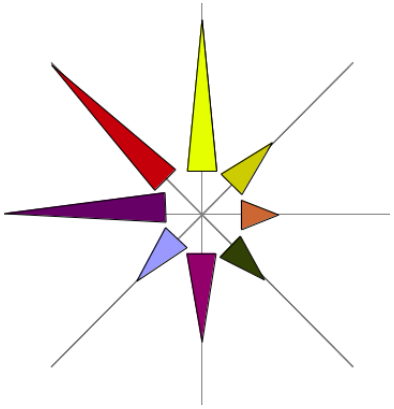
Domain size



Example Implementation

- Density function

```
float  df[2][L*L][9];
```



Grid 2x copy

Domain size

Directions

Implementation

Preliminary code

Data structures.
Computation of density and velocity.

Macroscopic variables

- Density: $$\varrho = \sum_{i=1}^{9} f_i$$

```
float rho=0;
for(int i=0; i<9; i++)
{
  rho = rho   + df[c][ x+y*L ][i];


}
```

Macroscopic variables

- Density:

- Velocity: $$\rho \mathbf{u} = \sum_{i=1}^{9} f_i \mathbf{e}_i$$

```
float rho=0,ux=0,uy=0;
for(int i=0; i<9; i++)
{
  rho = rho   + df[c][ x+y*L ][i];
  ux =  ux    + df[c][ x+y*L ][i] * ex[i];
  uy =  uy    + df[c][ x+y*L ][i] * ey[i];
}

ux /= rho;
uy /= rho;
```

LBM Algorithm

- Collision (BGK relaxation):

$$\tilde{f}_i(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{1}{\tau}(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t))$$

- Transport:

$$f_i(\mathbf{x}+\mathbf{c}_i, t+1) = \tilde{f}_i(\mathbf{x},t).$$

From Maxwell-Boltzmann distribution

Implementation

Part 1

Collision term

LBM Algorithm

- Collision (BGK relaxation):

$$\tilde{f}_i(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{1}{\tau}(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t))$$

From Maxwell-Boltzmann distribution

Equilibrium distribution function

- Equilibrium distribution from quadratic expansion of the Maxwell-Boltzmann distribution [1]:

$$f_i^{eq} = \varrho w_i \left[1 + \frac{\mathbf{u} \cdot \mathbf{c}_i}{c_s^2} + \frac{\mathbf{u}\mathbf{u} \cdot (\mathbf{c}_i \mathbf{c}_i) - c_s^2 \mathbf{I}}{2 c_s^4} \right]$$

- ϱ Fluid density
- c_s Lattice sound speed ($c_s^2 = \sum_i w_i c_i^2$)
- w_i Lattice direction weights
- $\mathbf{I}$ Unit matrix

[1] S. Succi, O. Filippova, G. Smith, E. Kaxiras *Applying the Lattice Boltzmann Equation to Multiscale Fluid Problems*, Comp. Sci. Eng., Nov-Dec 2001, 26–36
[2] Viggen, E. M., *The Lattice Boltzmann Method with Applications in Acoustics*, MSc, Department of Physics NTNU (Norway)

D2Q9 Model (various sources)

- **Lattice weights:**
- wi = 4/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, 1/36

- **Lattice vectors:**
- Ci = (0,0), (1,0), (0,1), (-1,0), (0,-1), (1,1), (-1,1), (-1,-1), (1,-1)

- **Sound speed:**

$$c_s^2 = \sum_i w_i c_i^2 =$$

1/9+1/9+1/9+1/9+2/36*4=4/9+8/36=6/9 = **1/3**

More details:

http://www.physics.buffalo.edu/phy411-506/2010/topic4/lec-4-5.pdf

Equilibrium distribution function

$$f_i^{eq} = \varrho w_i \left[1 + \frac{\mathbf{u} \cdot \mathbf{c}_i}{c_s^2} + \frac{\mathbf{u}\mathbf{u} \cdot (\mathbf{c}_i \mathbf{c}_i) - c_s^2 \mathbf{I}}{2 c_s^4} \right]$$

$$\downarrow$$

$$\boldsymbol{f_i^{eq}(\vec{x}) = w_i \rho(\vec{x}) \left[1 + 3 \frac{\vec{e}_i \cdot \vec{u}}{c^2} + \frac{9}{2} \frac{(\vec{e}_i \cdot \vec{u})^2}{c^4} - \frac{3}{2} \frac{\vec{u}^2}{c^2} \right]}$$

http://www.dis.uniroma1.it/~pellegrini/project-assignment/cpp/pdf/boltzmann-lattice-2.pdf

Collision step - Implementation

- Collisions in LBM:

$$f_i^{eq}(\vec{x}) = w_i \rho(\vec{x}) \left[1 + 3\frac{\vec{e_i} \cdot \vec{u}}{c^2} + \frac{9}{2}\frac{(\vec{e_i} \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}^2}{c^2} \right]$$

Collision step - Implementation

- Collisions in LBM:

$$f_i^{eq}(\vec{x}) = w_i \rho(\vec{x}) \left[1 + 3\frac{\vec{e}_i \cdot \vec{u}}{c^2} + \frac{9}{2}\frac{(\vec{e}_i \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}^2}{c^2} \right]$$

```
(... loop over all x/y in the lattice...)

for(i=0; i< 9; i++)
{




}
```

Collision step - Implementation

- Collisions in LBM:

$$f_i^{eq}(\vec{x}) = w_i \rho(\vec{x}) \left[1 + 3\frac{\vec{e_i} \cdot \vec{u}}{c^2} + \frac{9}{2}\frac{(\vec{e_i} \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}^2}{c^2} \right]$$

$c = 1$

```
(... loop over all x/y in the lattice...)

for(i=0; i< 9; i++)
{
      feq =  w[i] * rho * (1.0f + 3.0f * (ex[i] * ux + ey[i]*uy)
             + (9.0f/2.0f)*(ex[i]*ux +ey[i]*uy)*(ex[i]*ux+ey[i]*uy)
             - (3.0f/2.0f) *  (ux*ux + uy*uy));



}
```

Collision step - Implementation

- Collisions in LBM:

$$f_i^{eq}(\vec{x}) = w_i \rho(\vec{x}) \left[1 + 3\frac{\vec{e_i} \cdot \vec{u}}{c^2} + \frac{9}{2}\frac{(\vec{e_i} \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}^2}{c^2} \right]$$

$c = 1$

```
(... loop over all x/y in the lattice...)

for(i=0; i< 9; i++)
{
     feq =  w[i] * rho * (1.0f + 3.0f * (ex[i] * ux + ey[i]*uy)
            + (9.0f/2.0f)*(ex[i]*ux +ey[i]*uy)*(ex[i]*ux+ey[i]*uy)
            - (3.0f/2.0f) *  (ux*ux + uy*uy));



}
```

$$\tilde{f}_i(\mathbf{x}, t) = f_i(\mathbf{x}, t) - \frac{1}{\tau}(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t))$$

Collision step - Implementation

- Collisions in LBM:

$$f_i^{eq}(\vec{x}) = w_i \rho(\vec{x}) \left[1 + 3\frac{\vec{e_i}\cdot\vec{u}}{c^2} + \frac{9}{2}\frac{(\vec{e_i}\cdot\vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}^2}{c^2} \right]$$

```
(... loop over all x/y in the lattice...)

for(i=0; i< 9; i++)
{
     feq =  w[i] * rho * (1.0f + 3.0f * (ex[i] * ux + ey[i]*uy)
            + (9.0f/2.0f)*(ex[i]*ux +ey[i]*uy)*(ex[i]*ux+ey[i]*uy)
            - (3.0f/2.0f) *  (ux*ux + uy*uy));

   df[c][x+y*L][i]=df[c][x+y*L][i] - (1/tau)*(df[c][x+y*L][i]-feq);

}
```

$$\tilde{f}_i(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{1}{\tau}(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t))$$

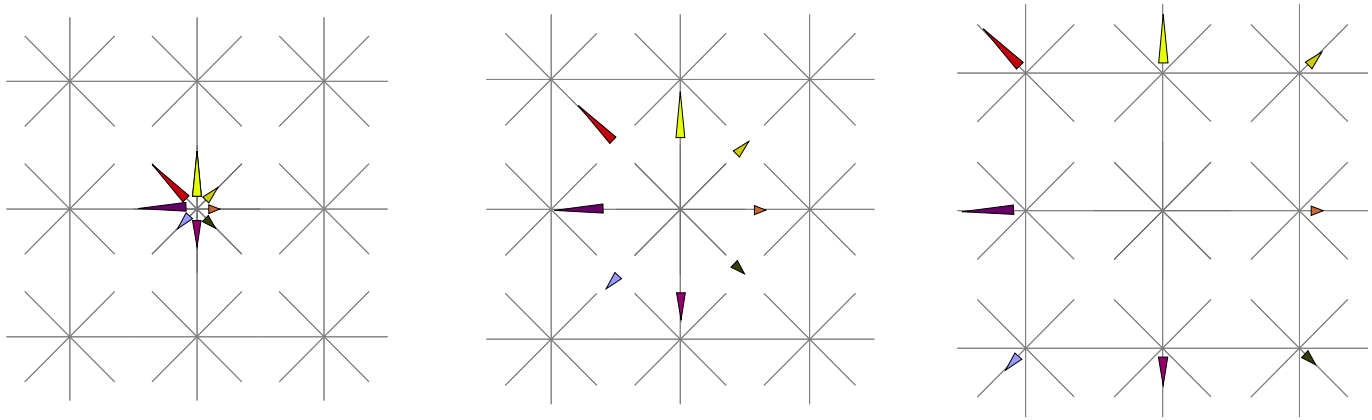
Implementation

Part 2

Transport of density function

Transport step

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = \tilde{f}_i(\mathbf{x}, t).$$



t=0 t=1/2

t=1

Streaming

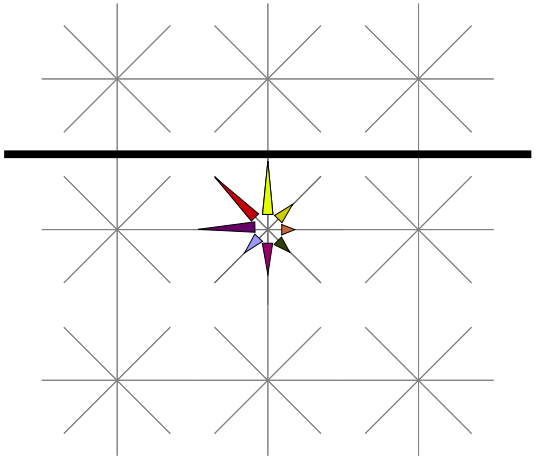
```
for(int x=0 ; x < L ; x++)
for(int y=0 ; y < L ; y++)
{
     for(int i=0; i< 9; i++)
     {

         int xp = ( x+ex[i] + L ) % (L);
         int yp = ( y+ey[i] + L ) % (L);



         df[1-c][ xp + yp*L ][i] = df[c][ x+y*L ][i];
     }
 }
```

Neighbour in direction e_i

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = \tilde{f}_i(\mathbf{x}, t).$$

No slip condition

- Bounce-back on a solid node

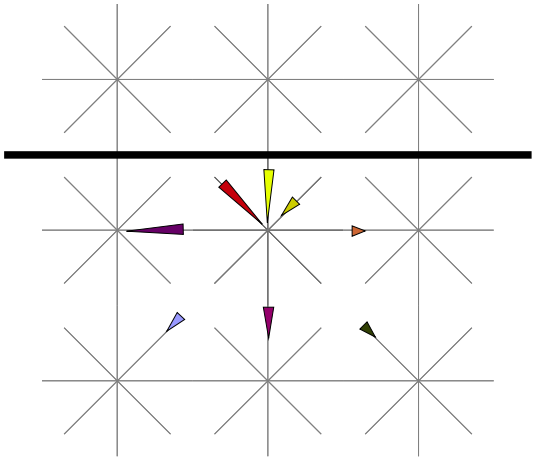


- 2nd order accuracy (mid-grid method)

Succi, Sauro (2001). The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press

No slip condition

- Bounce-back on a solid node

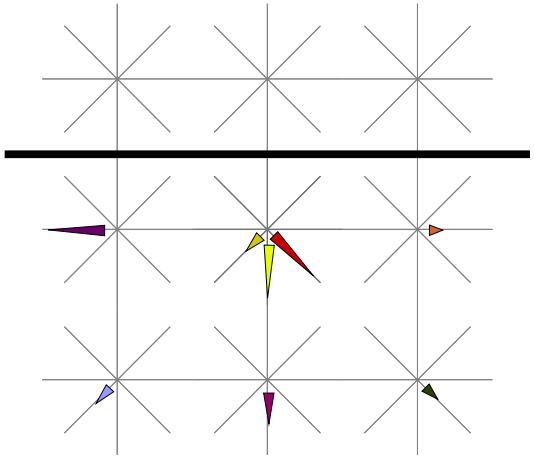


- 2nd order accuracy (mid-grid method)

Succi, Sauro (2001). The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press

No slip condition

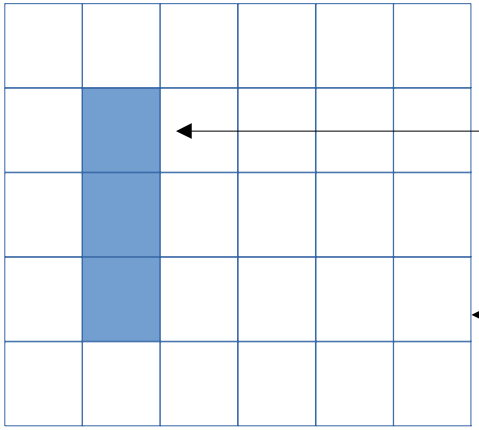
- Bounce-back on a solid node



- 2nd order accuracy (mid-grid method)

Succi, Sauro (2001). The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press

Boolean table for solid/fluid nodes.



Solid node → **FLAG**[ ] = 1

Fluid node → **FLAG**[ ] = 0

Streaming

```
for(int x=0 ; x < L ; x++)
for(int y=0 ; y < L ; y++)
if(FLAG[ x+y*L ] == 0)
{
     for(int i=0; i< 9; i++)
     {

         int xp = ( x+ex[i] + L ) % (L);
         int yp = ( y+ey[i] + L ) % (L);

          if( FLAG[ xp + yp*L ] == 1 )
              df[1-c][ x+y*L ][inv[i]] = df[c][ x+y*L ][i];
         else
              df[1-c][ xp+yp*L][i] = df[c][ x+y*L ][i];
     }
 }
```

Mid-grid Bounce back

Full LBM code

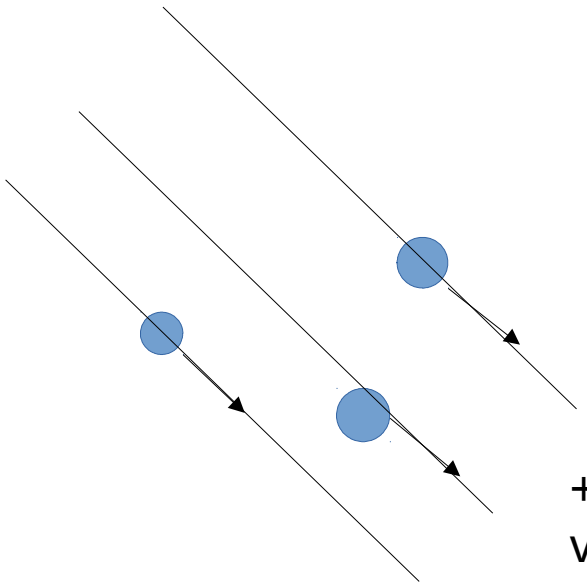
```
for(int i=0 ; i < L ; i++)
for(int j=0 ; j < L ; j++)
{
  idx =  i+j*L
  U[idx]=V[idx]=R[idx]=0;
  for(int k=0; k<9; k++)
  {
      tmp = df[ c ][ idx ][ k ];
      R[ idx ] = R[ idx ] + tmp;
      U[ idx ] = U[ idx ] + tmp * ex[ k ];
      V[ idx ] = V[ idx ] + tmp * ey[ k ];
  }
  U[idx] = U[idx]/R[idx] + fx;
  V[idx] = V[idx]/R[idx];
// transport + collision code here...
}
```

// velocity

// transport + collision

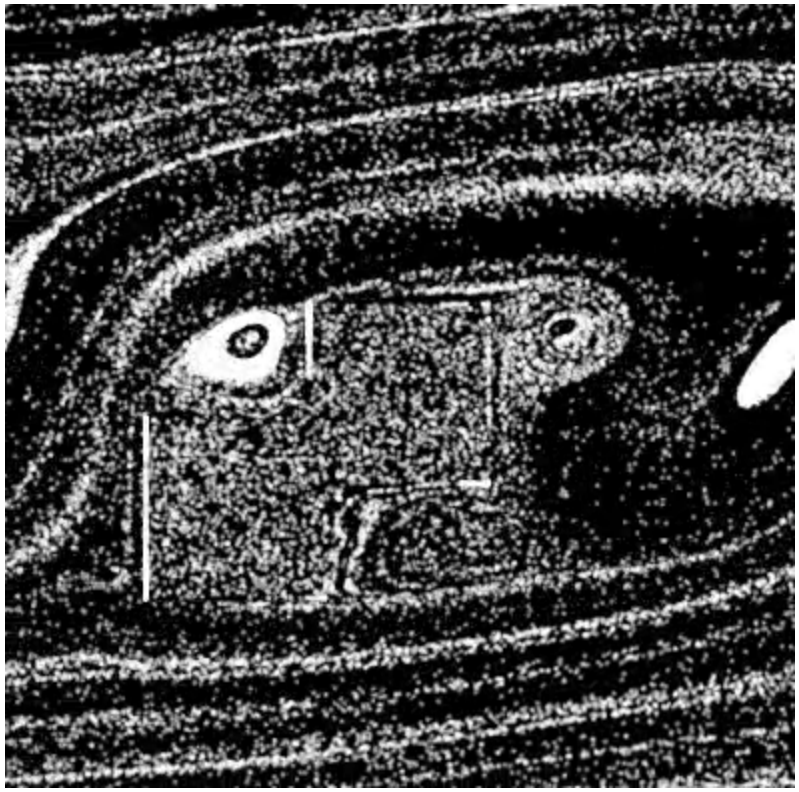
```
for(int k=0; k<9; k++)
{
    int ip = ( i+ex[k] + L ) % (L);
    int jp = ( j+ey[k] + L ) % (L);
    tmp = ex[k]*U[idx] + ey[k]*V[idx];
    feq = w[k] * rho * (1 – 1.5 * (U[idx]*U[idx]+V[idx]*V[idx]) + 3*tmp + 4.5 *tmp*tmp);
    if( FLAG[ip+jp*L] == 1 )
        df [1-c][idx][inv[k]] = (1-omega) * df[c][idx][k] + omega*feq
    else
        df [1-c][ip+jp*L][k] = (1-omega) * df[c][idx][k] + omega*feq;
}
```

Full LBM code



+ particles on top of the velocity
field

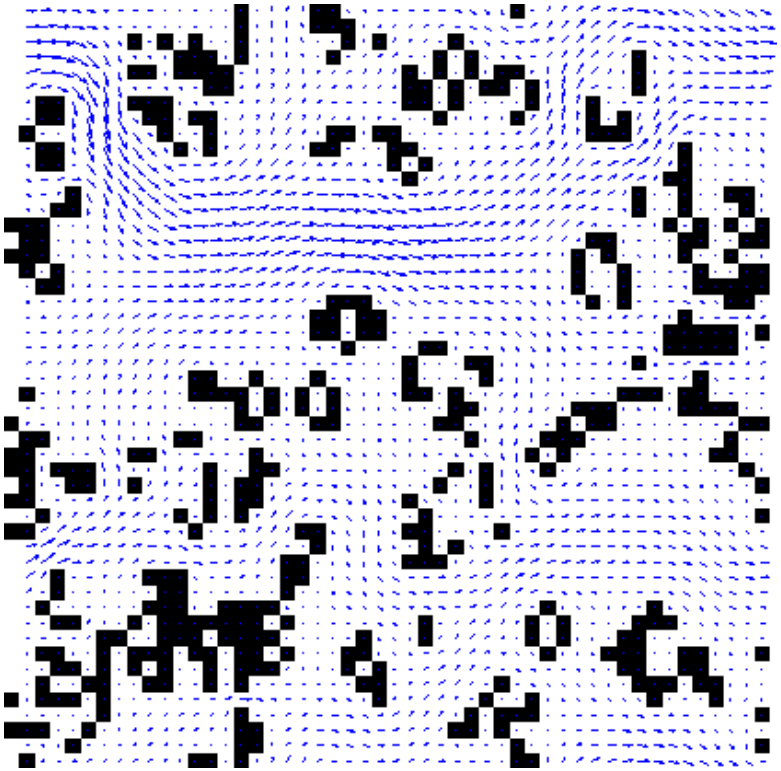
LBM in action



(lbm11.mkv)

Single page LBM implementation

```
% 2D Lattice Boltzmann (BGK) model of a fluid.
%  c4  c3   c2  D2Q9 model. At each timestep, particle densities propagate
%    \  |  /    outwards in the directions indicated in the figure. An
%  c5 -c9 - c1  equivalent 'equilibrium' density is found, and the densitie
%    /  |  \    relax towards that state, in a proportion governed by omega
%  c6  c7   c8      Iain Haslam, March 2006.
omega=1.0; density=1.0; t1=4/9; t2=1/9; t3=1/36; c_squ=1/3; nx=31; ny=31;
F=repmat(density/9,[nx ny 9]); FEQ=F; msize=nx*ny; CI=[0:msize:msize*7];
%a=repmat(-15:15,[31,1]);BOUND=(a.^2+a'.^2)<16;BOUND(1:nx,[1 ny])=1;
%BOUND=zeros(nx,ny);BOUND(1:nx,1)=1;%open channel
BOUND=rand(nx,ny)>0.7; %extremely porous random domain
ON=find(BOUND); %matrix offset of each Occupied Node
TO_REFLECT=[ON+CI(1) ON+CI(2) ON+CI(3) ON+CI(4) ...
            ON+CI(5) ON+CI(6) ON+CI(7) ON+CI(8)];
REFLECTED= [ON+CI(5) ON+CI(6) ON+CI(7) ON+CI(8) ...
            ON+CI(1) ON+CI(2) ON+CI(3) ON+CI(4)];
avu=1; prevavu=1; ts=0; deltaU=1e-7; numactivenodes=sum(sum(1-BOUND));
while (ts<4000 & 1e-10<abs((prevavu-avu)/avu)) | ts<100
    % Propagate
    F(:,:,4)=F([2:nx 1],[ny 1:ny-1],4);F(:,:,3)=F(:,[ny 1:ny-1],3);
    F(:,:,2)=F([nx 1:nx-1],[ny 1:ny-1],2);F(:,:,5)=F([2:nx 1],:,5);
    F(:,:,1)=F([nx 1:nx-1],:,1);F(:,:,6)=F([2:nx 1],[2:ny 1],6);
    F(:,:,7)=F(:,[2:ny 1],7); F(:,:,8)=F([nx 1:nx-1],[2:ny 1],8);
    BOUNCEDBACK=F(TO_REFLECT); %Densities bouncing back at next timestep
    DENSITY=sum(F,3);
    UX=(sum(F(:,:,[1 2 8]),3)-sum(F(:,:,[4 5 6]),3))./DENSITY;
    UY=(sum(F(:,:,[2 3 4]),3)-sum(F(:,:,[6 7 8]),3))./DENSITY;
    UX(1,1:ny)=UX(1,1:ny)+deltaU; %Increase inlet pressure
    UX(ON)=0; UY(ON)=0; DENSITY(ON)=0;
    U_SQU=UX.^2+UY.^2; U_C2=UX+UY; U_C4=-UX+UY; U_C6=-U_C2; U_C8=-U_C4;
    % Calculate equilibrium distribution: stationary
    FEQ(:,:,9)=t1*DENSITY.*(1-U_SQU/(2*c_squ));
    % nearest-neighbours
    FEQ(:,:,1)=t2*DENSITY.*(1+UX/c_squ+0.5*(UX/c_squ).^2-U_SQU/(2*c_squ));
    FEQ(:,:,3)=t2*DENSITY.*(1+UY/c_squ+0.5*(UY/c_squ).^2-U_SQU/(2*c_squ));
    FEQ(:,:,5)=t2*DENSITY.*(1-UX/c_squ+0.5*(UX/c_squ).^2-U_SQU/(2*c_squ));
    FEQ(:,:,7)=t2*DENSITY.*(1-UY/c_squ+0.5*(UY/c_squ).^2-U_SQU/(2*c_squ));
    % next-nearest neighbours
    FEQ(:,:,2)=t3*DENSITY.*(1+U_C2/c_squ+0.5*(U_C2/c_squ).^2-U_SQU/(2*c_squ
    FEQ(:,:,4)=t3*DENSITY.*(1+U_C4/c_squ+0.5*(U_C4/c_squ).^2-U_SQU/(2*c_squ
    FEQ(:,:,6)=t3*DENSITY.*(1+U_C6/c_squ+0.5*(U_C6/c_squ).^2-U_SQU/(2*c_squ
    FEQ(:,:,8)=t3*DENSITY.*(1+U_C8/c_squ+0.5*(U_C8/c_squ).^2-U_SQU/(2*c_squ
    F=omega*FEQ+(1-omega)*F;
    F(REFLECTED)=BOUNCEDBACK;
    prevavu=avu;avu=sum(sum(UX))/numactivenodes; ts=ts+1;
end
figure;colormap(gray(2));image(2-BOUND');hold on;
quiver(2:nx,1:ny,UX(2:nx,:)',UY(2:nx,:)');
title(['Flow field after ',num2str(ts),'\deltat']);xlabel('x');ylabel('y');
Flow through random grid of cells
```



http://exolete.com/lbm/

Next lecture

Next Week: continue on Lattice Boltzmann Method

- LBM review

- Extending the model (viscosity, 3d grids, etc.)

- Multi-relaxation time LBM

- Zou/He Pressure boundary conditions

- Multiphase LBM

- Immersed boundary method