An Introduction to the Lattice Boltzmann Method

Soham Mehta

Indian Institute of Technology - Bombay

Tutors :

Prof. V. Buwa, Prof. Rüde, C. Feichtinger

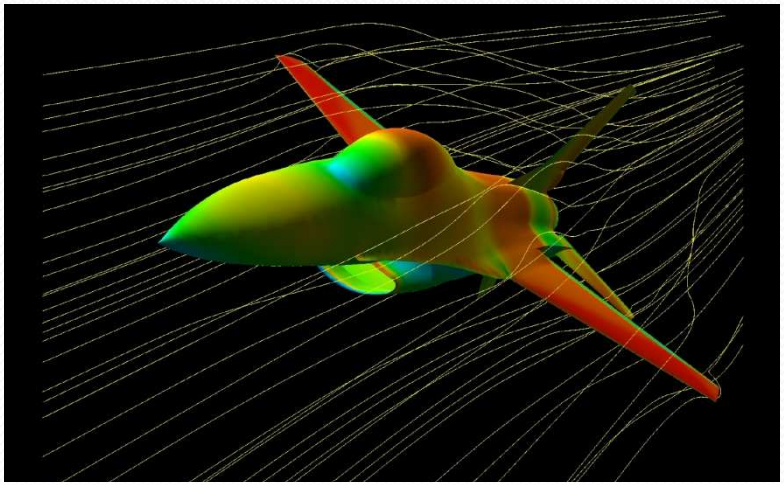
Indo-German Winter Academy 2009

What we'll be looking at

- Motivation for Computational Fluid Dynamics and specifically the Lattice Boltzmann Method
- Theoretical background
- Discretisation
- The LBM algorithm with Demonstration
- Advanced concepts
- Comparison with other methods

Why Simulate Fluids?

- Simulation of fluid flows in pipes
- Air flows for aircraft testing
- Weather Study
- Games and Graphics
- Chemical Reactions
- Medicine



Motivation and History

- During the last ten years the Lattice Boltzmann Equation (LBE) method has been developed as an alternative numerical approach in computational fluid dynamics (CFD).
- Originated from the discrete kinetic theory, the LBE method has emerged with the promise to become a superior modeling platform, both computationally and conceptually, compared to the existing arsenal of the continuum-based CFD methods.
- The LBE method has been applied for simulation of various kinds of fluid flows under different conditions. The number of papers on the LBE method and its applications continues to grow rapidly, especially in the direction of complex and multiphase media.

Quantities involved

- f(x, v, t) - the fluid particle distribution function; related to the probability to encounter particles with the continuous, microscopic velocity v at position x at time t.
- f^0 - equilibrium distribution function
- v - microscopic fluid velocity (discrete, in fixed directions)
- u - macroscopic (average) velocity
- p - fluid pressure
- ϑ - Kinematic viscosity
- ρ - fluid density n - local particle density
- Δx - physical cell size Δt - physical time step
- c - speed of 'sound', actually speed of information
- * denotes dimensionless, normalized quantity

Classical CFD

- Navier-Stokes Equation, essentially local conservation of momentum

$$\frac{\partial}{\partial t}(\rho \boldsymbol{v}) + \nabla(\rho \boldsymbol{v} \cdot \boldsymbol{v}) = -\nabla p + \nabla \sigma$$

- σ, the local viscous stress tensor, is the 'external force' component.
- Mass conservation – continuity equation

$$\frac{\partial n}{\partial t} + n(\nabla \cdot \boldsymbol{v}) = 0$$

- Energy conservation – heat conduction equation

  (for problems involving temperature variation)

Boltzmann Theory

- Minimise H functional (Entropy)

$$H(t) = \int f(x,u,t)\log\left(f(x,u,t)\right)dx\,du$$

- Equilibrium distribution function satisfies

$$\frac{\partial}{\partial t}H(t) = 0 \;\; at\, f = f^0$$

- f^0 is for a given energy $E = n\left(\frac{1}{2}mu^2 + \frac{3}{2}kT\right)$

- One can solve the above equations to obtain

$$f^0 = \frac{n}{(2\pi kT)^{3/2}}\exp\left(\frac{-(v-u)^2}{2kT}\right)$$

Boltzmann Equation

- Boltzmann Equation in hydrodynamics is:

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla f + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f = Q(\boldsymbol{u}, \boldsymbol{t})$$

- For simplicity, we consider the collision integral Q(u) for binary collisions only. The operator Q(f,f) represents change of the distr. function due to binary collisions between particles.

$$Q(f,f)(v) = \iint B(v_1, v_2, e)\left(f(v_1')f(v_2') - f(v_1)f(v_2)\right) de\, dv_2$$

- Here (v_1, v_2) and (v_1', v_2') are the pre-collision and post-collision velocities, e is a unit vector.
- $B(v_1, v_2, e) = |v|\sigma(|v|, (\hat{v} \cdot e))$ is the collision kernel with the cross- section σ and $v = |v_1 - v_2|$

The BGK Approximation

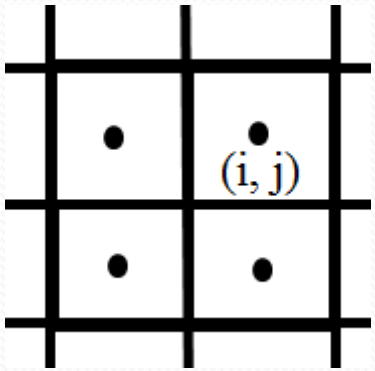
- The Bhatnagar-Gross-Kook(BGK) Approximation - “effect of the collision term is to bring the velocity distr. function closer to the equilibrium distribution”

$$Q(u,t) = -\frac{1}{\tau}(f - f^0)$$

- τ is the relaxation time, related to the density. This is a ‘Single Relaxation Time’ approximation.
- Boltzmann’s collision operator has the fundamental properties of conserving mass, momentum and energy. On Integrating the Boltzmann Equation with this approximation, after multiplying with 1, v, v^2 ,one obtains the mass, momentum and energy conservation conditions.

Discretisation

- Aim is to solve for ρ and u in discrete 'cells'.
- Typical mathematical techniques to solve differential equations – discrete approximation is used.

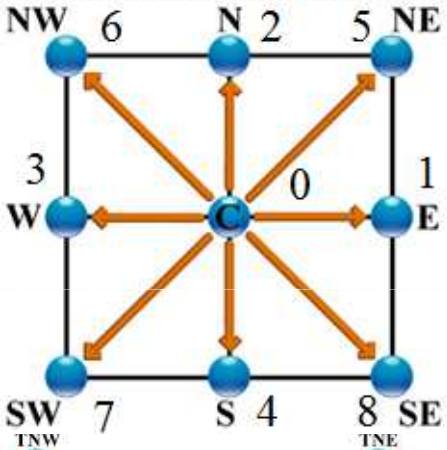


- Most of the discrete operators will only conserve mass and momentum up to the order of discretisation.
- A first order approximation to the Taylor expansion of the Boltzmann Equation is employed. With some transformations, the Boltzmann equation is linearised with discretisation of velocity into fixed number of directions – particles can now only move along these directions.

Lattice Models

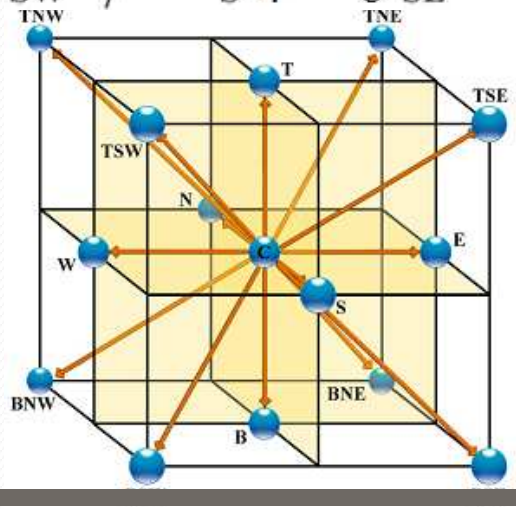
- Weights depend on the set of velocities used.

- D2Q9



i	W_i	V_i
0	4/9	(0,0)
1-4	1/9	(±1, 0)c (0, ±1)c
5-8	1/36	$(\pm1, \pm1)2^{1/2}$ c

- D3Q15



Discrete Boltzmann equation

- phase space (6D) = position space (3D) + impulse space (3D)
- Discrete Boltzmann Equation is :

$$f_i(x + v_i\Delta t,\ v_i, t + \Delta t) - f_i(x,\ v_i, t) + F(\ v_i) = \frac{1}{\tau}[f_i^0(n, \boldsymbol{u}, T) - f(x,\ v_i, t)]$$

- Where $F(v_i) \triangleq \boldsymbol{F} \cdot \nabla_{\mathbf{v}_i} f_i$

- Note that $(x+v_i\Delta t)$ is also a valid position in the physical lattice.
- The discrete equilibrium distribution function is given by

$$f_i^0 = nW_i\left(1 + \frac{3}{c^2}\boldsymbol{u}\cdot\boldsymbol{v}_i + \frac{9}{2c^4}(\boldsymbol{u}\cdot\boldsymbol{v}_i)^2 - \frac{3}{2c^2}\boldsymbol{u}\cdot\boldsymbol{u}\right)$$

Step Size and Stability

- Normalizing real world variable to non-dimensional simulation parameters

$$\Delta x \rightarrow \Delta x^* = 1, \qquad \Delta t \rightarrow \Delta t^* = 1$$

$$u \rightarrow u^* = u\frac{\Delta x}{\Delta t} \qquad \vartheta \rightarrow \vartheta^* = \vartheta\frac{\Delta t}{(\Delta x)^2} \qquad \tau \rightarrow \tau^* = \frac{\tau}{\Delta t}$$

- It can be shown that the kinematic viscosity is expressible as

  $\vartheta = c^2\Delta t\left(\tau^* - \frac{1}{2}\right)$, this requires $\tau^* > 0.5$, generally < 0.7 also.

- The LB method under consideration is valid only in the incompressible limit. Convergence of the solution to the incompressible Navier-Stokes equations for a fixed Reynolds number $R_e = Lu/\vartheta$ is obtained by letting the Mach number (~ u) become small enough to remove compressibility effects, and letting the lattice spacing $v_i\Delta t$(~ τ^*) become small enough to 'resolve' the flow.

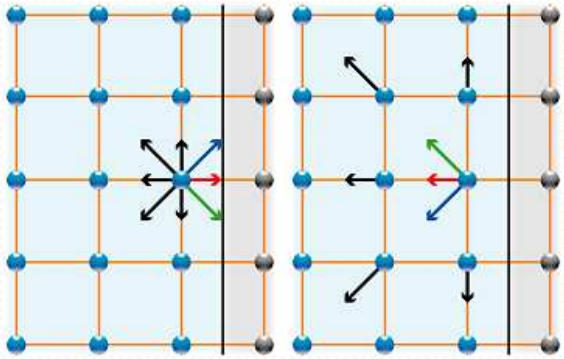
Fluids at solid boundaries

- The no-slip boundary conditions are used for walls, where (due to friction) a fluid velocity of zero is assumed.
- For stationary boundaries, this behavior can be achieved by reflecting the distribution functions in the opposite direction,

$$f(x, v_i, t+1) = f(x, -v_i, t)$$

No slip condition

(at t) (at t+1)



Moving Boundaries

- Galilean-transform the distribution into the rest frame of the boundary, then perform the bounce-back operation, and then transform the flow equations back into the original frame of reference.
- If U is the velocity of the moving boundary,

$$f^*(\boldsymbol{v}_i) = f(\boldsymbol{v}_i) + G(\boldsymbol{v}_i, \boldsymbol{U})$$

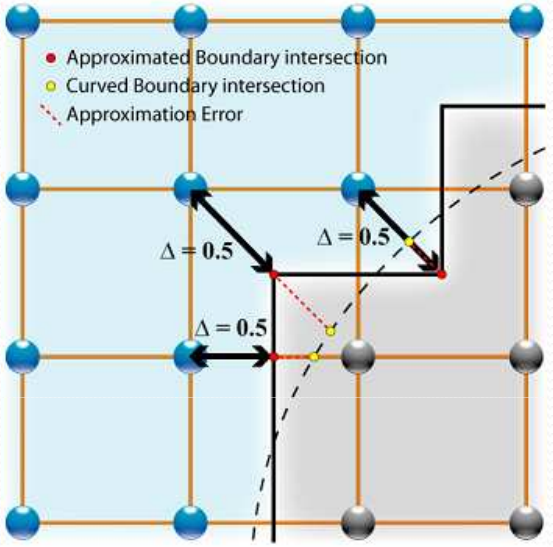
$$f(\boldsymbol{v}_i, t+1) = f_{-i}{}^*(\boldsymbol{v}_i) + G(-\boldsymbol{v}_i, -\boldsymbol{U}) = f_{-i}(\boldsymbol{v}_i) + G(\boldsymbol{v}_i, \boldsymbol{U}) + G(-\boldsymbol{v}_i, -\boldsymbol{U})$$

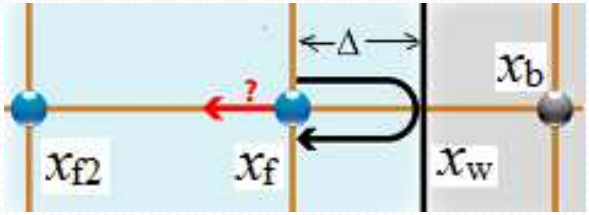
- The usual choice for G is such that

$$f_i(\boldsymbol{x}, t+1) = f_{-i}(\boldsymbol{x}, t) + \frac{6}{c^2}(w_i n \boldsymbol{U} \cdot \boldsymbol{v}_i)$$

Curved boundaries

- With boundaries parallel to the lattice cells, nodes are easily separated into ‘fluid cell’ or ‘obstacle cell’. Such an approximation to curved boundaries leads to large errors and instability.
- Direct treatment is done as below





$$f_i(\boldsymbol{x_f}, t+1) = \frac{(1-\Delta) f_{-i}(\boldsymbol{x_f}, t) + \Delta\left(f_{-i}(\boldsymbol{x_b}, t) + f_{-i}(\boldsymbol{x_{f2}}, t)\right)}{1+\Delta}$$

The Collide step

- Density $\rho = \sum_{i=1}^{N} f_i$ and momentum density $\rho \boldsymbol{u} = \sum_{i=1}^{N} f_i \boldsymbol{v}_i$
- The first step in the calculation of the macroscopic density and velocity in each cell. Using these, the equilibrium distributions are calculated, which are needed for the collision step.
- During the collision step, the distribution functions are relaxed towards the equilibrium state.

$$f_i(x, t+1) = f_i(x,t) - \frac{1}{\tau}\left(f_i(x,t) - f_i^0(x,t)\right)$$

The Stream step

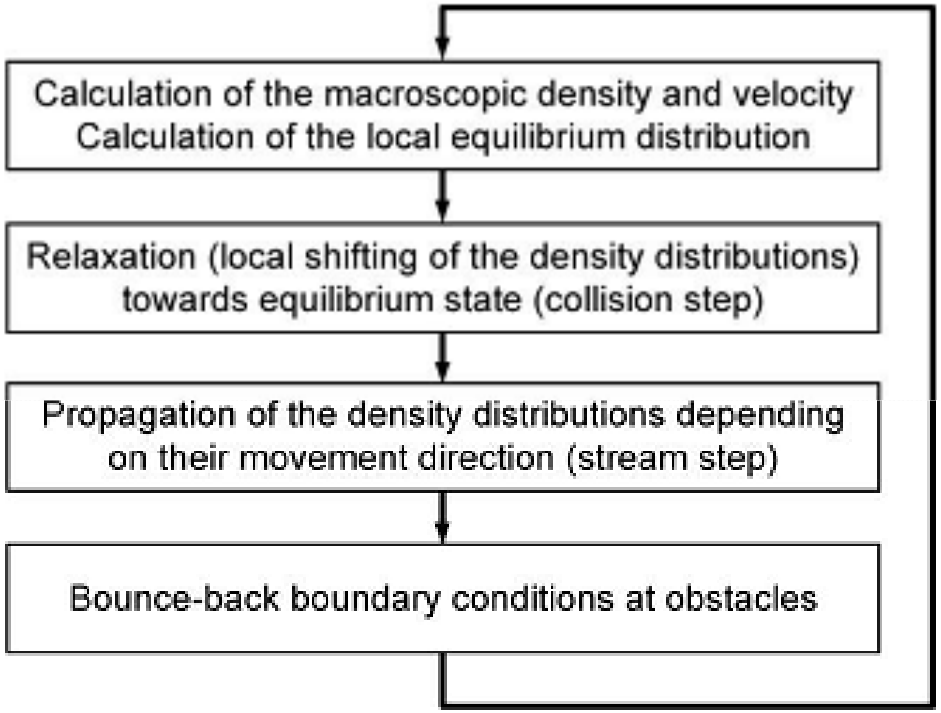
- In the streaming or propagation step, fluid particles are streamed from one cell to a neighboring cell according to the velocity of the fluid particles in this cell.

$$f_i(x + v_i, t + 1) = f_i(x, t + 1)$$

- Streaming can either be performed as a pushing operation from one cell to the surrounding cells or as pulling operation in one cell from the surrounding cells.

- Boundary conditions are also implemented with this step

Complete LBM Algorithm



- The LBM update order can be stream-collide or collide-stream. While the simulation itself does not change by switching the update order, it is interesting for performance considerations.

Demonstration

Simple 2-D LBM Fluid Simulation

Particles added to show advection (motion ONLY due to fluid)

Particle Advection and f^0

```
void ParticleAdvection::advect0(){
  for (int i=0; i<num_particles; i++){
       int x = particles[i][0];
       int y = particles[i][1];
       if (x <= 0 || x >= dimx-1 || y <= 0 || y >= dimy-1){…}
       particles[i][0] += 50*u[x][y][0];
       particles[i][1] += 50*u[x][y][1];
  }
}

double LatticeSite::fEq(int k, double rho, double u[2]){
  double u2 = u[0]*u[0] + u[1]*u[1];
  double vu = v[k][0]*u[0] + v[k][1]*u[1];
  return rho*w[k]*(1.0 + 3.0*vu + 4.5*vu*vu - 1.5*u2);
}
```

Other functions

```
void LatticeSite::computeRhoAndU(double& rho, double u[2]){
  rho = 0; u[0] = 0; u[1] = 0;
  for (int k=0; k<9; k++)   rho += f[k];
  for (int k=0; k<9; k++){
       u[0] += f[k]*v[k][0];
       u[1] += f[k]*v[k][1];
  }
  u[0] /= rho;        u[1] /= rho;
}
void mousefunc(int button, int state, int x, int y){
  if (button == GLUT_LEFT_BUTTON){
       u[0] = (x - oldx); u[1] = (oldy - y);
       m = sqrt(u[0]*u[0]+u[1]*u[1]);
       u[0] /= (1+2*m);   u[1] /= (1+2*m);}
  oldx = x; oldy = y;   ...
}
```

Coding the Stream and Collide Steps

```
void LatticeSite::collide(double& rho, double u[2]){
  if (type == Boundary){
       for (int k=0; k<9; k++)      tmp[k] = f[k];
       for (int k=0; k<9; k++)      f[k] = tmp[N-k]; ...}
  else if (type == Fluid){
       computeRhoAndU(rho, u);
       for (int k=0; k<9; k++)
       f[k] = f[k]*(1.0-omega)+ omega*fEq(k, rho, u);}
}
void LatticeBoltzmann::streamToNeighbors(int x, int y) {
  for (int k=0; k<9; k++) {
       int nx = x + LatticeSite::v[k][0];
       int ny = y + LatticeSite::v[k][1];
       if (nx>=0 && nx<dims[0] && ny>=0 && ny<dims[1])
            sites_[nx][ny].f[k] = sites[x][y].f[k]; }
}
```

Coding the algorithm

```
void LatticeBoltzmann::update(){
  for (int y=0; y<dims[1]; y++){
       for (int x=0; x<dims[0]; x++){
            sites[x][y].collide(rho[x][y], u[x][y]);
            streamToNeighbors(x, y);}
  }
}
void display() {
  for (int y=0; y<LBHEIGHT; y++) {
      for (int x=0; x<LBWIDTH; x++) {
          m = velocity[x][y][0]^2+velocity[x][y][1]^2;
          pallete.GetColor(m*50, r, g, b);}
          glBindTexture(GL_TEXTURE_2D, texnum);   ...//Texture
          glBegin(GL_QUADS);                      ...//Render
          glEnd();}
}
```

Beyond basics

The LB method can also be applied to complex situations:

- Multiphase fluids
- Free Surface and Bubbly flows
- Problems with temperature variation
- Microparticles in fluids
- Multi-component fluid systems
- Turblence and Turbulent Flow
- Blood Flow (medicine)

Thermo-hydrodynamics with LBM

Techniques to simulate thermohydrodynamics using LBM:

- To increase the number of velocities, and to include higher-order nonlinear terms (in flow velocity) in the equilibrium distribution function.
- To use an equilibrium distribution with variable temperature.
- To use two sets of distribution functions for particle number density and energy density which effectively doubles the number of discrete velocities.
- Passive scalar approach - temperature is treated as a passive scalar which is advected by the flow velocity but does not affect the flow fields. Two distribution functions: for the NS equation and the advection-diffusion equation satisfied by temperature and flow. Numerically, this is not very efficient.

Particle suspensions

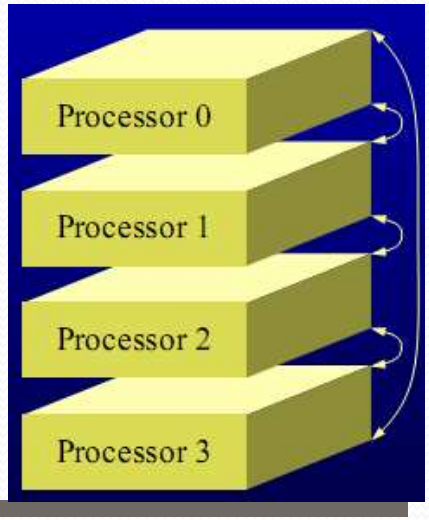
- Previously, we assumed that walls are unaffected by the fluid particles colliding with them. However, suspended particles have mass comparable to that of the fluid molecules.
- In the force step, the force F_p, applied by the fluid on the moving particles, is calculated. Forces exerted by the particles on the fluid have already been applied in the stream step.
- In order to calculate accurate forces on the moving particles, the momentum exchange method is used.

$$F_p = \sum_{all\ i}' \sum' v_i (f_i(x_f, t) + f_{-i}(x_b, t))$$

The outer sum is all over cells in the immediate vicinity of the particle, 'outside' it.

Reducing computational cost of the LBM

- Consider flow in porous media - Essentially a problem of complex boundary conditions. Extremely memory and processor intensive. Need to restructure memory usage.
- Idea: Store data only at 'active' sites, use circular buffer, eliminating need for 2nd data set.
- Saving of about 90% achieved for a 10% porosity.
- LBM has nearest-neighbor dependency, well suited to parallelization - At the end of each iteration, partial results from the bordering planes are exchanged between adj. blocks.



Why LBM over NS ?

- Whereas the NS equations are second-order PDEs, the DVM (Discrete Velocity Model) of the LBM is only first-order. This makes the discretization for the LBM much easier and simplifies the programming of the LBM approach.
- NS solvers inevitably need to treat the non-linear convective term, u.grad(u), which poses several numerical problems. In the LBM, the nonlinear convective term becomes simple advection.
- The LBM offers easier portability to modern computers of massively parallel processors. This makes the LBM especially interesting for industrial applications and supercomputer research.

Disadvantages of LBM

- In the LBM approach, the Courant-Friedrichs-Lewy (CFL) number (proportional to $\Delta x^*/\Delta t^*$), always equals 1. The consequence is a bad convergence for steady state problems, because the speed of convergence is dictated by acoustic propagation, which is very slow.

- The regular square grid is an obvious limitation of the LBM. The combination of high and low resolution areas or the application of curved grids, as needed for eg. in aerodynamic applications, is much more complicated in the LBM.

Summary and Conclusion

- We looked at the theoretical basis of NS and Lattice Gas Automata(LGA)-LBM approaches.
- We also saw, how the Boltzmann equation is discretised and implemented. We discussed the treatment of boundary conditions, and some complex situations which need some modifications to the basic approach.
- Overall, the Lattice Boltzmann method can be considered to be an efficient numerical method for simulation of fluid flow problems, and is used in a wide variety of situations. Ongoing research is extending the possibilities of the LBM implementation.

Bibliography

1. Alexander J. Wagner '**A Practical Introduction to the Lattice Boltzmann Method**'
2. Romana Begum, M. Abdul Basit '**Lattice Boltzmann Method and its Applications to Fluid Flow Problems**'
3. James D. Sterling, Shiyi Chen '**Stability Analysis of Lattice Boltzmann Methods**'
4. Klaus Iglberger '**Lattice Boltzmann Simulation of Flow around moving Particles**' (11)
5. Dazhi Yua, Renwei Mei, Li-Shi Luob, Wei Shyya '**Viscous flow computations with the method of lattice Boltzmann**'
6. John Hagedorn, Delphine Goujon, Nicos Martys, Judith Devaney '**A Parallel LB Algorithm for Multicomponent Fluid Flow in Complex Geometries**' (24)
7. Christoph P., '**Simulation und wissenschaftliches Rechnen**' (19)

Thank You