Abstract

The validity of the Lattice Boltzmann Model is assessed using some simple experiments. Flow through a channel and flow past a barrier at different Reynolds numbers are simulated. Flow through porous media is also investigated.
Acknowledgments

I would like to thank my project supervisor Stefan Hutzler for his help and direction during this project.

Also, my classmates who helped in many ways with useful comments and helpful advice.

Some friends deserve special mention:

Eoin Curran: for helping me with numerous tricky programming bits and suggesting many innovative approaches.

Michael Kerrin: for helping me sort out many of the tricky shell script problems encountered.

Anthony Brew: for offering many suggestions to some of the problems encountered.

Peter Stuart: did his 4th year project (2002) on the Lattice Gas Cellular Automata Model and came up with the extremely bright idea of concatenating the individual gnuplot images into a movie.
1 Introduction

Cellular automata are fully discrete models of physical and other systems. Lattice gases are just what the name says: a model of a gas on a grid. Thus lattice-gas cellular automata are a special kind of gas in which identical particles hop from site to site on a lattice at each tick of a clock. When particles meet they collide, but they always stay on the grid and appropriate physical quantities are always conserved.

This topic is interesting because it provides a new way of thinking about the simulation of fluids. It also provides an instructive link between the microscopic world of molecular dynamics and the macroscopic world of fluid mechanics.

Lattice gases bring something new to cellular automata: they are not only simple, but they are a realistic model of the equations of hydrodynamics. One would think that it would be impossible to model fluid mechanics by constructing a fictitious molecular dynamics lattice in which the molecules all move at the same speed and in only a limited number of directions. It turns out to be not only possible, however, but also the remarkable consequence of geometric symmetry and physical conservation laws.

Traditional computational fluid dynamics methods solve the Navier-Stokes equations, i.e. a set of second order partial differential equations, using approximations such as finite differences or finite elements and approximate numerical solvers. In lattice Boltzmann methods on the other hand, the hydrodynamic quantities are evaluated as phase space averages of the approximated single-particle distribution functions. It has been proven that this approach is a promising alternative to classical computational fluid dynamics methods which circumvents many deficiencies that are inherent in the existing methods. Potential fields of application include for example, aerodynamics, turbulence research, chemical engineering and combustion.
2 Theory

2.1 Cellular Automaton Methods

One of the first Cellular Automaton models was John von Neumann’s self-reproducing automaton which he constructed in 1948 in an attempt to explain biological reproduction. The model has many interesting features, including the fact that it was equivalent to a universal Turing machine. The most famous cellular automaton is probably Conway’s Game of Life.

2.1.1 Basic ingredients of cellular automaton models

Cellular automata model dynamical systems using discrete approximations including:

- Continuous space $x, y, z$ is replaced by a finite number of cells fixed in space, usually in a regular array or lattice.

- Continuous dynamical functions are also approximated by a discrete set of values at each cell site.

- Continuous time $t$ is made discrete.

- The dynamical equation of motion is replaced by a local rule: at each time step, cell values are given new values which depends on the cell values in a small local neighbourhood.

- The cell values are updated simultaneously or synchronously.

Cellular automata can be simulated very efficiently because cell variables have a finite number of values. Thus, integer arithmetic can be used.

2.2 Lattice Gas Cellular Automata model of fluid flow

This effective model of 2-D Navier-Stokes equations was published by U. Frisch, B. Hasslacher and Y. Pomeay [5]. (FHP model). Their model has the following properties:

- They found that a lattice with hexagonal symmetry was required to include a sufficient degree of rotational symmetry necessary
Figure 1: The total effect of the microscopic movement of particles is the pressure and temperature, which can be measured in the microscopic view.

for the conservation of angular momentum in a continuous fluid.

• To simulate variable density, each cell can have up to six fluid particles of mass \( m = 1 \). Thus the fluid density, \( \rho \), can take 7 different values.

• The free-streaming Euler properties of the model are implemented by the rules
  – Fluid particles have one of 6 possible velocities. The velocities are such that the particle moves to one of six neighbouring cells in one time step.
  – The velocities of particles in a particular cell must be distinct.

• The viscous (Navier-Stokes) properties are implemented by a set of collision rules. If there are 2, 3 or 4 particles at a site after the free-streaming rule has been applied, the particles are made to collide and change their velocities according to rules which conserve momentum (an \( X \) represents a particle with velocity in that particular direction):
In the first and fourth rules, there are two equivalent choices. One possibility is to select them at random: however this would make the automata probabilistic or stochastic. To keep the system deterministic the three choices in each case can be cycled in some order: choice of a particular order breaks handedness or chiral symmetry.

The Lattice Gas Cellular Automata model of Frisch, Hasslacher and Pomeau was much more efficient than standard CFD methods (finite grid or finite element) for simulating the Navier-Stokes equations. The main reasons for this efficiency is the use of integer arithmetic instead of floating point operations, and the fact that the collision rule is completely local. Fluid particles are moved to a neighbour site depending on their velocity. They then scatter according to collision rules which can be implemented as a simple look-up table.

However, it was found that the method suffered from significant problems. The resolution of these problems led to the introduction of Lattice Boltzmann methods.

2.3 Problems with Lattice Gas Cellular Automata

• Not Galilean Invariant. A Galilean transformation is a change to another initial reference frame moving with constant velocity. This should not affect the properties of the flow.
Figure 3: Particles jump to the next nearest nodes in discrete time steps

- Particles exchange momentum.
  - Example: 2-particle interaction:

- Locally conserved quantities:
  - Mass.
  - Momentum.

Figure 4: Particle Interaction

- The fluid pressure $P$ does not have the right dependence on the speed of sound in the fluid.

- The fluctuations in the fluid flow are too large: the flow is too noisy.

- Only low Reynolds numbers, i.e., very viscous fluid flows, can be
simulated correctly.

- There are unphysical conserved quantities in addition to the physical conserved quantities (mass, momentum, energy, angular momentum).

- The collision rules become exponentially complex in many models, including the simplest realistic 3-D model.

The last problem is interesting. In the 2-D model, each cell can have up to 6 particles, each with a different velocity. The number of possible states is $2^6 = 64$ because each particle type can either be present or absent. In a collision, each state is transformed to a definite other state. One way of implementing collisions is with a $64 \times 64$ lookup table.

In 2-D, Frisch, Hasslacher and Pomeau found that a hexagonal lattice of cells was sufficient to recover isotropy (rotational invariance) in Navier-Stokes equations: an earlier model with a square lattice did not have this property and did not give realistic results.

## 2.4 Lattice Boltzmann Method

![Diagram](image)

Figure 5: From lattice gas to lattice Boltzmann

The problems with Lattice Gas Cellular Automata were mostly resolved with the introduction of Lattice Boltzmann Models.

The continuum theory of fluids can be viewed at different length scales:

- At the molecular level, the fluid molecules obey deterministic Newtonian equations of motion for the position $\mathbf{r}_i(t)$ and velocities $\mathbf{v}_i(t)$ of the particles.
• At the kinetic theory level, and for a dilute gas, the individual particles are replaced by a probability distribution function \( n(r, v, t) \), which is the probability that there is a particle with position \( r_i \) and velocity \( v_i \) at time \( t \). This distribution function obeys the Boltzmann equation

\[
\frac{\partial n}{\partial t} + v \cdot \nabla n + F \cdot \nabla v n = \left( \frac{dn}{dt} \right)_{\text{coll}} \tag{1}
\]

where the collision term is determined by the 2-body scattering cross section

\[
\left( \frac{dn}{dt} \right)_{\text{coll}} = \int d^3v_2 d\Omega (n_1 n_2' - n_1 n_2) v_{\text{rel}} \sigma (v_{\text{rel}}, \Omega) \tag{2}
\]

• At the fluid-dynamics level, one averages over small but macroscopic regions of space and over time much longer than the molecular mean free time: the Boltzmann equation then reduces to the Navier-Stokes and continuity equations for fluid flow.

The Lattice Boltzmann model is derived in a similar way:
• The analogue of Newtonian dynamics is the Lattice Gas Cellular Automaton with its deterministic dynamics. The fixed number of particles with definite position and velocities are discretised by populating the cells with a finite number of particles with discrete velocity values.

• A Lattice Boltzmann Model is obtained by replacing the particles with definite velocities at each cell by a probability distribution \( n_i(r, t) \) to find a particle with discrete velocity \( c_i \) at discrete cell position \( r \) at time \( t \). This distribution function obeys the Lattice Boltzmann Equation

\[
n_i(r + c_i, t + 1) = n_i(r, t) + C_i(n) \tag{3}
\]

where \( C_i \) is a discrete version of the collision operator. In the approximation of Bhatnagar, Gross and Krook (BGK)

\[
C_i(n) = - \frac{n_i - n_i^{eq}}{\tau} \tag{4}
\]

• The fluid-dynamics limit of this model leads to the Navier-Stokes and continuity equations for fluid flow.
Lattice Gas Cellular Automaton Model: Velocities $c_i$ are discrete.

- **Advantage:** integer arithmetic can be used and computations are very fast.

- **Disadvantage:** the fluid velocity also takes only a finite number of discrete values. This is the origin of the breakdown of Galilean invariance and the anomalous dependence of pressure on the sound velocity.

Lattice Boltzmann Model: The velocities $c_i$ of the fluid particles are discrete, but one averages over an *ensemble* of systems to obtain a probabilistic distribution $n_i$ of velocities.

- **Disadvantage:** $n_i$ are real numbers and we need to use floating point arithmetic. The advantage over finite difference approximation is lost.
- Advantage: the fluid density

\[ \rho = \sum_i n_i \]  (5)

and velocity

\[ u = \sum_i n_i c_i \]  (6)

are both continuous, which makes it possible to satisfy Galilean invariance and to avoid other problems with the Lattice Gas Model.

2.6 Choice of Discrete Velocities

The set of allowed velocities in the Lattice Boltzmann Models is restricted by conservation of mass and momentum, and by rotational symmetry (isotropy). However, these restrictions turn out to be much less severe than in the Lattice Gas Cellular Automaton Models.

The following table gives some popular lattices. The weight of the velocity can be thought of as the mass of the fluid particle. The magnitude of the velocity of the particle is determined such that it moves to the nearest lattice in the direction of its velocity: this determines its kinetic energy.

The D1Q3 model has a 1-D lattice with one zero velocity and two oppositely directed velocities which move the fluid particle to the left and right neighbour lattice sites. The D1Q5 model extends D1Q3 by moving particles to the next-nearest neighbour sites in addition.

<table>
<thead>
<tr>
<th>Model</th>
<th>( c^2 )</th>
<th>Energy</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1Q3</td>
<td>1/3</td>
<td>0</td>
<td>4/6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/2</td>
<td>1/6</td>
</tr>
<tr>
<td>D1Q5</td>
<td>1</td>
<td>0</td>
<td>6/12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/2</td>
<td>2/12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1/12</td>
</tr>
<tr>
<td>D2Q9</td>
<td>1/3</td>
<td>0</td>
<td>16/36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/2</td>
<td>4/36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1/36</td>
</tr>
<tr>
<td>D3Q15</td>
<td>1/3</td>
<td>0</td>
<td>16/72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/2</td>
<td>8/72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/2</td>
<td>1/72</td>
</tr>
<tr>
<td>D3Q19</td>
<td>1/3</td>
<td>0</td>
<td>12/36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/2</td>
<td>2/36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1/36</td>
</tr>
</tbody>
</table>
Figure 7: Propagation in the model D2Q9

The D2Q9 model is a 2-D lattice (D2) with 9 discrete velocities: 0, N, S, E, W, NE, NW, SE, SW.

The D3Q15 model is a 3-D lattice with 15 discrete velocities: 0, 6 towards face centres, and 8 towards vertices of a cube.

The D3Q19 model is a 3-D lattice with 19 discrete velocities: 0, 6 velocities to the face centres, and 12 towards edge centres of a cube.

The Lattice Boltzmann Equation in the Bhatnagar-Gross-Krook approximation is

$$n_i(r + c_i, t + 1) = n_i(r, t) = n_i - n_{eq} i \tau$$

(7)

Here $\tau$ is a single relaxation time, and $n_{eq} i$ is the equilibrium configuration, which is given by

$$n_{eq} i = \rho w_i \left(1 + \frac{{\bf u} \cdot {\bf c}_i}{c_s^2} + \frac{(\frac{\rho}{\rho_s})^2 - c_s^2}{2c_s^4} \right)$$

(8)

where $c_s$ is the speed of sound given by

$$\sum_i w_i(c_i)_a (c_i)_b = \delta_{ab} c_s^2$$

(9)

and $w_i$ are a set of directional weights normalised to unity. The discrete lattice velocities and weights are constrained by conservation of
mass, momentum and angular momentum:

\[ \sum_i n_i^{eq} = \rho \]  

(10)

where \( \rho \) is the fluid density

\[ \sum_i n_i^{eq} c_i = \rho u \]  

(11)

where \( u \) is the fluid velocity, and

\[ \sum_i n_i^{eq} (c_i)_a (c_i)_b = \rho \left( (u_i)_a (u_i)_b + c_s^2 \delta_{ab} \right) \]  

(12)

The conservation laws imply the following conditions:

\[ \sum_i w_i = 1 \]  

(13)

\[ \sum_i w_i c_i = 0 \]  

(14)

\[ \sum_i w_i (c_i)_a (c_i)_b = P \delta_{ab} \]  

(15)

where \( P \) is the fluid pressure.

It can be shown that with these restrictions the Navier-Stokes equations are obeyed with the fluid pressure given by

\[ P = \rho c_s^2 \]  

(16)

and the kinematic viscosity given by

\[ v = c_s^2 \left( \frac{1}{\omega} - \frac{\delta t}{2} \right) \]  

(17)

where \( \omega = \frac{1}{\tau} \) is the relaxation frequency, and \( \delta t \) is the cellular automaton time step which was chosen to be \( \delta t = 1 \).

### 2.7 Porous Media

One of the applications of lattice gases is the study of complex flows through complex geometries. The complex geometry examined is one of the most complicated that nature has to offer - a porous rock. The flows are those of a simple fluid, such as water. The problems illustrated are not only of intrinsic interest for physics but have applications in fields as diverse as hydrology, oil recovery, and biology.
The science of flow through porous media had its genesis in the mid-nineteenth century work of the French engineer Henri Darcy. By conducting experiments with sand packs, Darcy found that the flow rate \( J \) of fluid flowing through porous media was proportional to the force applied to it. In symbols, we write Darcy’s law as

\[
J = -\frac{k}{\mu} (\nabla p - \rho g)
\]

(18)

where \( \mu \) is the viscosity of the fluid, \( \nabla p \) is the pressure gradient applied to it, \( \rho g \) is the gravitational force density, and \( k \) is the permeability. The permeability coefficient has dimensions of area, and is a measure of the conductivity of the porous medium to fluid flow through it.

For an incompressible fluid, we may be reasonably confident that the flow at the scale of a pore is described by the Navier-Stokes equations.

The lattice Boltzmann method is useful for flow through porous media because they are easy to apply to problems with irregular geometries.
3 Implementation

The code used for the simulations I investigated simulated 2-d flow in a rectangular region although other regions could have been used.

The discrete velocities were chosen according to the D2Q9 model. This means a 2-D lattice was used with 9 discrete velocities: 0, N, S, E, W, NE, NW, SE, SW.

3.1 Boundary Conditions

Various types of boundary conditions are possible:

- Periodic boundary conditions are useful for modeling bulk systems because they tend to minimise finite size edge effects.

- No-slip boundary conditions are appropriate for most fluids in contact with a wall.

- Frictional slip (or the limiting case of free-slip) boundary conditions may be appropriate for smooth boundaries with small (or negligible) friction exerted on the flowing gas or liquid.

- Open inlets and outlets.

3.2 Periodic Boundary Conditions

Boundary conditions are straightforward to derive once the model is specified. Consider the D2Q9 model with a rectangular region. The discrete velocities are numbered as follows:

6 2 5
3 0 1
7 4 8

The boundary values at the West end of the region ($x = 0, y$) are implemented by transferring the densities with positive $x$ component of velocity from the East boundary ($x = n_x, y$):

```c
/*
 * Compute weighing factors (depending on lattice geometry) for
 * increasing/decreasing inlet densities.
 */
```
t_1 = *density * (*accel / 9.0);
t_2 = *density * (*accel / 36.0);

for (y = 1; y <= ly; y++)
{
    /*
     * Accelerated flow only on non-occupied nodes and check to
     * avoid negative densities
     */
    if (((obst[1 + y*MAXX]) == 0)
        && ((node[3 + 1*9 + y*9*MAXX]) - t_1 > 0)
        && (node[6 + 1*9 + y*9*MAXX] - t_2 > 0)
        && (node[7 + 1*9 + y*9*MAXX] - t_2 > 0)){
        /*
         * increase east
         */
        node[1 + 1*9 + y*9*MAXX] = node[1 + 1*9 + y*9*MAXX] + t_1;
        /*
         * decrease west
         */
        node[3 + 1*9 + y*9*MAXX] = node[3 + 1*9 + y*9*MAXX] - t_1;
    }
}

Note that it is only necessary to transfer three of the 9 densities that will then flow into the region.

### 3.3 No-slip Boundary Conditions

Let’s consider the North wall with lattice sites \((x, y = ny + 1)\). The appropriate boundary conditions, which will ensure that the fluid velocity at the wall is zero, are implemented as follows:

/*
 * increase north-east
 */

node[5 + 1*9 + y*9*MAXX] = node[5 + 1*9 + y*9*MAXX] + t_2;

/*
 * decrease north-west
 */
It’s only need to set the densities with negative \( y \)– component of velocity, namely (4,7,8). The boundary conditions are implemented by simply reversing these velocities. This fluid velocity normal to the wall is proportional to
\[
(n_6 + n_2 + n_5) - (n_7 + n_4 + n_8) = 0
\] (19)
The tangential fluid velocity component is proportional to
\[
(n_5 + n_1 + n_8) - (n_6 + n_3 + n_7) = n_1 - n_3
\] (20)
Since the components, \( n_{1,3} \) parallel to the wall do not change during the simulation, we can set \( n_1 = n_3 \) at the wall initially so the parallel velocity component will remain zero.

### 3.4 Obstacle

The code used allows for obstacles with no-slip boundary conditions to be used. The size of the rectangle region used for most simulations was 1000 \( \times \) 500 lattice points. Obstacles were plotted using data files of \( x \)– and \( y \)– co-ordinates.

This code was written in the C programming language. Code was written to read in a data file which had the boundary and obstacles coordinates. Gnuplot was used to show the fluid interacting with the boundaries and various obstacles.

### 3.5 Poiseuille Flow Problem

This is the viscous flow through a channel under the action of a pressure gradient. With no-slip boundary conditions at the wall of the channel the flow develops a parabolic velocity profile which is stable up to Reynold’s numbers of about 2000.

There is a problem with simulating Poiseuille flow using the Lattice Boltzmann Equation because the system behaves like an ideal gas with equation of state:
\[
P = \rho c_s^2
\] (21)
where \( c_s \) is the speed of sound. For the D2Q9 model \( c_s^2 = \frac{1}{3} \). In addition, the flow is incompressible with constant \( \rho \). Thus in equilibrium, the pressure \( P \) is constant and there cannot be a pressure gradient to drive the flow.
Figure 8: Examples of some of the obstacles that were used.

In a real, incompressible fluid, the speed of sound is very large compared with the fluid velocity, and small pressures are consistent with almost constant density. But in the lattice model, the speed of sound is comparable to the fluid velocity! A trick to simulate a constant pressure gradient is to introduce a body force which transfers the same momentum to the fluid to overcome viscosity as would a pressure gradient. This is shown in the appendix.
3.6 Porous Media

Porous media are created by generating a random number in the interval \(0 < r \leq 1\) for every site on a lattice. A site is occupied by a barrier if its random number satisfies the condition \(r \leq p\). Thus, the higher the value of \(p\), the smaller the fraction of sites not containing barriers (porosity, \(\phi\)).

![Figure 9: From left to right p = 0.1, 0.25, 0.5, 0.9 respectively]

Consider Figure 9, the higher the value of \(p\) used to generate the system the more likely it is that a cluster of barrier sites will form that extends from one edge of the lattice to the other. Such a cluster is called a spanning cluster.
4 Evaluation

4.1 One Circular Barrier

Figure 10: Simulation of an incompressible flow past a circular barrier for a Reynolds number $Re = 100$. 
4.2 Four Circular Barriers

Figure 11: Flow past four circular objects.
4.3 Arrow-like Barrier

Figure 12: Flow past a sizable barrier.
4.4 Rectangular Barrier

Figure 13: Flow past a rectangular barrier
4.5 Porosity

Figure 14: Flow through a porous medium
4.6 Discussion

4.6.1 Flow Past a single circular barrier

Vectors show flow velocities. Initially the area behind the plane is turbulent. Then the fluid reaches a steady-state. When this happens, the velocity of fluid entering is increased. There are high levels of disturbance in the last two pictures.

Colour gradients indicate increasing levels of vorticity. The characteristic vortex shedding past the circular barrier appears as the long-term solution. Figure 10 depicts the distribution of the vorticity at the periodic shedding range obtained by starting at rest.

4.6.2 Flow Past 4 circular barriers

Simulation again suggests initial turbulence and then steady state. The vector plot has formed a stable vortex in the immediate wake of the final object.

4.6.3 Flow Past an arrow-like barrier

The Reynolds numbers is increased again and the turbulence increases accordingly. The final figure shows that the barrier is shedding vortices and a so-called von Karman vortex has built up.

4.6.4 Flow Past a rectangular barrier

There are high levels of turbulence at the left hand side of the block where the fluid initially crashes into the barrier. A vortex has formed at the right hand side of the block.

The bottom image in figure 13 illustrates the vortex shedding phenomenon of flow past a square block. The Reynolds number used in this analysis is 150.

Since this is a typical periodic unsteady flow problem, a perturbation was introduced in the analysis. This was achieved by specifying the inlet velocity above the block 10% larger than the inlet velocity below the block. After a time, the inlet velocity was made uniform.

This qualitative behaviour is in good agreement with theory and demonstrates how LBM simulations of fluid flow give realistic results.
4.6.5 Flow through a porous medium

Figure 14 shows the diffusion of a fluid through a lattice generated with $p=0.15$. A porous lattice was generated using the method previously discussed. Flow through the porous media was investigated by injecting particles at the left-hand-side of the lattice and at the right-hand-side. Once the particles met, they only entered from the left-hand-side. They approach allowed the simulation to achieve a steady state in a quicker time.
5 Conclusion

It has been shown that the Lattice Boltzmann method possesses many of the characteristics of a real fluid. The advantages of using the Lattice Boltzmann method over the Lattice Gas model have been demonstrated also. None of the LBM simulations had any of the characteristic noise of the LGCA approach.

Also, it has been demonstrated that the Lattice Boltzmann method is suitable to be applied to the field of porous media.

Many of the characteristic features of turbulence were reproduced and it was shown that the turbulence increased with the Reynolds numbers.

Overall, the Lattice Boltzmann method is a better way of modeling fluid flow.
6 References

6.1 Books
1. Rothman and Zaleski. Lattice-gas Cellular Automata

6.2 Internet
1. http://www.maths.tcd.ie/~pwalsht/Mpg/

6.3 Newsgroups
1. sci.physics.computational.fluid-dynamics
2. maths.test
A Lattice-Boltzmann method implementation: algorithm - summary

1. Data input (text file, reinforcement geometry, fluid flow problem definition, LB model parameters etc.)

2. Lattice building (cubic, uniform and regular)

3. Initialisation (initial conditions, tow permeability, particle distribution function (9) etc.)

4. Implementation of LBM on the lattice (loop computation with model time)
   - Computation of streaming of particles to new locations (shifting the particle distribution function to the neighbour node in the direction of the velocity)
   - Computation of effect of boundary conditions (ensuring the periodic boundary conditions; bounce-back rule if walls or solid tows; velocity or pressure gradient as driven force)
   - Computation of macroscopic variables (density, velocity, pressure etc.)
   - Computation of effective permeability
   - Computation of additional forcing terms (gravity etc.)
   - Computation of new equilibrium particle distribution function
   - Compute particle function redistribution due to collision
   - Check for reaching the steady flow or instability to stop the calculations

5. Results output.
B Code

/*
 * Square speed of sound
 */

c_squ = 1.0 / 3.0;

/*
 * Loop over all nodes
 * ATTN: actual densities are stored after the propagation
 * step in the help-array n_hlp
 */

for (x = 1; x <= lx; x++)
{
    for (y = 1; y <= ly; y++)
    {
        /*
         * Only free nodes are considered here
         */

        if (obst[x + y*MAXX] == 0)
        {
            /*
             * integral local density
             * initialise variable d_loc
             */

            d_loc = 0.0;

            for (i = 0; i <= 8; i++)
            {
                d_loc = d_loc + n_hlp[i + x*9 + y*9*MAXX];
            }

            /*
             * x- and y- velocity components
             */

            u_x =
                ((n_hlp[1 + x*9 + y*9*MAXX] + n_hlp[5 + x*9 + y*9*MAXX]
                    + n_hlp[8 + x*9 + y*9*MAXX]) -
                 (n_hlp[2 + x*9 + y*9*MAXX] + n_hlp[6 + x*9 + y*9*MAXX]
                    + n_hlp[9 + x*9 + y*9*MAXX]) -
                 (n_hlp[3 + x*9 + y*9*MAXX] + n_hlp[7 + x*9 + y*9*MAXX]
                    + n_hlp[10 + x*9 + y*9*MAXX]) -
                 (n_hlp[4 + x*9 + y*9*MAXX] + n_hlp[11 + x*9 + y*9*MAXX]
                    + n_hlp[12 + x*9 + y*9*MAXX])) -
(n_hlp[3 + x*9 + y*9*MAXX] + n_hlp[6 + x*9 + y*9*MAXX] +
  n_hlp[7 + x*9 + y*9*MAXX])) / d_loc;

u_y =
((n_hlp[2 + x*9 + y*9*MAXX] + n_hlp[5 + x*9 + y*9*MAXX]
  + n_hlp[6 + x*9 + y*9*MAXX]) -
(n_hlp[4 + x*9 + y*9*MAXX] + n_hlp[7 + x*9 + y*9*MAXX] +
  n_hlp[8 + x*9 + y*9*MAXX])) / d_loc;

/*
* Square velocity
*/

u_squ = ((u_x * u_x) + (u_y * u_y));

/*
* n- velocity components (n = lattice node connection vectors)
* This is only necessary for clearance, and only 3 speeds
* would be necessary.
*/

u_n[0] = u_x;
u_n[1] = u_y;
u_n[2] = -u_x;
u_n[3] = -u_y;
u_n[4] = u_x + u_y;
u_n[5] = -u_x + u_y;
u_n[6] = -u_x - u_y;
u_n[7] = u_x - u_y;

/*
* Equilibrium densities. This can be rewritten to improve
* computational performance considerably.
*/

/*
* Zero velocity density
*/

n_equ[0] = t_0 * d_loc * (1.0 - (u_squ / (2.0 * c_squ)));

/*
* Axis speeds (factor: t_1)
*/
n_equ[1] =
    t_1 * d_loc * (1.0 + u_n[0] / c_squ +
    pow (u_n[0],
    2) / (2.0 * pow (c_squ, 2)) -
    u_squ / (2.0 * c_squ));

...  /*
    * Diagonal speeds (factor: t_2)
    */

n_equ[5] =
    t_2 * d_loc * (1.0 + u_n[4] / c_squ +
    pow (u_n[4],
    2) / (2.0 * pow (c_squ, 2)) -
    u_squ / (2.0 * c_squ));

...

/*
    * Relaxation step
    */

for (i = 0; i <= 8; i++)
    {
    node[i + x*9 + y*9*MAXX] =
    (n_hlp[i + x*9 + y*9*MAXX] +
    (*omega * (n_equ[i] - n_hlp[i + x*9 + y*9*MAXX])));
    }