# Lattice-Boltzmann Fluid Dynamics

Physics 3300, Weber State University, Spring Semester, 2012

In this project you will write a Java program to simulate the flow of a two-dimensional fluid. This simulation will use several of the computational techniques you learned in previous projects, combined in a new, richer context. You will use your simulation to gain some conceptual understanding of how fluids behave, and to conduct a few numerical experiments.

## Fluid Dynamics Background

Fluid dynamics is a notoriously difficult branch of physics, in which very few problems can be solved analytically. The difficulties are hardly surprising: Anyone can see that the behavior of a fluid is more complicated than that of, say, a vibrating solid. While a fluid, like a solid, can undergo linear oscillations (sound waves), it can also flow around obstacles and, in many circumstances, curl around itself to form vortices. This behavior is inherently nonlinear and difficult to understand quantitatively.

We normally describe the macroscopic state of a fluid in terms of its density, $\rho$, and its velocity, $\vec{u}$, both of which are functions of position and time ("fields," if you like). These functions obey a set of coupled nonlinear partial differential equations, called the *Navier-Stokes equations*, which you can derive using Newton's laws and some reasonable assumptions about how fluids behave.

Because the Navier-Stokes equations are so difficult to solve analytically, computational methods are often the most fruitful approach to fluid dynamics problems. In fact, computational fluid dynamics (CFD) is a massive field of pure and applied research. The traditional approach to CFD is to discretize the Navier-Stokes equations and then solve them, numerically, for the desired boundary conditions. Unfortunately, coding such a simulation from scratch can be quite laborious.

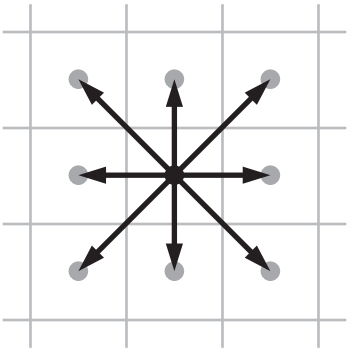
## The Boltzmann Distribution on a Lattice

In the early 1990s, researchers developed a completely different approach to CFD, starting from physics at the molecular scale. Suppose that the fluid is an ideal gas, and suppose for now that it has no macroscopic velocity ($\vec{u} = 0$) and is in thermal equilibrium at temperature $T$. Then the molecules will have *thermal* velocities $\vec{v}$ that are distributed according to the Boltzmann distribution of statistical mechanics. For a two-dimensional gas, this distribution is

$$D(\vec{v}) = \frac{m}{2\pi kT} e^{-m|\vec{v}|^2/2kT}, \tag{1}$$

where $m$ is the mass of each molecule and $k$ is Boltzmann's constant. This is the function which, when integrated over any range of $v_x$ and $v_y$, gives the probability that a particular molecule will have a velocity in that range. It is basically just a "Boltzmann factor," $e^{-E/kT}$, with $E$ given by the ordinary kinetic energy, $\frac{1}{2}m|\vec{v}|^2$. The factor in front of the exponential is needed to normalize the distribution so that its integral over *all* $v_x$ and $v_y$ equals 1. (Please check this in your lab notes.)

In the *lattice*-Boltzmann approach, we discretize both space and time so that only certain discrete velocity vectors are allowed. In this project, we will use the so-called D2Q9 lattice in which there are two dimensions and just nine allowed displacement vectors, shown in the illustration below. One of the allowed displacements is zero, while the other eight take a molecule from its current site into one of the eight nearest sites in the square lattice—either horizontally, vertically, or at a 45-degree diagonal. We can write the nine allowed displacement vectors as $\vec{e}_i$, where $i$ runs from 0 through 8 and each $\vec{e}_i$ has $x$ and $y$ components that are $-1$, 0, or 1, in units of the lattice spacing, $\Delta x$. If the simulation time step is $\Delta t$, then the allowed velocity vectors are given by $c \cdot \vec{e}_i$, where $c$ is an abbreviation for $\Delta x/\Delta t$.



$$\vec{e}_0 = 0$$

$$\vec{e}_1 = (1,0) \qquad \vec{e}_5 = (1,1)$$

$$\vec{e}_2 = (0,1) \qquad \vec{e}_6 = (-1,1)$$

$$\vec{e}_3 = (-1,0) \qquad \vec{e}_7 = (-1,-1)$$

$$\vec{e}_4 = (0,-1) \qquad \vec{e}_8 = (1,-1)$$

Our task, now, is to attach *probabilities* to these nine velocity vectors, to model the continuous Boltzmann distribution as accurately as possible. For a fluid at rest ($\vec{u} = 0$), equation (1) says that *zero* must be the most probable velocity, while the longer diagonal velocity vectors must be less probable than the shorter horizontal and vertical vectors. Velocities with the same magnitude must have the same probability, regardless of direction. The optimum probabilities turn out to be 4/9 for velocity zero, 1/9 for the four cardinal directions, and 1/36 for the diagonals. I'll denote these probabilities (or *weights*) by $w_i$:

$$w_0 = \frac{4}{9}, \qquad w_1 = w_2 = w_3 = w_4 = \frac{1}{9}, \qquad w_5 = w_6 = w_7 = w_8 = \frac{1}{36}. \tag{2}$$

These weights have the right qualitative properties, and they're correctly normalized to add up to 1. But in addition, they predict the same average values (or "moments")

of $v_x$, $v_y$, and all powers of $v_x$ and $v_y$ up to the fourth power. For example,

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_x^2\, D(\vec{v})\, dv_x dv_y = \sum_{i=0}^{8} (e_{i,x}\cdot c)^2\, w_i, \quad \text{etc.} \tag{3}$$

The left-hand side is the true average value of $v_x^2$ computed from the Boltzmann distribution (that is, the sum of $v_x^2$ over all possible velocities, weighted by their probabilities), while the right-hand side is the lattice version of the same average, where we sum only over the nine allowed velocity vectors. Similar relations hold for the average values of $v_y^2$, $v_x^4$, $v_y^4$, and $v_x^2\, v_y^2$. In other words, the weights $w_i$ have been chosen to approximate the continuous distribution as well as possible, in the sense that all of these average values are preserved. However, this works *only* if the constant $c$ is related to the temperature by

$$c^2 = \frac{3kT}{m}. \tag{4}$$

In your lab notes, please evaluate both sides of equation (3) and thus verify equation (4). You may optionally wish to check that the average values of $v_x^4$ and $v_x^2 v_y^2$ work out correctly. (Average values of odd powers of $v_x$ or $v_y$ are zero, as they must be for a fluid whose macroscopic velocity is zero.)

So much for a fluid at rest. Now what about a fluid that is flowing with a nonzero macroscopic velocity $\vec{u}$? Then the total velocity of any molecule is $\vec{u}$ plus the thermal velocity $\vec{v}$, and we will still constrain this total velocity to be one of our nine lattice velocities:

$$\vec{e}_i\cdot c = \vec{u} + \vec{v}, \qquad \text{or} \qquad \vec{v} = \vec{e}_i\cdot c - \vec{u}. \tag{5}$$

The Boltzmann distribution for thermal velocities is unaffected by $\vec{u}$, so we can plug this difference into equation (1) to obtain a new set of probabilities:

$$D(\vec{v}) \longrightarrow \frac{m}{2\pi kT}\, \exp\left(-\frac{m}{2kT}|\vec{e}_i c - \vec{u}|^2\right). \tag{6}$$

But the idea of constraining the total velocity to these nine values makes sense only if the flow velocity $\vec{u}$ is much less than $c$; otherwise there would be too big a chance of finding molecules moving *two* lattice sites per time step. Assuming, then, that $|\vec{u}| \ll c$, we can simplify equation (6) as follows: Expand the square in the exponent to get three terms, and then break up the exponential into three exponential factors, one for each of these terms. The first factor, together with the prefactor $m/2\pi kT$, should correspond to the probability of having velocity $\vec{e}_i\cdot c$ when the flow velocity $\vec{u}$ is *zero*, that is, $w_i$. Meanwhile, expand each of the remaining exponential factors

in a Taylor series, keeping terms up to order $(\vec{u}/c)^2$. When the smoke clears, you should find

$$D(\vec{v}) \longrightarrow w_i \left[ 1 + \frac{3\,\vec{e}_i \cdot \vec{u}}{c} + \frac{9}{2}\left(\frac{\vec{e}_i \cdot \vec{u}}{c}\right)^2 - \frac{3}{2}\frac{|\vec{u}|^2}{c^2} \right]. \tag{7}$$

This expression for the probability of the $i$th discrete velocity vector is the heart of the lattice-Boltzmann method. Please derive it carefully in your lab notes (and note that the $\rightarrow$ symbol does not represent equality, because of the transition from a continuous to a discrete distribution). Also please check that for any velocity $\vec{u}$, the sum of all nine probabilities equals 1.

## The Lattice-Boltzmann Algorithm

In a lattice-Boltzmann simulation, the fundamental dynamical variables are the nine different *number densities*, of molecules moving at the nine allowed velocities, at each lattice site. Thus, your simulation will need nine two-dimensional arrays of real numbers to represent these densities. I'll call them $n_i(x, y)$, although in Java notation you'll want to replace this with something like `n0[x][y]`, `nE[x][y]`, `nNE[x][y]`, and so on, labeling each density to keep track of the velocity direction that it represents.

At any given time, at a given lattice site, these nine densities can have any positive values. From these values you can then compute the total density, $\rho$, as well as the $x$ and $y$ components of the average (that is, macroscopic) velocity, $u_x$ and $u_y$. And from these three macroscopic variables, you can compute what the nine densities *would* be if the molecules at this site were in thermal equilibrium:

$$n_i^{\text{eq}} = \rho\, w_i \left[ 1 + 3\,\vec{e}_i \cdot \vec{u} + \tfrac{9}{2}\,(\vec{e}_i \cdot \vec{u})^2 - \tfrac{3}{2}|\vec{u}|^2 \right]. \tag{8}$$

This is the same as expression (7) for the probability, multiplied by the current density $\rho$ and in units where $c = 1$. The most obvious next step would then be to set all the nine densities equal to these equilibrium values; doing so would mimic the process of collisions among the molecules, which brings them closer to thermal equilibrium. But the time scale for reaching equilibrium needn't be identical to the time step of the simulation. So a more general procedure is to change the value of each $n_i$ toward its equilibrium value by a variable fraction:

$$n_i^{\text{new}} = n_i^{\text{old}} + \omega(n_i^{\text{eq}} - n_i^{\text{old}}). \tag{9}$$

Here $\omega$ is a unitless number that can vary between about 0 and 2. A smaller value of $\omega$ means that collisions take longer to bring the densities into equilibrium, while a larger value of $\omega$ means that the collisions happen more quickly.

The lattice-Boltzmann algorithm, then, proceeds as follows:

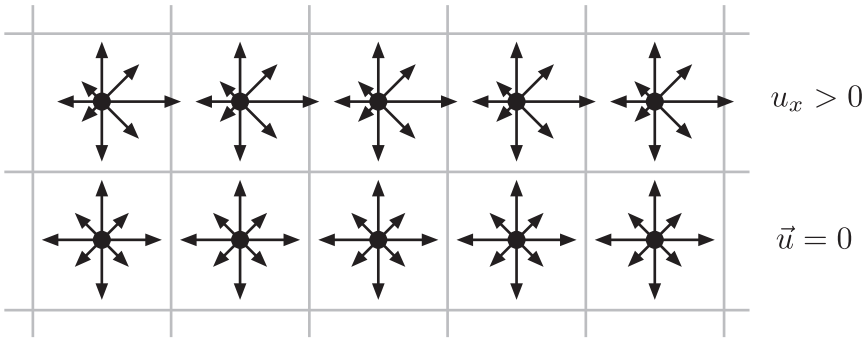
- **Collisions.** For each lattice site, do the following:
    1. From the nine microscopic densities $n_i$, compute the macroscopic density $\rho$ and velocity components $u_x$ and $u_y$.
    2. From these three macroscopic variables, use equation (8) to compute the equilibrium number densities $n_i^{\text{eq}}$.
    3. Update each of the nine number densities according to equation (9).
- **Streaming.** Move all the moving molecules into adjacent or diagonal lattice sites, by copying the appropriate $n_i$ values.

Amazingly, this alternate sequence of collisions and streaming results in the same large-scale flowing behavior as the Navier-Stokes equations, at least under the limited range of conditions that are consistent with our assumptions. (The most important limitation is that we cannot use this method to simulate fluid flow at speeds comparable to, or larger than, the speed of sound.)

## Viscosity

Notice the similarity of equation (9) to the relaxation method for solving boundary value problems, where $\omega < 1$ represents under-relaxation and $\omega > 1$ represents over-relaxation. However, in this case $\omega$ is not merely a property of the algorithm but actually has physical significance.

To understand the physical meaning of $\omega$, consider the situation shown below, in which the top layer of the fluid is initially moving to the right, while the bottom layer is at rest. Then, if all the number densities are at their equilibrium values, the top layer will have more right-moving molecules, and fewer left-moving molecules, than the bottom layer. But in any given cell, 1/6 of all the molecules also have a downward velocity component. After the next streaming step, all down-moving molecules in the lower layer will be replaced by those that came from the upper layer.



Thus, 1/6 of all the rightward momentum in the upper layer will be transferred to the lower layer.

After this streaming step, the molecules within each lattice cell will "collide" and exchange momentum. If $\omega = 1$, the rightward momentum in the lower layer will then be redistributed so that only 1/6 of it is still moving downward (while 1/6 is then moving upward and the remaining 4/6 is purely horizontal). Thus, only 1/6 of the lower layer's rightward momentum will be transfered down to the *next* layer below, during the next streaming step.

On the other hand, if $\omega < 1$, the lower layer won't fully equilibrate during a single collision step and so *more* than 1/6 of its momentum will be transferred down during the next step. In this case, the rightward momentum will be transferred downward through the fluid more effectively, so the top layer "drags" more of the lower layers along with it. We then say that the fluid has a higher *viscosity*.

The case $\omega > 1$ is a little harder to understand. Then, after the first streaming step, the lower layer "over-equilibrates" so that even less than 1/6 of its rightward momentum is transferred on down during the next streaming step. In effect, the equilibration happens in less than a full time step. In any case, the transfer of rightward momentum down through the fluid is less effective, so the top layers slide across with less drag and we say the fluid has lower viscosity.

With a similar but more sophisticated analysis, one can show that the so-called *kinematic viscosity coefficient*, $\nu$, is related to $\omega$ by

$$\nu = \frac{1}{3}\left(\frac{1}{\omega} - \frac{1}{2}\right), \tag{10}$$

in units where $\Delta x = \Delta t = 1$. Thus, $\omega = 1$ corresponds to $\nu = 1/6$, while $\omega = 0$ corresponds to infinite viscosity and $\omega = 2$ corresponds to viscosity zero. (In practice, you'll want to avoid both of these extreme values.)

## Boundary Conditions

It's easy to add walls and other obstacles to a lattice-Boltzmann simulation. Just use a boolean array to keep track of which lattice sites contain obstacles rather than fluid. Then during or after each streaming step, any fluid that would normally flow into one of these sites should instead bounce straight back to the site where it came from, now moving with the opposite velocity. (So, for instance, fluid moving southeastward into an obstacle would bounce back and now be moving northwest.)

Other lattice sites, often along the edges of the simulated space, can contain fluid that is always assigned to have the equilibrium number densities for some fixed, given density and velocity.

Periodic boundary conditions are another option, in which opposite edges of the simulated space are identified with each other so that particles streaming off one

edge come in from the opposite edge. In this case, the streaming step requires some temporary storage space to hold the first row of density values until all the others have been moved to make room for it on the other side.

Depending on the situation being modeled, the fluid could start at rest or at any other reasonably low velocity. In units where $c = 1$, the maximum velocity before the simulation becomes unstable is typically about 0.4. Lower viscosities (higher $\omega$ values) require lower flow velocities. The density scale is arbitrary, so the most natural choice for the initial density is 1.0.

## Simulation Design

Your assignment is to write a lattice-Boltzmann simulation of a "wind tunnel" in two dimensions, with fluid flowing in one side and out the other, and obstacles in between. You'll want to make the lattice two or three times as wide as it is high, so use different variables for the two dimensions. Once your code is optimized, it should run fast enough for interactive use when the width is 200 or 300 and the height is 100 lattice sites. But start with somewhat smaller values.

In addition to your nine fundamental $n_i$ arrays, please create arrays for $\rho$, $u_x$, and $u_y$. You need to compute these quantities during the collision step anyway, so you might as well store them for later use by your `paint` method. That method should draw each lattice site as a small square, colored according to these macroscopic variables or some combination of them. A good first choice is to color the squares according to the speed, $|\vec{u}|$. (Compute the speed in your `paint` method, not in your collision code, so you don't take any more square roots than necessary. There's no need to call `repaint` more than about once for every 10 time steps.)

The simplest way to treat the edges is probably to force these sites to have density 1 and a fixed, given, rightward velocity at all times (with the appropriate equilibrium $n_i$ values for these macroscopic variables). Feel free to try placing walls (obstacles) along the top and bottom edges, or to use periodic boundary conditions to associate the top with the bottom. For the right-hand edge where fluid leaves the simulated space, you can get somewhat nicer results by letting the fluid stream into these sites and then copying the left-moving densities (including northwest and southwest) into them from the next row to the left, and finally scaling all nine of the densities to force the macroscopic density to 1.

The fun way to create obstacles is to draw them with the mouse. You might also want to add some buttons that quickly create simple obstacle shapes such as lines and circles. When you delete an obstacle, you should probably fill in its space with fluid.

Be sure to put in a scrollbar to adjust the flow velocity and another to adjust the viscosity (or $\omega$). You'll need a "reset" button to start the simulation over when it

becomes unstable. Besides plotting the speed of the fluid (using your color scheme), you may want options to plot the density, the separate components of $\vec{u}$, and the *curl* of $\vec{u}$, defined as $\partial u_y/\partial x - \partial u_x/\partial y$ (and thus computed from the four neighboring sites).

Be sure to use your lab notes to document your major decisions and progress in writing the simulation. Also document the difficulties that you encounter and how you resolve each of them.

## Data

Once your simulation is working, the number of scenarios you can study, and experiments you can perform, is truly enormous. Here are a few possibilities:

- **Speed of sound.** Create a sudden disturbance and measure the speed at which the pressure waves travel away from it.
- **Drag forces.** Put obstacles of different shapes in the middle of the flowing fluid, and measure the total force on these obstacles (by keeping track of the densities and directions of the molecules that bounce off). How does the drag force depend on the fluid speed? To what extent can you reduce the drag force by giving the obstacle a more streamlined shape?
- **Vortex shedding.** Under what circumstances will an obstacle produce an unstable flow, shedding vortices alternately from either side as the fluid flows past? What determines the size and frequency of the vortices? Explain your results in terms of the *Reynolds number*, $R = uL/\nu$, where $u$ is the average speed of the fluid relative to the obstacle, $L$ is the size of the obstacle, and $\nu$ is the kinematic viscosity coefficient.

Don't feel limited by this list, but do perform enough experiments, and gather enough data, to produce a lab report that you can be proud of.