Winter Semester 2011 / 2012

Lattice-Boltzmann Methods

Prof. Georg May

Project 7

Due: March 16, 2012

Introduction

In this project we complete the implementation of a Lattice-Boltzmann solver of the form

$$f(\mathbf{x}+\mathbf{c}_i, t+1) = f(\mathbf{x}, t) + \omega(f^{(eq)}(\mathbf{x}, t) - f(\mathbf{x}, t)), \tag{1}$$

where we use the D2Q9 model to define the equilibrium distribution $f^{(eq)}$, as discussed in class.

We need the following new ingredients:

- implement wall boundary conditions;
- set the collision parameter ω;
- add a forcing function (modeling external forces or a known pressure gradient).

As before, the lattice will be set up such that we have a N_x (interior) lattice points in x-direction, and N_y (interior) lattice points in y-direction. It is recommended for the application of boundary conditions to add *ghost points* at every boundary. This means that the total number of lattice points is $N_x + 2$ and $N_y + 2$, in x-direction and y-direction, respectively, and indexing runs from $i = 0, \ldots, N_x + 1$, and $j = 0, \ldots, N_y + 1$.

Poiseuille Flow

Poiseuille Flow is a solution of the Navier-Stokes equations roughly corresponding to steady viscous flow in a pipe under a constant pressure gradient. While normally the Poiseuille solution is formulated for a 3D flow using cylindrical coordinates, here we restrict ourselves to the corresponding 2D case, which is given by flow bounded by two walls in positive and negative y-direction, and having a negative pressure gradient in x-direction.

From the Navier-Stokes equations we infer that this problem admits a steady solution with $\mathbf{u} = (u(y), 0)^T$. Let us denote the (negative) pressure gradient as $p' = -G$, with $G > 0$ [1]. The Navier-Stokes equations yield

$$-G = \mu \frac{\partial^2 u}{\partial y^2}. \tag{2}$$

If we assume that the pressure gradient is known this can be integrated to give the exact solution to the problem. We need only the maximum velocity, which is given as

$$u_m = \frac{G}{2\mu}\left(\frac{H}{2}\right)^2, \tag{3}$$

[1] We chose an explicitly negative pressure gradient for convenience, as this will lead to a positive x-velocity.

where H is the height of the channel. This yields the pressure gradient that for a given viscosity is required to reach a certain maximum velocity :

$$G = \frac{8\mu}{H^2} u_m. \tag{4}$$

Part 1: Set ω and Add the Source Term

As discussed in class the pressure gradient may be added as a source term, as long as its magnitude is known. Here we use eq. (4) to set the value so that the maximum velocity $u_m = 0.1$ is reached[2]. We use $\omega = 1.25$ and calculate the corresponding value of the viscosity from

$$\mu = \rho_0 \left(\frac{1}{\omega} - \frac{1}{2} \right) c_s^2, \tag{5}$$

where $c_s^2 = 1/3$. For simplicity we use constant viscosity, and thus compute μ with a constant $\rho_0 = 1$, which we will use later to also initialize the lattice density. This allows one to compute the value of G from eq. (4), where the channel height H is given by $H = N_y$. The source term can be added as discussed in class, i.e. $f_i \leftarrow f_i + \Delta f_i$ with

$$\Delta f_i = \begin{cases} +G/6 & i = 1,5,8 \\ -G/6 & i = 3,6,7 \\ 0 & \text{otherwise} \end{cases} . \tag{6}$$

It is easy to verify that the momentum added by this is $\sum_i \mathbf{c}_i \Delta f_i = G\mathbf{e}_x$, where $\mathbf{e}_x$ is the unit vector in x-direction, while the net density added by this is $\sum_i \Delta f_i = 0$.

Part 2: Add the Wall Boundary Condition

We previously set periodic boundary conditions for the whole domain. We keep periodic boundary condtitions in x-direction, but change to wall boundary conditions in y. We implement the wall boundary via *bounce back rule*, as discussed in class. Fo the lower and upper wall the links $2, 5, 6$, and $4, 7, 8$ of the ghost cells must be populated, respectively. For example, at the lower wall link 6 in the ghost cells (i.e. $j = 0$) is set via

$$f_{6,i,0} = f_{8,i-1,1}. \tag{7}$$

Again, corner points must be treated with care, taking into account the periodic boundary condtions in x. If the boundary conditions are set via ghost cells in this manner, the streaming operator implemented in the previous homework assignment does not need to be changed.

Part 3: Test the Solver

Using the previously implemented functions we may now code a Lattice-Boltzmann solver using the following algorithm [3]

- Initialize: Set $u_{i,j} = u_0$, $v_{i,j} = 0$, $\rho_{i,j} = \rho_0$

[2]To make our lives easier, we work with dimensionless quantities directly. This way we don't have to bother with units and conversion between dimensional and non-dimensional quantities.

[3]Whenever indices (i, j) appear, by default all operations are to be carried out for all $(i, j) \in [1, Nx] \times [1, Ny]$.

- intitialize $f_{i,j} = f_{i,j}^{(eq)}$ from $\rho_{i,j}, u_{i,j}, v_{i,j}$
- for $step = 1, 2, 3, \ldots$
  - compute $\rho_{i,j}, u_{i,j}, v_{i,j}$ from $f_{i,j}$
  - compute $f_{i,j}^{(eq)}$ from $\rho_{i,j}, u_{i,j}, v_{i,j}$
  - perform collision $f_{i,j} \leftarrow f_{i,j} + \omega(f_{i,j}^{(eq)} - f_{i,j})$
  - apply boundary condtions
  - perform streaming
  - add the forcing term G

Use $u_0 = 0.1$, $\rho_0 = 1$, and a lattice of size $N_x \times N_y = 10 \times 20$. Plot the velocity at $i = 5$ against j after the solution has reached a steady state. (You need to monitor the change of the solution at each step, and make sure you verify that you have reached a steady state.)

Part 4: Porous Media with Arbitrary Porosity Function

Assume that you are given a porosity function, P, with

$$P_{i,j} = \begin{cases} 1 & (i,j) \text{ is a fluid node} \\ 0 & (i,j) \text{ is a solid node} \end{cases}, \qquad (i,j) \in [1, N_x] \times [1, N_y]. \tag{8}$$

All you need to do is implement an additional routine that assigns wall boundary conditions for arbitrary $P_{i,j}$, as discussed in class[4]. Only a few changes to the algorithm presented in Part 3 are necessary:

- Implement a preprocessing step that generates the data structure for the solid/fluid interfaces, as discussed in class. (Note that this should work for *arbitrary* porosity functions!)
- Implement a new routine to apply wall boundary conditions for interior solid/fluid interfaces, as discussed in class.
- At solid nodes, there is no need to perform any of the lattice operations, i.e. collisions and streaming. Nor does one need to compute ρ, $\mathbf{u}$, or $f^{(eq)}$ at solid nodes.

Test this using a lattice of size $N_x = N_y = 70$, and a simple porosity function that simply "cuts" out a circle:

$$P_{i,j} = \begin{cases} 0 & (\frac{i-i_m}{N_x})^2 + (\frac{j-j_m}{N_y})^2 < 0.05 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

where $i_m = N_x/2$, and $j_m = N_y/2$. You may initialize the lattice, as in Part 3, and also use the same viscosity and pressure gradients. Generate a vector plot of the velocity after the solution has reached a steady state. (You may use MATLAB's `quiver` function.)

[4]It is recommend that you simply keep the boundary conditions that you implemented already, i.e the upper and lower wall, as well as the periodic boundary conditions. Just write an additional routine that applies wall boundary conditions also for the new solid nodes in the interior that arise from the porosity function.