Prof. Dr. Ulrich Rüde
Martin Bauer
Sebastian Eibl
Christoph Rettinger

Summer Term
2016

Simulation and Scientific Computing 2
Assignment 4

Exercise 4 (*Lattice Boltzmann*)

In this assignment your task is to write a 2D fluid simulation using the lattice Boltzmann method (LBM). The basic LBM data structure is a two dimensional grid of cells where 9 values, denoted as f_q, have to be stored in each cell. Each of these 9 values are associated with a specific direction, which are labeled according to the points of the compass as shown in Figure 1.

$$q \in \{C, NW, N, NE, W, E, SW, S, SE\}$$

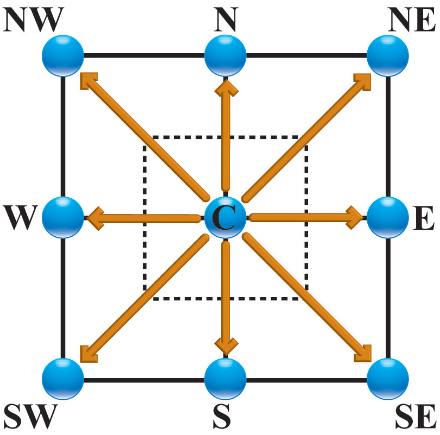

Figure 1: Lattice cell

Physically these 9 values f_q correspond to particle distribution functions i.e. describe the number of particles in that cell moving in direction q.
The directions can be represented by the vectors $\overrightarrow{e_q}$:

$$\overrightarrow{e_C} = (0,0), \quad \overrightarrow{e_N} = (0,1), \quad \overrightarrow{e_W} = (-1,0), \quad \overrightarrow{e_{SE}} = (1,-1), \quad \ldots$$

All macroscopic flow quantities like density and velocity are given as functions of the local f_q:

$$\text{Density:} \quad \varrho = \sum_q f_q \tag{1}$$

$$\text{Velocity:} \quad \overrightarrow{u} = \frac{1}{\varrho} \sum_q f_q \overrightarrow{e_q} \tag{2}$$

The lattice Boltzmann algorithm itself can be separated into two parts: a *streaming* and a *collide* step.

Stream Step

In the streaming step all particle distribution functions are propagated (copied) to the next cell (Fig. 2), e.g. the north value of the current cell is copied to the north value of the northern neighbor cell. During the stream operation values in neighboring cells are overwritten. The simplest solution is to copy the grid first, and read all values from the copy while updating the grid.

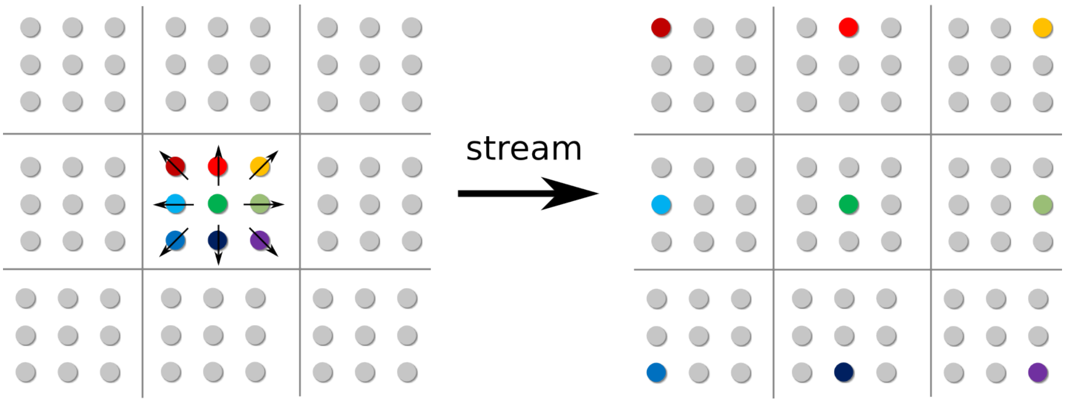

Figure 2: Normal stream step in inner part of domain

Special care has to be taken at the boundary of the domain. We are going to simulate a channel filled with fluid. At the north and south boundary we apply reflection (no-slip) boundary conditions which enfore zero velocity at the wall. The streaming rule for these boundary cell is shown in Fig. 3
To simplify the boundary handling we extend our domain by one row of helper cells and set up the values in these helper cells before streaming such that the normal streaming operation in the inner part of the domain leads to correct no-slip boundaries (Fig. 4).

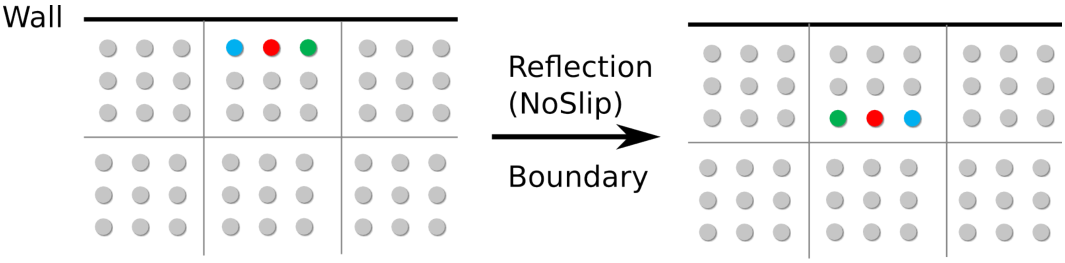

Figure 3: Streaming for reflection (no-slip) boundaries

At the remaining west and east boundaries we apply periodic boundary conditions, again with the help of an additional layer of cells. Here the last inner column is copied to the helper column at the opposite side of the domain.

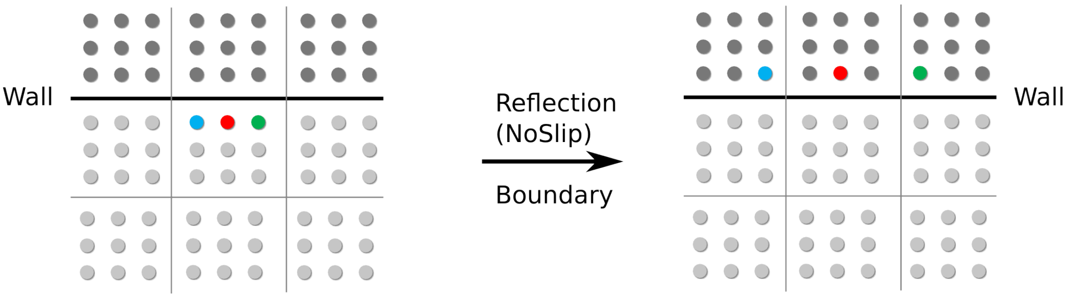

Figure 4: Reflection (no-slip) boundary with helper cells

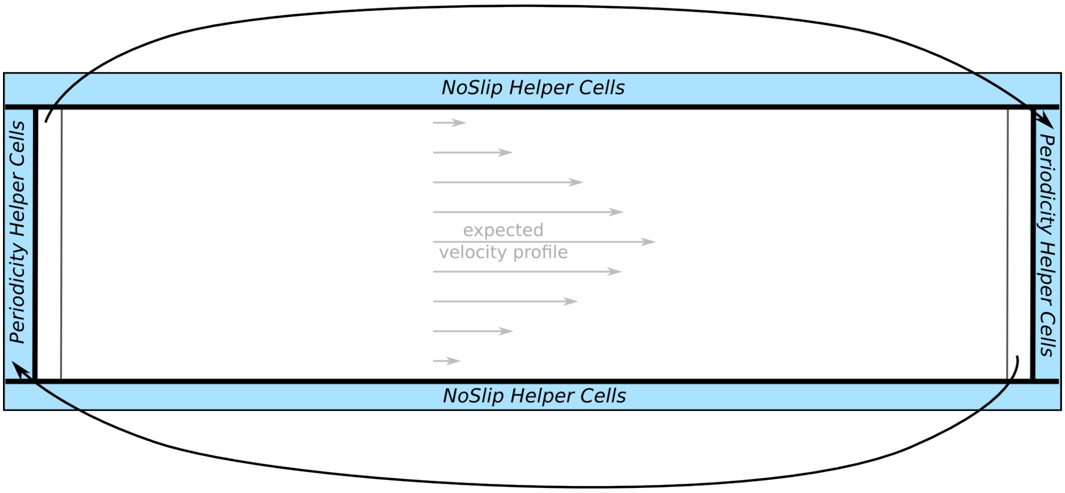

Figure 5: Full simulation domain with reflection boundaries at N,S and periodic boundaries in E-W direction.

Collide Step

The collide step is a purely cell-local operation, i.e. each cell can be treated independently. It models the collision of particles by relaxing them to a local thermodynamic equilibrium. First the density and velocity of the cell are computed according to the equations given above. Then an equilibrium distribution f_q^{eq} is calculated for each direction:

$$f_q^{eq} = f_q^{eq}(\varrho, \overrightarrow{u}) = w_q \varrho \left(1 + 3\overrightarrow{e_q} \cdot \overrightarrow{u} + \frac{9}{2}(\overrightarrow{e_q} \cdot \overrightarrow{u})^2 - \frac{3\overrightarrow{u}^2}{2} \right), \tag{3}$$

with the weighting constants:

$$\begin{aligned} w_q &:= \quad 4/9 \qquad for \quad q = C \\ w_q &:= \quad 1/9 \qquad for \quad q = N, E, S, W \\ w_q &:= \quad 1/36 \qquad for \quad q = NE, NW, SW, SE. \end{aligned} \tag{4}$$

The following update rule has to be applied in all cells for all f_q:

$$f_q \leftarrow f_q - \omega(f_q - f_q^{eq}) + 3 w_q \varrho \, \overrightarrow{e_q} \cdot \overrightarrow{a} \tag{5}$$

We use a single relaxation time (SRT) model, where f_q is relaxed to the equilibrium distribution f_q^{eq} with a single relaxation parameter $\omega \in (0, 2)$. To drive the channel we apply an external acceleration $\overrightarrow{a}$ which is added after the collision step according to above rule.
As derived in the lecture, the relaxation rate ω is a function of the lattice viscosity:

$$\omega = \frac{1}{3\nu_L + \frac{1}{2}}$$

In order for the simulation to be stable ω has to be in the following region: $0 < \omega < 2$

Initial Setup

Initially all cells should be set to equilibrium with a density of 1.0 and zero velocity:
$f_q \leftarrow f_q^{eq}(1.0, (0, 0)) = w_q$

LBM Algorithm Summary

One time step of the lattice Boltzmann method consists of the following steps:

1. handle periodic boundary in E-W directions
2. handle no-slip boundaries in N-S directions
3. stream step
4. collide step
5. (optional) write density/velocity as image

Tasks

Lattice class

To store the 2D lattice, a three dimensional array data structure is required where 2 dimensions are used for space and the third dimension indexing the distributions functions inside one cell.

Implement Stream Step and Boundary Handling

Implement the stream step as well as boundary conditions. Initialize all cells of your domain with the lattice weights w_q and run your boundary handling function and one streaming step. If everything is implemented correctly the f_q's should have the same value as before. Implement a test function to validate this.

Implement Collide Step

Implement the collide step. Initialize the domain as in the previous test and run your collide routine with different (random) choices of $\omega \in (0, 2)$ and an acceleration $\overrightarrow{a} = 0$. Again the f_q's should remain unchanged regardless of your choice of ω. Validate this by adding another test to your project.

Output

To view the results of your simulation you should be able to write out the density and velocity fields. You can use the provided `GrayScaleImage` class for this. Normalize the image such that zero velocity/density corresponds to a gray value of 0 (white) and the maximum value to a gray value of 255 (black).

Simulate Scenario

Now simulate a channel with a circular obstacle as shown in Fig. 6. The inner boundaries can be treated exactly like the helper cells at the north and south domain boundaries. To mark inner boundary cells, use a second 2D boolean array.

Your program should automatically create a visualization of the final state of the scenario, by writing the x-component of velocity as an image file called `scenario1.png` and `scenario2.png`.

	Scenario 1	Scenario 2
Viscosity	$10^{-6}\,\frac{m^2}{s}$	$10^{-6}\,\frac{m^2}{s}$
Time to simulate	$3\,s$	$5\,s$
Acceleration	$0.01\frac{m}{s^2}$	$0.016\frac{m}{s^2}$
Resolution of cylinder diameter	30 cells	60 cells

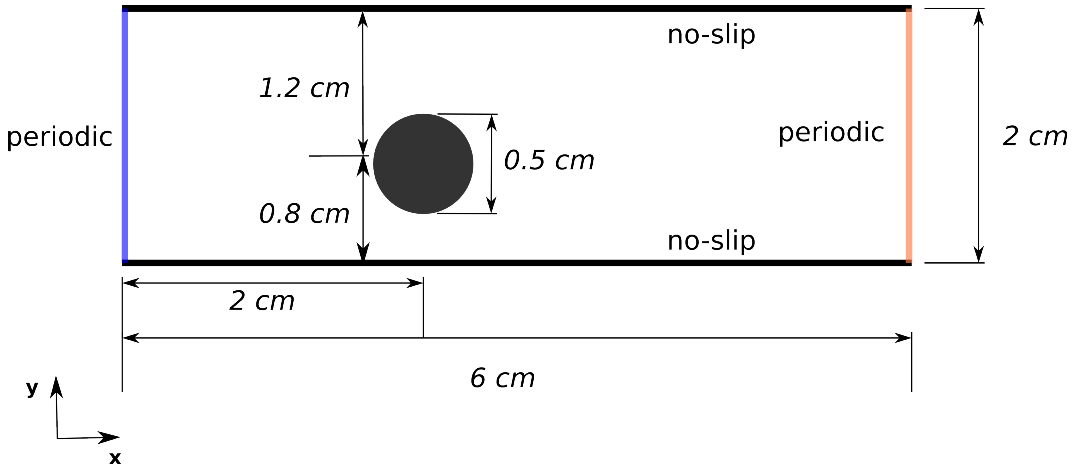

Figure 6: Full simulation domain with reflection boundaries at N,S and periodic boundaries in E-W direction.

Please hand in your solution to this exercise until **Wednesday, July 13, 2016** at 3:00 am! Make sure the following requirements are met:

1. The program should be compilable with a Makefile that you have to provide.
2. The program should compile without errors or warnings with (at least) the following g++ compiler flags:

   ```
   -Wall -pedantic -std=c++11
   ```

3. The program should be callable by `./lbm scenario1` and `./lbm scenario2`. These runs have to create visualizations of the x-velocity as image files named `scenario1.png` / `scenario2.png`. Your program should print the selected timestep δt, lattice spacing δx, acceleration in lattice units and the relaxation rate ω to stdout at the beginning of the run.
4. When submitting the solution, remove all temporary files from your project directory and pack it using the following command:

   ```
   tar -cjf ex04.tar.bz2 ex04/
   ```

   where `ex04` is the folder containing your solution. Then submit your solution via StudOn as a team submission in the known fashion.

Credits

1. You are awarded with up to 2 points if your program correctly performs the above tasks and fulfills all of the above requirements. Submissions with compile errors will lead to zero points! Therefor the computers in the computer science CIP pool act as reference environments.
2. Bonus task: You can obtain additional 0.5 points by extending the program such that you can load arbitrary obstacles into your channel specified via an image file.