Prof. Dr. Ulrich Rüde
Dominik Bartuschat
Kristina Pickl
Christian Godenschwager

Summer Term
2013

Simulation und wissenschaftliches Rechnen 2
Assignment 3

Exercise 3 (*The Lattice Boltzmann method*)

Initial remarks

You have learned in class that the *Lattice Boltzmann method (LBM)* characterizes a class of cellular automata which can be used to solve problems in computational fluid dynamics (CFD). The goal of this programming project is to implement the Lattice Boltzmann method and to solve a common CFD model problem: the so-called lid-driven cavity.

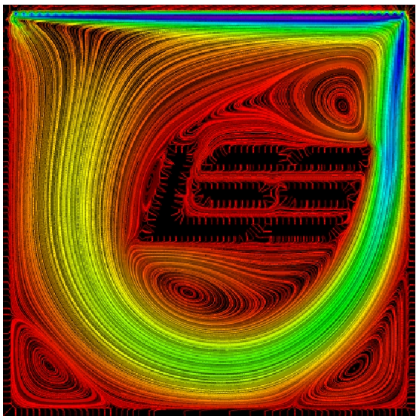

Figure 1: The lid-driven cavity

The lid-driven cavity consists of a closed box completely filled with fluid. The top of the box, the lid, is continually dragged across the fluid, causing the fluid to form a circular flow after a while.

The Lattice Boltzmann method

The basic equation for the Lattice Boltzmann method is the time, space, and velocity discrete Lattice Boltzmann equation:

$$f_\alpha(x + \overrightarrow{c_\alpha}\Delta t, t + \Delta t) - f_\alpha(x,t) = -\omega(f_\alpha - f_\alpha^{eq}). \tag{1}$$

In this equation, the f_α are the probability distribution functions for the discrete velocity $\overrightarrow{c_\alpha}$, f_α^{eq} is the equilibrium distribution function, ω is the collision frequency (containing information about the viscosity of the fluid), and Δt is the size of a single time step.

This equation is solved in two steps:

- Collide step:

$$\tilde{f}_\alpha(x, t + \Delta t) = f_\alpha(x, t) - \omega(f_\alpha - f_\alpha^{eq}) \tag{2}$$

- Stream step:

$$f_\alpha(x + \vec{c_\alpha}\Delta t, t + \Delta t) = \tilde{f}_\alpha(x, t + \Delta t) \tag{3}$$

The equilibrium distribution function necessary for the collide step should be calculated as

$$f_\alpha^{eq} = f_\alpha^{eq}(\varrho, \vec{u}) = t_\alpha \left(\varrho + \frac{3}{c^2}\vec{c_\alpha} \cdot \vec{u} + \frac{9}{2c^4}(\vec{c_\alpha} \cdot \vec{u})^2 - \frac{3\vec{u}^2}{2c^2} \right), \tag{4}$$

where t_α is a model depending weighting factor, ϱ is the fluid density, $\vec{u}$ is the fluid velocity, and c is another model depending parameter (which turns out to be 1 in our case). The equilibrium distribution should also be used as initial condition for the distribution functions (note that initially the velocity is 0 and the density is 1, which simplifies the equation to $f_\alpha^{eq} = t_\alpha$). The fluid density and momentum density (which contains the velocity) can be calculated with the following equations:

$$\text{Density:} \quad \varrho = \sum_{\alpha=0}^{N} f_\alpha \tag{5}$$

$$\text{Momentum density:} \quad \varrho\vec{u} = \sum_{\alpha=0}^{N} f_\alpha \vec{c_\alpha} \tag{6}$$

For this simulation, we choose the 2-dimensional D2Q9 model that uses 9 discrete velocities $\vec{c_\alpha}$ for the discretization of the velocity space:

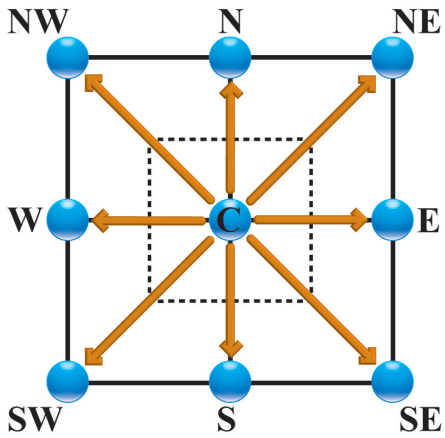

Figure 2: The D2Q9 model

For this model, the model depending weighting factor t_α is

$$\begin{aligned} t_\alpha &= 4/9 \quad for \quad \alpha = C \\ t_\alpha &= 1/9 \quad for \quad \alpha = N, E, S, W \\ t_\alpha &= 1/36 \quad for \quad \alpha = NE, NW, SW, SE. \end{aligned} \tag{7}$$

As boundary condition for the lid-driven cavity, we will use the standard no-slip bounce back boundary conditions. For the moving lid, we will extend these boundary conditions by an additional term:

$$f_{\bar{\alpha}}(x,t) = f_{\alpha}(x,t) - 2t_{\alpha}\frac{3}{c^2}\overrightarrow{c_{\alpha}} \cdot \overrightarrow{u_w}, \tag{8}$$

where f_{α} is the previous distribution function in direction α, $f_{\bar{\alpha}}$ is the distribution function in the opposite direction $\bar{\alpha}$, $\overrightarrow{c_{\alpha}}$ is the velocity in direction α, and $\overrightarrow{u_w}$ is the velocity of the moving wall.

Tasks

1. Implement the LBM in C++ using the D2Q9 model that you know from the lecture. Use the *stream-and-collide* update order; i.e. your code must start with a streaming step.

2. Use the grid layout that is illustrated in Fig. 3.

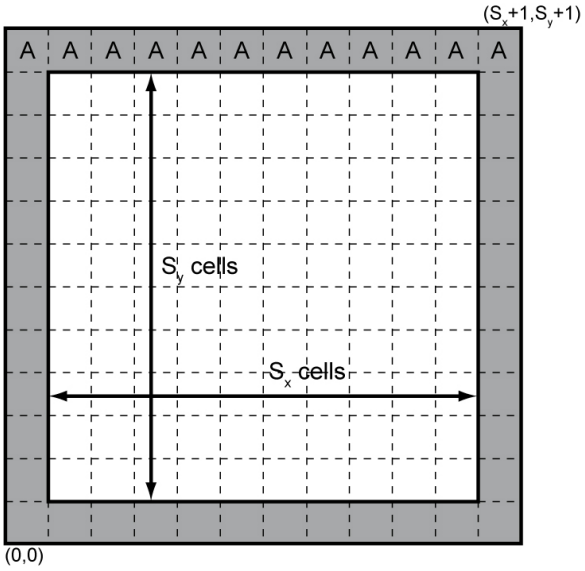

Figure 3: Grid layout: your grid should consist of (S_x, S_y) fluid cells (white) and an additional boundary layer (gray) which results in a grid of $(S_x + 2, S_y + 2)$ cells. The boundary layer consists of no-slip boundary cells where the cells at the top are moving no-slip cells (A) with a velocity of $(0.08, 0)$.

3. The parameters of the Lattice-Boltzmann simulation are specified via a parameter file of the following form:

```
sizex 70
sizey 80
timesteps 1000
omega 1.9
vtk_file example.vtk
vtk_step 300
```

This file specifies the number of fluid cells for your simulation as well as the number of time steps. The `omega` value determines the collision frequency, which lies in the range of 0.5 to 1.95. `vtk_file` gives the name of the visualization files and `vtk_step` specifies the spacing between two visualized time steps (Note: a value of zero for `vtk_step` should turn the visualization off). The first time step to be visualized should be time step `vtk_step`.

4. Visualize your results using the open source visualization tool *Paraview*, which is installed on the LSS CIP pool computers. Type `/usr/local/bin/paraview` on the command prompt to execute it. As *Paraview* is based on the Visualization Toolkit (VTK), write your data as legacy VTK files[1], which use a simple ASCII format.

   Since the fluid cells are of interest, it is sufficient to only visualize these inner fluid cells. For every fluid cell, the fluid velocity and the density should be written to the VTK files.

   As an example consider the following annotated VTK file:

```
# vtk DataFile Version 4.0          # file header
SiWiRVisFile                        # this is only a comment
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 50 50 1                  # number of points in all dimensions
ORIGIN 0 0 0
SPACING 1 1 1
POINT_DATA 2500                     # total number of points

SCALARS flags unsigned_int 1        # scalar field with identifier "flags"
LOOKUP_TABLE default                # default color table
1                                   # flag of first point
...

SCALARS density double 1            # scalar field with identifier "flags"
LOOKUP_TABLE default
1
...

VECTORS velocity double             # vector field with identifier "velocity"
-7.00552e-07 1.90304e-08 0
...
```

   In order to create time series, the VTK files must be tagged with a number: `example0.vtk`, `example1.vtk`, ... *Paraview* automatically detects these time series and offers the functionality to visualize them in an animation. The tag for the time step and the file extension `.vtk` should be appended to the filename specified in the parameter file.

5. Run your simulation with the parameter file provided on the course website, together with reference outputs. Verify your code by comparing your results with these outputs.

6. Measure the performance of your code. The usual performance unit for the LBM is *Mega lattice site updates per second* (MLUps). In order to calculate your MLUps value, count the fluid cells in your domain, multiply that by the number of time steps you are using and divide this number of total updated cells by the time you needed for the simulation.

7. In case you want to improve the performance of your LBM simulation, the master thesis of Klaus Iglberger and the technical report from 2003 (03-3) might contain valuable hints in order to guide your optimization efforts. All documents can be accessed from our web pages (under '*publications/technical reports*' and '*publications/master and short thesis*'). But always check that the simulation results have not changed.

8. Please hand in your solution until Wednesday, June 26, 2013. Make sure the following requirements are met:

   (a) The program should be compilable with a Makefile.

   (b) The program should compile without errors or warnings with (at least) the following g++ compiler flags:

   ```
   -O3 -Wall -ansi -pedantic
   ```

   (c) The program should be callable as `./lbm params.dat`
   where `params.dat` is the parameter file.

   (d) When submitting the solution, remove all temporary files from your project directory and pack it using the following command:

   ```
   tar -cjf ex03_groupXX.tar.bz2 ex03_group_XX
   ```

   where `XX` stands for your group number and `ex03_group_XX` contains your solution. Then send the archive to the course mailing list:

   ```
   siwir2@i10.informatik.uni-erlangen.de
   ```

Credits

In this assignment the 25 credits are awarded in the following way:

1. Up to 25 credits are awarded if your program correctly performs the above tasks and fulfills all of the above requirements. Submissions with compile errors are marked with 0 points. Therefor the computers in the LSS CIP pool act as reference environments.

2. Up to 5 bonus credits can be gained by extending the program such that it is possible to use a pgm image file to specify the domain for the 2D simulation. White pixels with a value of 255 should be considered fluid cells, where all other gray values are considered no-slip cells. Additionally, extend your parameter file such that the `sizex` and `sizey` parameters are replaced by a `geometry` parameter that specifies the name of the pgm geometry file:

   ```
   timesteps 1000
   omega 1.9
   vtk_file example.vtk
   vtk_step 300
   geometry image.pgm
   ```

References

[1] http://www.vtk.org/VTK/img/file-formats.pdf