Tutorial 5

ESPResSo: Lattice-Boltzmann fluid

Georg Rempfer*

Apr 16, 2012
ICP, Uni Stuttgart

Contents

* georg@icp.uni-stuttgart.de

1 Introduction

In this tutorial we will investigate the transition from laminar to turbulent flow, using one of the Lattice-Boltzmann implementations included in ESPResSo. There are actually two implementations, one running on the CPU and one running on the GPU. The GPU version is significantly faster (up to 50x depending on the graphics card) but it requires an NVIDIA graphics card, which the computers in the ICP CIP pool unfortunately lack. If you have a computer with linux and a NVIDIA graphics card at your disposal, you should consider running the simulations on that machine, since it will allow you to create results of much higher resolution.
There are two other topics this tutorial will cover, which are relevant to simulations in general: visualization of your results with with high quality pictures and videos and how to run simulations unattendedly over long periods of time.

2 Simulation with ESPresSo

▷ 3 Points

Write a simulation script to simulate the system as described below in ESPResSo and hand it in.

Our system consists of a pattern of 3d periodic stripes as obstacles, surrounded by a newtonian fluid and a homogeneous force density acting on the fluid as shown in fig. 1.

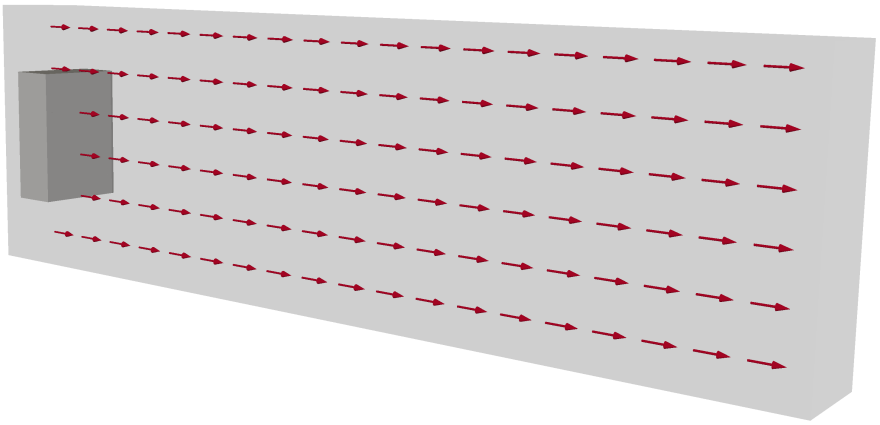

Figure 1: System to investigate. Light gray is the simulation volume with periodic boundaries and the LB fluid. Dark grey represents an obstacle with no-slip boundary conditions and red stands for the applied constant force density.

Turbulence is a phenomena comprising processes on very different time- and length scales, which makes it hard to simulate with moderate computational effort. You are welcome to experiment with the parameters to produce turbulence and any working set of parameters will be appreciated but be warned that this is a time intensive task.

The following set of parameters will work for this task:

- fluid viscosity $\nu = 1.1$
- fluid density $\rho = 1$
- fluid force density $f = 0 \ldots 0.2$
- fluid grid spacing $a = 1$
- time step $\tau = 0.01$
- box size $= 200x60x20$
- obstacle dimensions $= 10x30x20$

You will need to compile ESPResSo with the following features defined in `myconfig.h`:

`EXTERNAL_FORCES`, `LB`, `LB_BOUNDARIES`.

> Should you want to take advantage of the GPU Lattice-Boltzmann implementation, you need a CUDA capable NVIDIA graphics card with the appropriate drivers and the CUDA toolkit (available at http://developer.nvidia.com/cuda-downloads). You need to pass the location of the CUDA toolkit to ESPResSo before compiling with
>
> `./configure --with-cuda=/usr/local/cuda`,
>
> assuming you installed the toolkit in `/usr/local/cuda`. Then you need to compile ESPResSo with the options
>
> `EXTERNAL_FORCES`, `LB_GPU`, `LB_BOUNDARIES_GPU`
>
> enabled in `myconfig.h`. Of course you can ask your tutor for help.

To set up the simulation you will need the following ESPResSo commands, apart from the commands you already used in previous tutorials

- `thermostat off`
- `lbfluid <cpu|gpu> dens` ρ `agrid` a `tau` τ `visc` ν `ext_force` f `0 0`
- `lbboundary rhomboid corner` p_x p_y p_z `a` a_x a_y a_z
  `b` b_x b_y b_z `c` c_x c_y c_z `direction outside`

Please consult the user's guide that came with this tutorial if you need further information on those commands. The commands to produce output are given in section 3.1. Also note that although we are not going to use particles, you will have to set a Verlet skin with

`setmd skin 0.1`

3 Visualization

For this part of the tutorial we are going to use paraview, an easy to use, open source visualization program for scientific data. You should find a preinstalled version on the CIP pool computers, which you can execute with the command

```
/share/sw/ParaView-3.14.1-Linux-64bit/bin/paraview
```

If you want to use it on other computers, it is part of most linux distributions' repositories or you can get a copy at http://www.paraview.org/. Versions up to 3.13 may contain a bug that causes a memory leak while rendering movies. So if your repositories contain a version up to that, better get the current release from the website.

3.1 Laminar Picture

▷ 1 Point

Create a nice plot of the stationary state of a laminar flow field around the obstacle and hand it in as jpg or png.

You can output the LB velocity field and the boundary geometry from ESPResSo in a paraview compatible format with the following commands:

- `lbfluid print vtk boundary` *`filename`*
- `lbfluid print vtk velocity` *`filename`*

For this tutorial, it will be sufficient to know about the 7 `paraview` controls shown in fig. 2.

1. Load data files into the pipeline for visualization
2. Adjust color settings of the filter selected in the pipeline browser
3. Add one of the most used filters to the element selected in the pipeline browser
4. The pipeline browser – shows loaded data files and filter applied to them for visualization
5. Configuration panel for the filter chosen in the pipeline browser
6. The preview panel – you can rotate, zoom and move this with the mouse
7. Controls for videos

With a little exploration, these controls should be self explanatory. If you need help, consult your tutor, the paraview help (F1) or the paraview online documentation at http://www.paraview.org.

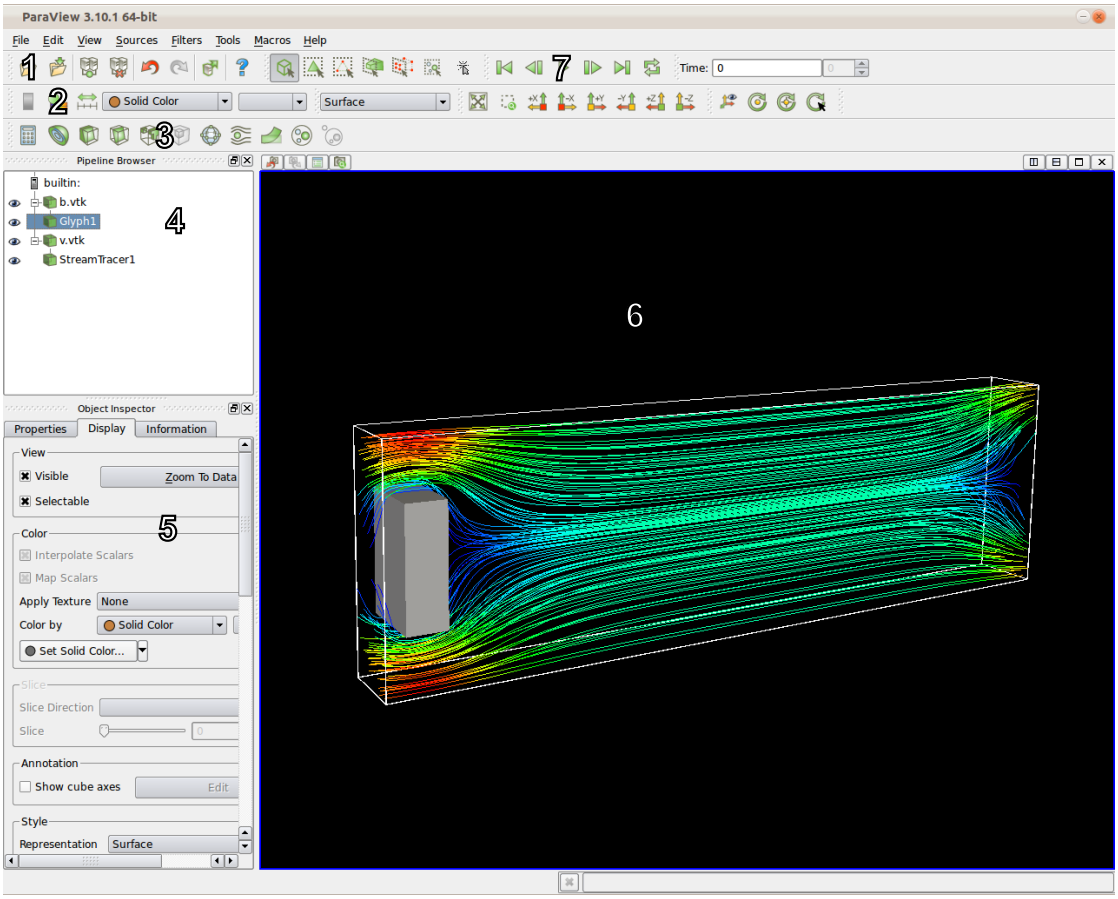

Figure 2: Paraview with a sample visualization and the most important controls.

3.2 Turbulent Video

▷ 1 Point

Create a video of the time evolution of the system with a force density from the upper end. Choose the simulation time between frames such that the video looks smooth and the occuring turbulences are interesting to watch.

Outputting so much data significantly slows down the simulation and requires 6.6 MB per frame. If you don't want to spend that much time on the simulation and the rendering or do not have that much memory available, you can just output a few hundred frames from time step 9000 on.

Creating videos with paraview works the same way as creating images. The only difference is that you have to output one vtk file for every frame in ESPResSo (about one frame every 10 integration steps will do) and include a counter in the file names. You can then open all those files at once with paraview and use the video controls to watch a preview of your animation.

You can save an animation through the "Files" menu. How the video is going to be encoded depends on the codecs available on the machine. Watch out, some codecs only allow certain video dimensions or frame rates (powers of 2).

If the resulting video is too big to hand in, store it on your CIP account and just hand in the path. Also there is an example video available on the website.

4 Unattended Simulations

After having verified that the chosen system actually produces laminar and turbulent flow, we want to quantitatively investigate the flow rate. In section 5.1 you will show that for low forces, the net flow rate scales linearily with the force density. For flow velocities that produce a non neglible convective contribution, however, there is no theory that predicts the scaling for such a complex geometry. We will therefore run the simulation for many force densities ranging from $f = 0.0$ to $f = 0.2$. Since it is a computation time intensive task to obtain data of sufficient resolution, the following sections explain how to automize this process so that no user interaction is required.

4.1 Batched Execution

▷ 1 Point

Change the simulation script such that the force density can be specified as a command line argument. Determine the time needed for one simulation and adjust the given bash script such that it takes between 10 and 20 hours to sample the specified range of force densities.

Running the same simulation several times while varying a parameter can easily be achieved by writing a second program that executes the simulation in a loop and passes the parameter as

a command line argument. A simple bash script will suffice in this case. The command line arguments can be accessed in an ESPResSo TCL script through the list

```
$argv
```

One problem is that the bash itself does not support calculations involving floating point numbers. This can however be fixed by using `bc`. The following example demonstrates how to do this:

```
f = 0
f = `echo $f+0.1 | bc -l`
```

Due to the lack of support for floating point numbers, you will only be able to compare for equality as stings in loops and conditions. Therefore, if more complicated calculations are needed for the batched execution, a full featured language might be better suited for this task.

The given bash script `batch_execute.sh` works in the way described above and also executes the analyzation as described in section 4.2.
Adjust it so that it will run on one of the CIP pool machines for about 10 to 20 hours, sampling the force density range $f \in [0 \dots 0.2]$ in such a way, that about a third of the calculation time is spent on the interval $f \in [0 \dots 0.002]$, a third on $f \in (0.002 \dots 0.02]$ and another third on $f \in (0.02 \dots 0.2]$. Of course you can also prepare several of those scripts to split the work amongst several computers.

4.2 Automatic Analyzation

▷ 1 Point

Change the existing simulation script to calculate the time dependent net flow rate of each run and store it into a file. Complete the existing analyzation so that it reliably calculates the averages asymptotic net flow rate for each run and briefly explain how it works.

The ESPResSo command

```
lbnode x y z print <velocity|boundary|...>
```

allows you to access LB information during the simulation. Change the simulation script so that the simulation is interrupted every 100 steps and the time and the current flow rate is written to a file called `flowrate_$f.dat` with `gnuplot` readable format in a folder `results`.
The flow rate can be calculated by summing up the flow in force direction of all elements on a plane perpendicular to the force direction.

It is important that the determination of the averaged asymptotic flow rate from this time resolved flow rate profile works in a completely automatic fashion for all force densities ranging from $f = 0.00001$ to $f = 0.2$. So first run some simulations in this range (specifically at

the upper and lower limit) by hand to see how long it takes to reach a stationary or oscillatory state and how much time you need for averaging to get a reliable mean value in the oscillatory case. Then adjust the analyzation routine in `batch_execute.sh` and `average.gp` so that it averages over the determined time interval.

4.3 Unattended and Remote Runs

Run your whole setup first with a very small number of force densities. This serves two purposes: First, you might still have bugs in your scripts which you only discover during a full run. Especially when using scripting languages this is a problem, since even syntax errors only become visible the moment the respective line is executed and there is nothing more frustrating than coming back after 12 hours only to realize that the system crashed after 5 minutes.
Second, you need to be able to plan how long your simulations are going to take. Usually you would start with a short run of low resolution to find the ranges of interesting behaviour and subsequently start runs of higher resolution for those intervals.
To shortcut this procedure for you, a good way to sample the given range was already given in section 4.1.

These simulations should take at least between 10 and 20 hours. Usually you can't just block a big number of computers for that amount of time and even if you do, people might just kill your jobs to regain access to the machines. You could log in via ssh to start your jobs without locking the graphical interface but should the connection break during the simulations, your jobs would also get killed.
What you need to do is detach your program from the terminal you used to start it with. The reason for your jobs getting killed is not that you got logged out. The reason is that the input/output of your program is a terminal program that gets terminated when you log out. So if you use something other than a terminal for input and output, your programs keep running. There is a simple way to do this on Unix systems:

`nohup` *`mycommand`* `&`

This starts the program with no input and reroutes the output to a file called `nohup.out` by default. The `&` puts the job to the background. There is no way of interact with a program run that way. The only thing you can do is stop it with `kill`, `killall`, `htop` or `top` (cf. the respective `man` pages for information). Should you ever need to regain access to the input of a program, consider using `screen` instead of `nohup`. `screen` allows you to run complete terminal sessions independently from a specific terminal program. For our purposes `nohup` suffices, though.

Since you can't be present all the time while the simulations run, you might want to be able to start simulations, check on them and pick up results from home. You can easily do this using `ssh`. On Linux and Mac machines this is a terminal command included by default, for Windows machines you can use PuTTY (http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html). `ssh` allows you to access a terminal on a remote machine over an encrypted connection. The

machines in the ICP CIP pool are not directly reachable from the internet, but there is a gateway machine that you can access from any computer located on campus. To gain access from home, first connect to a computer in the physCIP pool (since those machines are actually reachable from the internet and you should have an account there) via

```
ssh phyXXXXX@raiden.physcip.uni-stuttgart.de
```

From there you can establish a connection to the gateway machine at the ICP CIP pool using

```
ssh simXXXX@gate.cip.ica.uni-stuttgart.de
```

Once you have access to the terminal, use it to connect to the machine you want to use with

```
ssh name
```

where *name* is the name of the machine which runs your jobs. You find the name on a label on the case of the computer. Transferring files from and to the CIP pool machines is best done with a graphical file browser. Quite likely the one that came with your Linux distribution allows you to do this. If you are on campus, this can be done by simply accessing

```
sftp://gate.cip.ica.uni-stuttgart.de/students/simXXXX
```

If you use windows, you again have to use an external program such as WinSCP (http://winscp.net/eng/download.php) Also MacOS relies on external programs for this task, e.g. Cyberduck (http://cyberduck.ch/?l=en).

If you are not on campus, this is much more complicated since your file browser will not be able to establish a direct connection. It is possible however, to use a ssh tunnel to make this possible. The command

```
ssh -L 8000:gate.cip.ica.uni-stuttgart.de:22
  phyXXXXX@raiden.physcip.uni-stuttgart.de
```

establishes a ssh connection to `raiden.physcip.uni-stuttgart.de` and forwards all traffic coming in from connections to your local port `8000` through the ssh connection and to `gate.cip.ica.uni-stuttgart.de`, port `22` (the default sshd port). That way, the connection to the gateway machine originates from `raiden.physcip.uni-stuttgart.de`, a machine on campus and is therefore allowed to pass the firewall. You can then access you ICP CIP pool home folder through

```
sftp://localhost:8000/students/simXXXX
```

using you `simXXXX` account.

5 Investigating the Force-Flowrate Relation

5.1 Analytical Theory for low Re

▷ 1 Point

Proof that in the stationary state for low force densities, there is a linear relation between the applied force density and the net volume flux in our system.

Flow of Newtonian fluids is conventionally described by the equation of continuity (1) and the Navier-Stokes equations (2)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0\,, \tag{1}$$

$$\rho \frac{\partial \vec{v}}{\partial t} + \underbrace{\rho\,(\vec{v}\cdot\vec{\nabla})\vec{v}}_{\substack{\text{convective}\\ \text{momentum}\\ \text{transport}}} = -\nabla p + \underbrace{\eta \vec{\Delta}\vec{v}}_{\substack{\text{viscous}\\ \text{momentum}\\ \text{transport}}} + \vec{f}\,, \tag{2}$$

with ρ the density, $\vec{v}$ the velocity and η the viscosity of the fluid. $\vec{\nabla}$ and $\vec{\Delta}$ stand for the in cartesian coordinates component wise gradient and laplacian respectively. By rescaling all the occuring variables, one can eliminate one of the two occuring parameters and formulate (2) in a completely unitless way, depending only on the dimensionless Reynold's number

$$\frac{\partial \vec{v}'}{\partial t'} + (\vec{v}'\cdot\vec{\nabla})\vec{v}' = -\nabla p' + \frac{1}{Re}\vec{\Delta}\vec{v}' + \vec{f}'\,, \tag{3}$$

where $Re = \frac{\rho L \bar{v}}{\eta}$ with L an characteristic length and $\bar{v}$ the average fluid velocity. Check http://en.wikipedia.org/wiki/Reynolds_number#Derivation for more information on the nondimensionalization and Reynold's number. From equation (3), one deduce that for very low Re, e.g. small densities, small systems, low velocities or big viscosities, the viscous momentum transport dominates over the convective one. Assuming this is the case, one can neglect the convective contribution and find the stationary state of an incompressible Newtonian fluid by solving the resulting Stoke's equations

$$\vec{\Delta}\vec{v} = \frac{1}{\eta}\nabla p - \frac{1}{\eta}\vec{f}\,, \tag{4}$$

$$\nabla \cdot \vec{v} = 0\,. \tag{5}$$

Pay attention to the fact that the pressure in this equation is not a given quantity but also a variable. In fact it is what couples the two equations and makes the solutions interesting. As boundary conditions we impose $\vec{v} = 0$, so called no-slip boundary conditions.

From this you should be able to deduce that the resulting $\vec{v}$ scales linearily with the applied force density, as it would be the case for a simple linear, uncoupled differential equation.

5.2 Results from the Simulations

▷ 2 Points

Plot the relation between applied force density and asymptotic average net flow rate. Discuss the result.

Questions you could address are:

- For which range of f does the law deduced in section 5.1 hold?
- Where does the flow field start to really look turbulent?
- Can you propose any law that describes the scaling of the asymptotic average net flow rate on the whole investigated interval of f?
- Is there information in the literature for this behaviour and if yes, is it theoretically deduced and from what or is it experimentally or heuristically obtained?

6 Bonus Questions

▷ 2 Points

Figure out what system the simulation corresponds to in reality. What would a temperature of 1 in simulation units correspond to? What caused the symmetry break in the turbulent simulations?

ESPResSo does not specify a unit system and depending on how one sets certain units, other ones are rescaled. Assuming our system consists of water with density $\rho = 1000\,\mathrm{kg/m^3}$ and kinematic viscosity $\nu = 10^{-6}\,\mathrm{m^2/s}$ and the video you created shows the system in real time – how big is the system?
In the produced video you saw that the flow lost its mirror symmetry in the y-direction after a while although the geometry of the problem and the applied force density this symmetry. What caused this symmetry break in the simulation? What causes it in reality? How could one cause it earlier to maybe save some computational time? Assuming one naively set $k_\mathrm{B}T = 1$, what temperature would that correspond to in the real world?

You can find information about how to figure out the unit system in section 4.1 of the bachelor thesis delivered with this homework.

The deadline for handing in the homework is 03.07.2012 .

DO NOT START A DAY BEFORE THE DEADLINE, THIS TUTORIAL INCLUDES SIMULATIONS WITH CALCULATION TIMES OF UP TO A DAY!