Worksheet 5: Hydrodynamics and the Lattice-Boltzmann-Method

Solutions

Georg Rempfer Johannes Zeman

July 23, 2015

Institute for Computational Physics, University of Stuttgart

Contents

Remarks

The Solutions provided here show possible approaches to solve the tasks from the corresponding worksheet and may not be exhaustive.

1 Introduction

In this worksheet, you will first have to answer some theoretical questions about hydrodynamics. In the remaining practical part, you 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 a factor of 50 depending on the graphics card), so you are better off using the GPU implementation.

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 Short Questions - Short Answers

Task (4 points)

Answer the following questions:

- What could be considered the main advantages of the Lattice-Boltzmann-Method (LBM) over "classical" computational fluid dynamics (CFD) methods (Finite Volume or Finite Elements Navier-Stokes solvers)?
- In which cases does the LBM *not* work?
- What's the meaning of a Reynolds number and how is it defined?
- How is Stokes' law defined and what does it describe?

Solution

- One major advantage of the LBM is its linear scalability in parallel computing, since collisions are computed locally. This makes the LBM an ideal candidate for GPU computing.
  Also, multiphase flows are no big deal with LB. In fact, inter-phase interactions are automatically included in the collisions.
  Furthermore, the LBM facilitates the realization of boundaries for arbitrarily complex geometries.

- The LBM can only be used at small Mach numbers and low compressibilities. Also, the method introduces an implicit compressibility, so that simulations of completely incompressible fluids are not possible.
- The Reynolds number Re describes the ratio of inertial (momentum) to viscous forces in a fluid under given flow conditions. It is defined as $Re = \frac{vL}{\nu}$, where v is the velocity of the considered object/boundary relative to the fluid, L its characteristic length scale ("linear dimension"), and ν is the fluid's kinematic viscosity. At low Reynolds numbers ($Re \ll 1$), where viscous forces dominate, the flow is laminar, while at high Reynolds numbers ($Re \gg 1$), the flow is turbulent.
- Stokes' law describes the drag force (frictional force) exerted on a spherical object in a fluid at low Reynolds numbers. It is defined as

  $$\boldsymbol{F}_d = 6\pi\mu R\boldsymbol{v} \quad ,$$

  where $\boldsymbol{F}_d$ is the drag force acting on a sphere with radius R moving at velocity $\boldsymbol{v}$ through a fluid with dynamic viscosity μ.

---

3 Stokes' Law, Computer Game Superheros and Low Reynolds Numbers

Imagine you are in your living room after a long day of studying. As a refreshment, you just got yourself a nice glass of delicious dihydrogen monoxide (H_2O), as suddenly Duke Nukem kicks in the door, shouts "Hail to the king, baby!", and fires at you with his supercharged shrink ray. Ouch! Within the blink of an eye, you are shrunk to micrometer size and, to complete your misery, fall into your own glass of water.

Fortunately, supercharged shrink rays cause the majority of the target's mass to be temporarily beamed to a parallel universe, so that the mass density of the target remains unchanged. However, you just happen to wear that brand new heavy-duty spherical body armor you got from *www.gimmemoreofthatfreakygeekstuffilikesomuch.com*, so your mass density is about twice the density of H_2O.

Task (5 points)

- Now being a sphere of $2R = 1\ \mu m$ in diameter, how do you expect to fall into the water? Will you plunge in smoothly or will you be stopped rather abruptly? Give reasons for your answer!
- Use Stokes' law in combination with Archimedes' principle to calculate your terminal velocity under water. Assume $\rho_s = 2000\ \frac{kg}{m^3}$ (your density), $\rho_{H_2O} = 1000\ \frac{kg}{m^3}$ (density of water), $g = 9.81\ \frac{m}{s^2}$ (gravitational acceleration), and $\mu = 0.001\ Pa\ s$.
- Why is Stokes' law applicable in this case?
- As you just figured out, without doing anything, you will sink deeper and deeper into the water. Since giving up and drowning is not an option, you start swimming to get back to the surface. Is it better to use butterfly strokes, breast strokes, or freestyle strokes to propel yourself upwards? If you had the choice, would you prefer a paddle or a ship's propeller to facilitate moving?
- Could Stokes' law still be applied in air? ($\mu_{air} = 1.8 \times 10^{-5}\ Pa\ s$, $\rho_{air} = 1.225\ \frac{kg}{m^3}$ and $v \approx 1\ \frac{\mu m}{s}$)

Hint

- In order to get an idea what this task is about, read Purcell's (very entertaining!) paper *Life at low Reynolds number.*
  You can find it under `/group/sm/2015/purcell77a.pdf`.

Solution

- Since you are a micron-sized shpere, your Reynolds number is very low and inertial forces are negligible. This means that your velocity depends entirely on the viscosity of the fluid within which you are moving. Thus, an abrupt change of viscosity also means an abrupt change in velocity, so that you will be slowed down rather abruptly when crossing the air/water interface.
- The terminal velocity is reached as soon as the force F_g caused by gravitational acceleration and the viscous drag force F_d in the fluid cancel out:

$$|F_g| \overset{!}{=} |F_d|$$

According to Archimedes' principle, the gravitational force is reduced by the buoyant force in the fluid, so that

$$|F_g| = (\rho_s - \rho_{H_2O})\, Vg = (\rho_s - \rho_{H_2O})\, \frac{4}{3}\pi R^3 g \quad .$$

Plugging this into Stokes' law and solving for v yields

$$v = \frac{2}{9}\frac{g}{\mu}\,(\rho_s - \rho_{H_2O})\, R^2 \approx \underline{\underline{0.545\frac{\mu\text{m}}{\text{s}}}}$$

- Since you're in your spherical body armor, you are... right, you're spherical. Your Reynolds number is $Re = \frac{2vR\rho_{H_2O}}{\mu} = 0.545 \times 10^{-7}$ (which can be considered as low), so it is justified to apply Stokes' law.
- Using the butterfly stroke, only the whip-like motion of your body will be of use, since the motion of your arms is pretty symmetrical. For breast strokes, the situation is different, since the motion of your arms is not symmetrical. Moving your arms forward, their effective cross section is smaller than in the backward motion, so this should be less exhausting and would still propel you forward.
  Mounting paddles to your armor like in a row boat would only allow completely symmetric motion, so the ship propeller is the better option.
- In air, the Reynolds number for the given values is 6.806×10^{-8}, so Stokes' law is still applicable.

4 Simulation of Hydrodynamic Flow Around a Rigid Object

Now we will go back to more serious problems and perform hydrodynamic simulations using the LBM. 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.

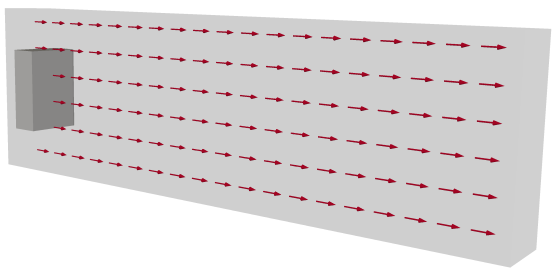

Figure 1: System to investigate. Light gray is the simulation volume with periodic boundaries and the LB fluid. Dark grey represents an obstacle with no-slip boundary conditions and red stands for the applied constant force density

Turbulence is a phenomenon 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-consuming task.

The following set of parameters will work for this task:

- fluid viscosity $\nu = 1.1$
- fluid density $\rho = 1$
- fluid force density $f = 0 \dots 0.2$
- fluid grid spacing $a = 1$
- time step $\tau = 0.01$
- box size $= 200 \times 60 \times 20$
- obstacle dimensions $= 10 \times 30 \times 20$

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

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

Should you want to take advantage of the GPU Lattice-Boltzmann implementation, you need to compile ESPResSo with CUDA support:

```
$> ./configure --with-cuda=/usr/local/cuda
```

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 ESPResSo user's guide if you need further information on those commands. The commands to produce output are given in section 5.1. Also note that although we are not going to use particles, you will have to set a Verlet skin with `setmd skin 0.1`

For the following simulations, you can use the Tcl scripts provided on the course website.

5 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.

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.

5.1 Laminar Picture

Task (2 points)

- Create a nice plot of the stationary state of a laminar flow field around the obstacle and include it in your report. Using a body force density of 0.00001 should produce laminar flow.

Hints

- 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.

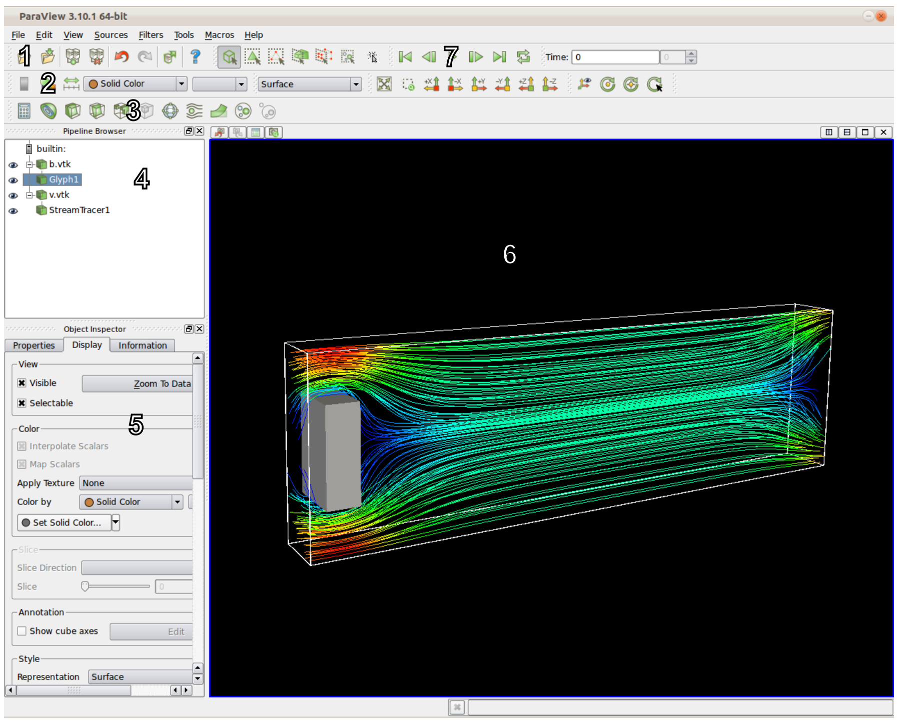

Figure 2: Paraview with a sample visualization and the most important controls.

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.

Solution

Running the following code with ESPResSo will generate the required files for the visualization in paraview:

```
# set parameters
set force 0.00001
set box_x 200
set box_y 60
set box_z 20
set visc 1.10
set n_steps 40000
setmd box_l $box_x $box_y $box_z
setmd time_step 0.01
setmd skin 0.1
thermostat off
# set up LB
lbfluid gpu dens 1 agrid 1 tau $dt visc $visc ext_force $force 0 0
# set up obstacle
lbboundary rhomboid corner 5 [expr $box_y/4] 0 a 10 0 0 \
b 0 [expr $box_y/2] 0 c 0 0 $box_z direction outside
# store geometry of box with obstacle for visualization
lbfluid print vtk boundary b.vtk
# run simulation
integrate $n_steps
# store flow field
lbfluid print vtk velocity v.vtk
```

After loading the files in paraview, adding streamlines and tweaking the visualization parameters, the resulting picture should look similar to figure 3.

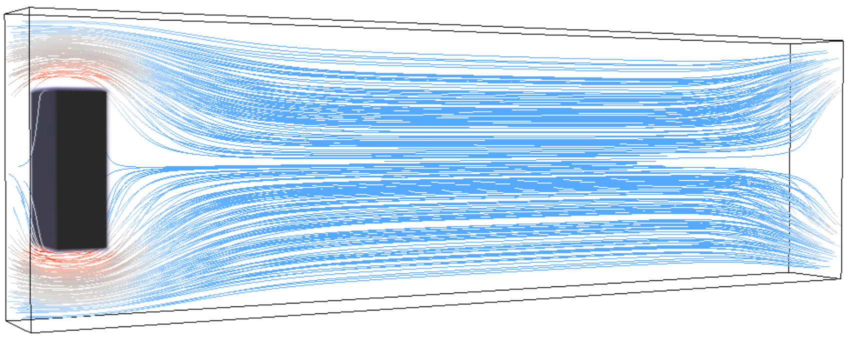

Figure 3: Laminar flow visualization with streamlines.

5.2 Turbulent Video

Task (2 points)

- 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. Using a body force density of 0.2 should produce turbulent flow.

Hints

- Outputting so much data significantly slows down the simulation and requires roughly 6.2 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. On the ICP CIP pool machines, consider running the simulations under `/work/<username>`.
- 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 in ogv format.
- If the resulting video is too big to hand in, store it on your CIP account and just hand in the path. Also, there are example videos available on the website or under `https://youtu.be/A2zCOfrjH-Q` and `https://youtu.be/GYRRGuYoNF4`.

Solution

The following ESPResSo script will do the job:

```
#set up simulation
set box_x 200
set box_y 60
set box_z 20
set force 0.2
set visc  1.10
set n_cycles         4000  ;# measurement cycles
set steps_per_cycle   10  ;# steps per cycle
setmd box_l $box_x $box_y $box_z
setmd time_step 0.01
setmd skin 0.1
thermostat off
# set up LB
lbfluid gpu dens 1 agrid 1 tau $dt visc $visc ext_force $force 0 0
# set up obstacle
lbboundary rhomboid corner 5 [expr $box_y/4] 0 a 10 0 0 \
b 0 [expr $box_y/2] 0 c 0 0 $box_z direction outside
# store geometry of box with obstacle for visualization
lbfluid print vtk boundary b.vtk
# run simulation
for {set cycle 0} {$cycle < $n_cycles} {incr cycle} {
    integrate $steps_per_cycle
    lbfluid print vtk velocity v_$cycle.vtk ;# video frame
}
```

6 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 7.1 you will show that for low forces, the net flow rate scales linearily with the force density. For flow velocities producing 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 $f = 0.00001$ to $f = 0.2$. Since it is a computation time intensive task to obtain data of sufficient resolution, you should automize this process so that no user interaction is required.

Task (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 a couple of hours to sample the specified range of force densities.

Hint

- Command line arguments can be accessed in an ESPResSo Tcl script through the list `$argv`, and the number of passed arguments can be accessed through `$argc`.

Solution

The following code snippet reads in a command line argument and stores it in a variable `force`. If no argument is given, a usage message is printed to the screen.

```
set force 0.0
if { $argc > 0 } then {
    set force [lindex $argv 0]
} else {
    puts "USAGE:␣Espresso␣template.tcl␣FORCE"
    exit 1
}
```

6.1 Batched Execution

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.

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

```
f=0
f=$(awk "BEGIN{printf \"%e\", $f + 0.00001}")
```

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

The given bash script `batch_execute.sh` works in the way described above. Adjust it so that it will run on one of the CIP pool machines for a couple of hours, sampling the force density range $f \in [0.00001 \ldots 0.2]$. Of course you can also prepare several of those scripts to split the work amongst several computers.

6.2 Automatic Analyzation

Task (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 average asymptotic net flow rate for each run and briefly explain how it works.

Hints

- 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.

- Computing averages of time series can be done directly in the simulation script. Use the tcl command `lappend` to append measured values to a list and use the function `uwerr` to compute the corresponding average and statistical error.
- 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. Implement an equilibration routine before the measurement run during which no flow rate data is recorded. If needed, adjust the routine in `batch_execute.sh` gathering the computed averages.

Solution

The following script will work:

```
# read force from command line
set force 0.0
if { $argc == 1 } then {
    set force [lindex $argv 0] ;# 0.2 for turbulent video
} else {
    puts "USAGE:␣Espresso␣template.tcl␣FORCE"
    exit 1
}
#set up simulation
set box_x 200
set box_y 60
set box_z 20
set visc  1.10
set dt    0.01
set eq_cycles        800  ;# equilibration steps
set n_cycles         800  ;# measurement cycles
set steps_per_cycle  100  ;# steps per cycle
setmd box_l $box_x $box_y $box_z
setmd time_step $dt
setmd skin 0.1
thermostat off
# set up LB
lbfluid gpu dens 1 agrid 1 tau $dt visc $visc ext_force $force 0 0
# set up obstacle
lbboundary rhomboid corner 5 [expr $box_y/4] 0 a 10 0 0 \
b 0 [expr $box_y/2] 0 c 0 0 $box_z direction outside

# equilibration routine
```

```
for {set cycle 0} {$cycle < $eq_cycles} {incr cycle} {
    integrate $steps_per_cycle
}

# open flow rate output file
set flowrate_file [format "flowrate_f%10.8f.dat" $force]
set fp [open $flowrate_file "w"]
puts $fp "#time␣flow_rate"

# run measurement cycles
set flowrate_list ""
for {set cycle 0} {$cycle < $n_cycles} {incr cycle} {
    integrate $steps_per_cycle
    # calculeat flow rate
    set flowrate 0.0
    for {set y 0} {$y < $box_y} {incr y} {
      for {set z 0} {$z < $box_z} {incr z} {
        set flowrate [expr $flowrate + \
        [lindex [lbnode 100 $y $z print velocity] 0]]
      }
    }
    # store flow rate
    set t [expr $cycle * $steps_per_cycle * $dt]
    puts $fp "$t␣$flowrate"
    lappend flowrate_list "$flowrate␣"
}
# flush buffer and close flow rate file
flush $fp
close $fp

# run analysis
set uwerr_frate [uwerr $flowrate_list [llength $flowrate_list] 1]
set mean_frate [lindex $uwerr_frate 0]
set error_frate [lindex $uwerr_frate 1]
# print the values to screen
puts [format "flow_rate␣=␣%17.10e" $mean_frate]
puts [format "flow_rate_error␣=␣%17.10e" $error_frate]
```

7 Investigating the Force-Flowrate Relation

7.1 Analytical Theory for low *Re*

Task (2 points)

- Prove 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.

Hint

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

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

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

with density ρ, velocity $\vec{v}$ and viscosity of the fluid η. $\vec{\nabla}$ and Δ 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 Reynolds number:

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

Check `http://en.wikipedia.org/wiki/Reynolds_number#Derivation` for more information on the nondimensionalization.
From equation (3), one deduces 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 Stokes' equations

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

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

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.

Solution

For low Reynold's numbers Stokes' equations (4),(5) are a valid approximation for the Navier-Stokes-Equations (1),(2).

Suppose $\vec{v}$ is a solution of Stokes' equations with body force density $\vec{f}$. Then, by multiplying with a constant factor α and using the linearity of $\vec{\nabla}$ and $\vec{\cdot}$, we obtain

$$\begin{aligned}\Delta\left(\alpha\vec{v}\right) &= \frac{1}{\eta}\vec{\nabla}\left(\alpha p\right) - \frac{1}{\eta}\left(\alpha\vec{f}\right)\,,\\ \vec{\nabla}\left(\alpha\vec{v}\right) &= 0\,,\end{aligned}$$

which means that $\alpha\vec{v}$ is a solution to Stokes' equations with a body force $\alpha\vec{f}$. The resulting pressure is just scaled by α as well. Obviously, where $\vec{v} = 0$, it is also true that $\alpha\vec{v} = 0$, and therefore $\alpha\vec{v}$ fulfills the same no-slip boundary conditions that $\vec{v}$ originally fulfilled.

This means that if one obtains a solution for a certain body force density, in the limit of low Reynolds numbers, all solutions for other body force densities can be obtained by just scaling the original solution appropriately. Furthermore, this also means that in the limit of low force, there is a linear relation between force density and flow rate with a geometry-dependent proportionality constant.

7.2 Results from the Simulations

Task (3 points)

- Plot the relation between applied force density and asymptotic average net flow rate and discuss the result.
- For which range of f does the law deduced in section 7.1 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?

Hint

- In order to make the different regimes clearly visible, you might want to show your results in a log-log plot.

Solution

- The results of the simulations are plotted below:

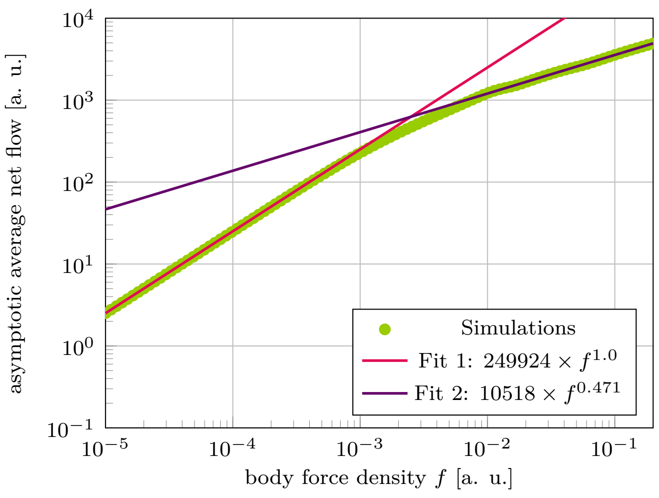

Figure 4: Average asymptotic flow rate as a function of applied body force density. Green dots (●): simulations, red line (——): fit to lower 20% of data, purple line (——): fit to upper 20% of data.

- The proprotionality deduced in the previous task is clearly reproduced in the first part of the data up to about $f = 3 \times 10^{-4}$ (compare fig. 4 fit 1 (—)).
- From figure 4 it is not possible to draw conclusions at which body force density the fluid really shows turbulent flow. In order to answer the question, one has to investigate the time series of flow rates for different body force densities, including equilibration:

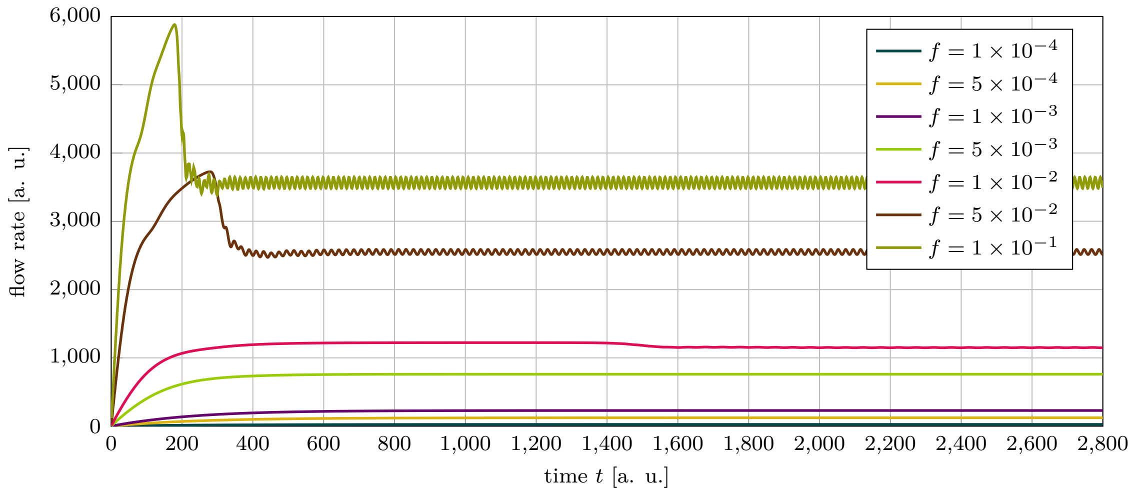

Figure 5: Flow rate over time for different body force densities f. At high force densities (— and —), the transition from laminar to turbulent flow occurs early is clearly visible. At low force densities (—, —), the flow stays laminar as expected. At intermediate force densities (— and —), however, the flow stays laminar up to $f = 0.005$, even though this is already far in the transitional regime. Note the small but visible change from laminar to turbulent flow for $f = 0.01$ (—) at $t \approx 1500$.

Figure 5 shows the flow rate over time for different body force densities f. While for high values of f, the transition from laminar to turbulaent flow occurs early and is clearly visible, the flow stays laminar for $f \leq 0.005$, even though, according to figure 4, this value already lies far in the transitional regime. For $f = 0.01$ (—), the transition to turbulent flow occurs late ($t \approx 1500$) and is not as pronounced as for higher values of f.

Thus, the flow field starts to look turbulent at $0.005 < f < 0.01$.

- For the laminar regime, the asymptotic average flow rate is proportional to the applied body force density and in the turbulent regime it shows a square root behavior, with a smooth transition in between. However, a unified formulation for the whole investigated range of force densities cannot be given.

- The square root dependence is experimentally well tested and there are a lot of older, heuristic works on it:
`http://en.wikipedia.org/wiki/Darcy%E2%80%93Weisbach_equation`
However, turbulence is one of the phenomena for which physics does not provide a complete description and so there is no derivation from first principles.

---

8 Bonus Questions

Task (3 points)

- 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 using 2000 frames with a spacing of 10 time steps (of $0.01\,t$ each) in between and a frame rate of 60 fps – how big is the system?
- Assuming one naively set $k_\mathrm{B}T = 1$, what temperature would that correspond to in the real world?
- 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 is symmetric. 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?

Hint

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

Solution

- First, the time scale of the simulation has to be determined. The duration of the video in seconds is
$$t_\mathrm{video} = \frac{2000\,\mathrm{frames}}{60\,\mathrm{fps}} = \frac{100}{3}\,\mathrm{s}\,, \tag{6}$$
which corresponds to a simulation time of
$$t_\mathrm{sim} = 2000\,\mathrm{frames} \times 10\,\frac{\mathrm{steps}}{\mathrm{frame}} \times 0.01\,\frac{t}{\mathrm{step}} = 200\,t\,. \tag{7}$$

Requiring $t_{\text{video}} \overset{!}{=} t_{\text{sim}}$ yields the time scale

$$[t] = \frac{100}{3}\,\mathrm{s} \times \frac{1}{200} = \frac{1}{6}\,\mathrm{s}\,. \tag{8}$$

Next, we know that the kinematic viscosity of water is $\nu = 10^{-6}\,\mathrm{m}^2/\mathrm{s}$, so that $\nu = \frac{[x]^2}{[t]}$ yields the length scale

$$[x] = \sqrt{\nu \times [t]} \approx 0.408\,\mathrm{mm}\,. \tag{9}$$

In the simulation, the dimensions of the box were set to $L_x = 200\,, L_y = 60\,, L_z = 20$, thus corresponding to $L_x = 8.165\,\mathrm{cm}\,, L_y = 2.449\,\mathrm{cm}\,, L_z = 0.816\,\mathrm{cm}$. The box volume thus corresponds to

$$V = \underline{\underline{1.632 \times 10^{-5}\,\mathrm{m}^3}}\,. \tag{10}$$

- Now that we know the length scale $[x]$ of the system, we can get its mass scale from the density of water:

  $$[m] = \frac{\rho}{[x]^3} \approx 0.068\,\mathrm{mg} \tag{11}$$

  This allows us to answer the question about setting the temperature to 1: ESPResSo implicitly sets $k_B = 1$, so that with $[E] = [m][x]^2/[t]^2 = k_B T$ one obtains

  $$T \approx \underline{\underline{2.95 \times 10^{10}\,\mathrm{K}}}\,. \tag{12}$$

  This seams reasonable, considering that simulations at this temperature show macroscopic density fluctuations which do definitely not occur at room temperature.

- The symmetry break comes from the fact that the algorithm iterates over the grid in a certain order, which does not preserve the symmetry. The actual difference is then caused by the fact that in floating point arithmetic, adding numbers of different order of magnitude causes a different error, depending on the order of summation. *(Small + small) + big* gives a smaller error than *(big + small) + small*, since for the latter, the accuracy is already lower after the first addition.

  In reality, this is a flow of microscopic particles whose dynamics are to some degree chaotic, leading to local fluctuations causing the symmetry break. This is a phenomenon essentially caused by a non-vanishing temperature. One could incorporate random fluctuations into the Lattice-Boltzmann algorithm to account for this. In fact, ESPResSo's implementations of LB provide this thermalization.
