HANDS-ON EXERCISE FOR THE LECTURE

Multiparticle Collision Dynamics: A method to unravel the hydrodynamic properties of active matter

Alberto Medina, Joscha Mecke, Marisol Ripoll

Theoretical Physics of Living Matter, Institute of Biological Information Processing, Forschungszentrum Jülich, 52425 Jülich, Germany

January 28, 2021

Introduction

This is an introductory exercise to the multiparticle collision dynamics (MPC) simulation technique. This text and the exercise are complementary to the lecture, to the corresponding notes, and to all the references there included. The exercise tries to adapt to the different background of the PhD students attending this course, and to any other potential interested reader. Some parts of the exercise are therefore simpler, and some others more involved, but still accesible. We leave the deeper understating of the code details to the most interested and expert participants. The main idea is that the principles of the method are revised and become more clear. *We hope you enjoy it.*

Note that: **To do this exercise, you need a C compiler and python 2 or higher**
C compilers: gcc (GNU), Visual Studio (Win), free version of intel icc (Win/GNU/Mac), ...

Physical frame

The code coming with this exercise contains a simple but complete MPC program in two-dimensions. The idea is to study the fluid behaviour in a microchannel geometry, for which periodic boundary conditions in x-direction and walls in y-direction at $y = 0$ and $y = L_y$ are considered. To generate fluid in motion, a constant force g, "gravity", is applied to all particles in the direction parallel to the walls.

The MPC method is a mesoscopic approach designed to simulate fluid hydrodynamics. The transport properties can be analytically calculated by means of kinetic theory (see the lecture notes). In particular, viscosity is the sum of two contributions, $\eta = \eta_{\rm kin} + \eta_{\rm col}$. The kinetic contribution $\eta_{\rm kin}$ refers to the momentum transfer due to the translation of the particles, while the collisional contribution $\eta_{\rm coll}$ refers to the momentum transferred over multiple particle collisions. In two dimensions these expressions are,

$$\begin{aligned} \eta_{\rm kin} &= \frac{\rho k_{\rm B} T h}{a^2} \left\{ \frac{\rho/\left(1-\cos 2\alpha\right)}{\rho - 1 + e^{-\rho}} - \frac{1}{2} \right\}, \\ \eta_{\rm col} &= \frac{m\left(1-\cos\alpha\right)\left(\rho - 1 + e^{-\rho}\right)}{12h}, \end{aligned} \tag{1}$$

where h is the MPC collision time, ρ is the fluid density, α is the rotation angle, a is the collision box length, m is the particle mass and $k_{\mathrm{B}}T$ is the thermal energy (typically, the collision box length, particle mass, and system average temperature are chosen to be the systems units, *i.e.*, $a = 1 = m = \mathrm{k}_B T$). Note that the derivation of the transport properties required some approximations, as discussed in the lecture notes.

The two dimensional fluid flow in the microchannel satisfies the Navier-Stokes equation, which in the case of small Reynolds number (small velocities and laminar flow) reduces to the *Stokes equation*, and in this case further simplifies to

$$\partial_y^2 v_x = \frac{1}{\eta} g, \tag{2}$$

once the system has reached the steady-state. Hence, the resulting flow field $\boldsymbol{v}(\boldsymbol{x})$ is constant along the x-direction and only depends on the y-direction. Integrating equation eq. (2) and fixing the integration constants with the no-slip boundary conditions $v_x(0) = 0$ and $v_x(L_y) = 0$ yields a parabolic profile. Exercises 1 and 2 you will check among other things, these profiles and their relation with the viscosity values.

An important requirement for the simulation method is how to couple the interaction of a solute (typically large object) to the fluid, *e.g.*, a colloid or a polymer. In the lecture we have seen the most standard methods, which is to consider the object as a heavy MPC particle, or to interact with Molecular dynamics or the surface boundary. In the last exercise we propose an interesting alternative method, which on a microscopic level, might be regarded as unphysical, although on a coarse-grained level, yields the desired behaviour.

Technical code comments

For a start, it might be a good idea to **copy your files to a working directory** so that a pristine copy to revert to always remains.

The simulation program can be compiled as `gcc ./mpc_2d.c -o ./mpc_2d.exe`. Once you have the executable, it can be run as `./mpc_2d.exe`. The parameters are read in from the file `system_parameters.h`. You do not have to recompile after changing the parameters, but only after changing the source code. For modifying the input parameters and the source code, you can use whatever text editor you wish.

The simulation results are visualized with python using the plotting library `matplotlib`. You may find prepared plotting scripts in the folder `output/`. Once your simulation finishes, you can plot the results by typing `./plot_flowprofile.py` or `./plot_flowfield.py`. The graphs will then be saved to `./flowprofile.pdf`, `./flowfield.pdf`, respectively. If this file already exists then it will be overwritten, so you need to rename them if you would like to save them permanently. If you change the system properties (channel size, fluid density, etc.) in the input file, you will have to edit these files as well in the python scripts so that the visualisation follows suit.

Exercises

Exercise 1 The idea in this exercise is to become more familiar with the method and the related variables. In this way, you only need to modify the input parameters to understand the fluid behaviour.

1. Compile and run the simulation. Plot the flow field and the flow profile. Is the latter parabolic?
2. Experiment on changing the equilibration time and the total simulation time. What happens?
3. Measure the velocity profile for different fluid densities, time steps, and applied external forces. The theoretical profile is here measured with eq. (1). When do and when do not the measured and theoretical profiles agree? Why?
4. Switch off the grid shift by editing the grid-shift parameter in `system_parameters.h` and run the simulation again. What happens?

In the seconde part of this exercise, we model an immobile obstacle, *e.g*, a fixed colloid, by consodering a number of heavy fixed fluid particles which take part in the collision step. These collisions as very high friction in the obstacle region. The internal degrees of freedom of the obstacle are updated using a molecular dynamics scheme. During the streaming step, the fluid particles may still enter the obstacle.

5. Add an obstacle to the flow by editing the respective value in `system_parameters.h` to, *e.g.*, 5. Plot the flow field and the flow profile. What happens? Try different radii.
6. With an obstacle present, turn off the grid-shift and rerun the simulation. What happens? Can you find the physical explanation for these different behaviors ?

Exercise 2 The idea now is to understand a bit more in depth the method and the code, such that you also have to modify the source code of the simulation, compile and run.

An unfortunate researcher attempted to optimize the code a little bit, but got something wrong. It is your task to fix it in this exercise. Each time that you change the simulation code you have to recompile. (Note the hints at the end of the exercise)

1. Look at the source code, study its structure and identify where the MPC streaming and collision steps take place. The called functions `md()`, `stream()` and `collide()` can be found in the file `modules/mpc_routines/mpc_routines.h`.
2. The collision step is broken: The velocities in the given cell always rotate counter-clockwise. What we want is that within a given collision cell, the velocities rotate in the same direction. Whether this is clockwise or counter-clockwise for a given cell should be randomly chosen, both with equal probability. Fix the collision step, recompile and run the simulation and look at the flow profile.
3. The fluid is now at thermal equilibrium, this is just resting. Boring ? Implement a gravity-type of force: A force, proportional to the global variable `grav` should accelerate every fluid particle in the positive x-direction. How does this change the flow profile?

4. The friction between the fluid and the wall is not quite OK. If you look at the source code, you will see that the fluid particles bounce off the walls like billiard balls bounce off the side of the pool table, so on average they move fast close to the wall. This is the slip boundary condition. What we want is the no-slip boundary condition, in which the average velocity of the fluid close to the wall should be zero. For this, the momentum component of the bouncing fluid particles parallel to the wall should also be reversed, *i.e.*, the fluid particles fly back towards the direction they were coming from. Fix the boundary condition and study the flow profile.

5. The correct fluid-particle bouncing is not enough to produce a no-slip boundary condition. Also so-called virtual wall particles are needed. They have already been implemented but a final `#define VIRTUAL_PARTICLES` is missing in `mpc_2d.c` before the main function. Find the part of code implementing the virtual particles, and study it briefly. What does it seem to do and how does this affect the flow profile?

6. Now everything should be fixed. Run a couple of simulations with the parameters you used in exercise 1 to check that everything works fine. To this end, compare your results with those of exercise 1.

Hint 1 In 2 (b), compare the two-dimensional rotation matrices which cause a rotation counter-clockwise and clockwise about an angle α or $-\alpha$.

$$\boldsymbol{R}_{\text{ccw}}(\pm\alpha) = \begin{pmatrix} \cos\alpha & \mp\sin\alpha \\ \pm\sin\alpha & \cos\alpha \end{pmatrix} \tag{3}$$

The macro `RND1` gives you a random floating point number in the range $[0.0, 1.0)$. The main function `main()` can be found in `mpc_2d.c`.

Hint 2 In 2 (c), you have to modify the streaming step. A standard integration scheme is the so-called Velocity-Verlet integration. The scheme is as follows

1. Calculate $\boldsymbol{v}(t + \frac{\delta t}{2}) = \boldsymbol{v}(t) + \frac{1}{2}\boldsymbol{a}(t)\delta t$
2. Calculate $\boldsymbol{x}(t + \delta t) = \boldsymbol{x}(t) + \boldsymbol{v}(t + \frac{\delta t}{2})\delta t$
3. Calculate $\boldsymbol{a}(t + \delta t)$ form the interactions
4. Calculate $\boldsymbol{v}(t + \delta t) = \boldsymbol{v}(t + \frac{\delta t}{2}) + \frac{1}{2}\boldsymbol{a}(t + \delta t)\delta t$.

This can be significantly simplified considering that the acceleration $\boldsymbol{a}(t)$ is constant

Acknowledgments

The very first version of this exercise was prepared already in 2006 by Marisol Ripoll, and since them several generation of students of the group in Jülich have used, iteratively strongly modified, and very much improved it until its present, and most likely not yet final form. Thanks to all those that already helped and to you for your interest. Any suggestion you might have for further improvement is also most welcome.