Electro-Osmotic flow

This part is practical. It is concerned with the movement of ions in an charged slit pore. It is similar to the systems that are discussed in the Bachelors thesis of Georg Rempfer which is recommended reading. A slit pore consists of two infinite charge walls as shown in the figure to the right. In this exercise you should simulate such a system with ESPResSo. You are supposed to use a Lattice Boltzmann fluid coupled to explicit ions which are represented by charge Week-Chandler-Anderson spheres. In addition to the charge on the walls, the ions are also subject to an external electrical field parallel to the walls. Electrostatics should be handled by the P3M algorithm with ELC. A set of realistic parameters and an more in detail description of the system can be found in the thesis. You should measure the flow profile of the fluid and the density and velocity profiles of the ions. The case of the slit pore can be solved analytically either in the case of only counter ions (the so called salt free case) or in the high salt limit (Debye-Hueckel-Limit). Calculate the ion profiles in one or both of these cases and compare the results with the simulation.

Background

Electro-Osmotic flow (EOF) is a hydrodynamic flow created by the movement of ions driven by an external field. See your tutor for general info on EOF.

Some reading

- (info on LB, implementation of boundaries) General part and parts 1,2,3 4 & 6 of the ESPResSo Lattice-Boltzmann tutorial [https://github.com/espressomd/espresso/tree/python/doc/tutorials/04-lattice_boltzmann on github]
- (info on P3M, ELC and implementation) General part and part 7 of the ESPResSo Charged Systems tutorial [https://github.com/espressomd/espresso/tree/python/doc/tutorials/charged_system on github]
- (info in EOF) General and part 2 of the Electrokinetics tutorial [https://github.com/espressomd/espresso/blob/python/doc/tutorials/07-electrokinetics/07-electrokinetics.pdf]
- Georg Rempfer, “Lattice-Boltzmann Simulations in Complex Geometries”, 2010, Institute for Computational Physics, Stuttgart

Helpful scripts

These files can be found under ESPResSo’s ./samples/ directory

- constraints.py create repulsive planar surfaces

- visualization_constraints.py create repulsive planar surfaces
- p3m.py 3D period electrostatics
- minimal-charged-particles.py also 3D periodic electrostatics
- lbf.py initialize a LB fluid
- lbf.py fix a particle in space
- visualization_poisseuille.py visualize your sims (good for development/debugging)
- visualization_poisseuille.py planar LB boundaries
- h5md.py output particle data in a h5MD file format

These files can be found under ESPResSo's ./docs/tutorials/ directory

- /02-charged_system/scripts/nacl_units_confined.py uses ELC for proper quasi 2D geometry
- /04-lattice_boltzmann/scripts/poisseuille.py Poisseuille LB flow in planar geometry

Tasks

You will incrementally build the simulation script while making sanity checks along the way. Use the provided sample scripts for help with the syntax and don't forget about ESPResSo Users guide. While building your system, you should be careful about keeping the simulation time short (the development stage). Once you are satisfied with your script you can then try better (more expensive) simulations (the production stage).

- Create free particles between two plates
  - inspect for correctness (visualization is fine, but provide a screenshot)
- Add embedded particles
  - inspect for correctness (visualization is fine, but provide a screenshot)
- Add electrostatics with P3M (don't expect valid results, P3M is not suited for 2D systems)
  - inspect for correctness (visualization is fine, but provide a screenshot)
- Correct the electrostatics with ELC
  - inspect for correctness
  - validate with Poisson-Boltzmann prediction
- Add HI with LB
  - inspect for correctness (validate expected statistical behaviour)
- Looking further
  - EOF is characterized by a flat flow profile, explain your results
  - name three ways your sims can be modified to attempt to achieve a flat profile
  - try one of them (be clever, stay away from situations which are computationally expensive)

Provide streamline scripts - Your tutor should be able to run your code and reproduce your final plot.

Common pitfalls

There are many parameters to choose, many will seem arbitrary but they can make or break your sims. Here are some good starting points, do not change these unless you know what you are doing.

Integrators

- system.time_step = 0.01
- system.cell_system.skin = 0.4 (you can tune this later)
- if/when you use the Langevin thermostat, keep gamma=1.0: system.thermostat.set_langevin(kT=kT, gamma=1.0)
- if/when you want to transition from LD to LB:
  - turn off LD system.thermostat.turn_off()
  - remove existing particle momentum: system.galilei.kill_particle_motion()
  - remove existing CM momentum: system.galilei.galilei_transform()
- if/when you want to use LB, use these parameters lbf = lb.LBFluidGPU(agrid=1.0, dens=1.0, visc=1.0, tau=0.01, fric=20.0)
- remember to set the a fluctuating LB fluid with system.thermostat.set_lb(kT=kT)

Electrostatics

- for P3M electrostatics, use an accuracy of 1e-4, p3m = electrostatics.P3M(prefactor=l_bjerrum, accuracy=1e-4)
- for ELC, use maxPWerror of 1e-4, elc = electrostatic_extensions.ELC(gap_size=elc_gap, maxPWerror=1e-4)
- it is a good idea to warmup (integrate for a few timesteps) your system everytime you add a new feature
- in simulation units, you can use l_bjerrum=2.0 as a starting point

Geometry and constraints

- visualization of vtf and h5md files do not show constraints, you need to trust your code or use online visualisation
- constraints with a potential actually occupy space, find out the accessible volume to the ions
- use a cubic box and have some empty space between the plates and the sim boundary, use elc_gap accordingly (talk to your tutor).

Fail early and fail often

- do NOT try to build your script in one go. Take small steps and validate as often as possible
- use the online visualization, use H5MD or VTF files to analyze the trajectory AFTER the sims. Try not to overuse on-the-fly analysis (it is risky if you fail). Most observables can be extract form the trajectories after the fact.