Free-Solution DNA electrophoresis

Background

General

You are to investigate the motion of polyelectrolytes in bulk solution. Hydrodynamic interactions cannot be neglected here so you will use a Lattice Boltzmann fluid coupled to explicit ions which are represented by charge Week-Chandler-Anderson spheres. The salt/counter-ions as well as the charges on the polyelectrolyte backbone are subject to an external electrical field. Electrostatics should be handled by the P3M algorithm A set of realistic parameters and an more in detail description of the system can be found in the attached papers.

You should measure the DNA mobility from its total displacement from being driven by a weak external field. Verify how the mobility changes with DNA length (N) for very small chains and compare the results with the published simulation and experimental results.

You will also use a separate set of simulations and to to obtain the zero-field mobility from the Green-Kubo relation. Compare your results between the two approaches.

Some reading

- General part and parts 5 of the ESPResSo Lattice-Boltzmann tutorial [https://github.com/espressomd/espresso/tree/python/doc/tutorials/04-lattice_boltzmann on github]
- K. Grass and U. Böhme and U. Scheler and H. Cottet and C. Holm. "Importance of Hydrodynamic Shielding for the Dynamic Behavior of Short Polyelectrolyte Chains". Physical Review Letters 100(096104), 2008.
- K. C. Grass. "Towards realistic modelling of free solution electrophoresis: a case study on charged macromolecules". PhD thesis, Goethe-Universität Frankfurt am Main, 2008.
- [https://www.icp.uni-stuttgart.de/~icp/mediawiki/images/a/ad/MMSD07-poster.pdf poster

Helpful scripts

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

- p3m.py 3D period electrostatics
- diffusion_coefficient.py uses the velocity-autocorrelation function
- minimal-charged-particles.py also 3D periodic electrostatics
- minimal-polymer.py creates a polymer

- lbf.py initialize a LB fluid
- visualization_charged.py visualize your sims (good for development/debugging)
- h5md.py output particle data in a h5MD file format
- electrophoresis.py electrophoresis is a charged polyelectrolyte
- diffusion_coefficient.py calculate a single particle diffusion coefficient

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

- /04-lattice_boltzmann/scripts/polymer_diffusion.py Polymer diffusion in LB
- /04-lattice_boltzmann/scripts/single_particle_diffusion.py single particle diffusion in LB
- /05-raspberry_electrophoresis/scripts/simulate_raspberry_electrophoresis.py electrophoresis of a raspberry particle

Tasks

Theory

You should explain the difference between free-draining and non-free draining polymer. You should explain the Einstein–Smoluchowski relation in the context of electrophoretic mobility. Know when/if is applicable to charged polyelectrolytes.

Simulations

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). the simulations results will get better with more statistics. You should set up the simulations so you can immediately obtain results (which should progressively get better over time).

- create an equilibrated real polymer, equilibrate
- add mobile particles, equilibrate
- add electrostatics with P3M, equilibrate
- add HI with LB, equilibrate
- add external force to all charged particles (E-field), equilibrate
- use the same scripts, exploit command-line arguments, use Condor
- continuous simulation strategy
- create scripts that analyze a inflow of results and refine errorbars of mobility

- as more simulation results pour in, your plots should get better

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)
- in simulation units, you can use l_bjerrum=2.0 as a starting point

Fail early and fail often

- do NOT try to build your script in one go. Take small steps and validate as often as possible
- to get good statistics you will need a lot of data, so think ahead. File sizes can become quite large (or writing to disk can slow down the sims).
- once you are satisfied with your script, you need to transition to from development mode to production mode: only output the strict minimum.
- Exploit online analysis:
  - velocity autocorrelation,
  - center-of-mass position,