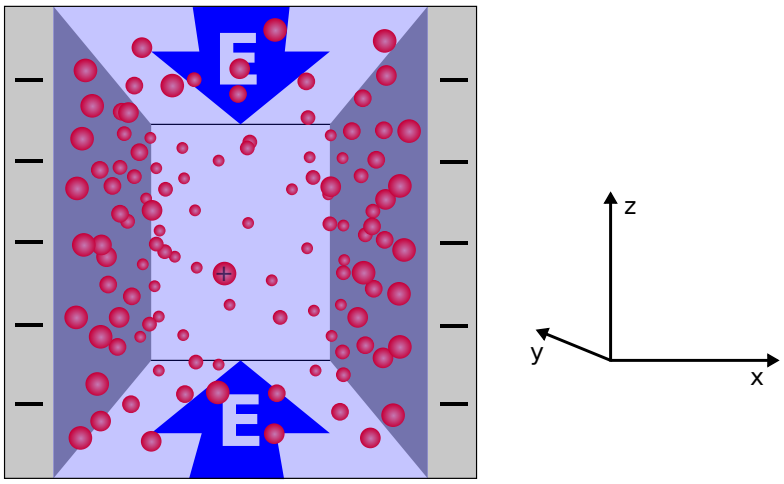
ESPResSo Tutorial

The GPU Electrokinetics method in ESPResSo: Electroosmotic Flow

July 27, 2017

Institute for Computational Physics, University of Stuttgart



Contents

1 Introduction

In recent years the lattice-Boltzmann method (LBM) has proven itself to be a viable way to introduce hydrodynamic interactions into coarse-grained MD simulations with moderate computational cost. The success of the GPU LBM implementation in **ESPResSo** and similar developments in other software packages created demand for further developments in this area. **ESPResSo** features two such algorithms, namely ELECTROHYDRODYNAMICS, and ELECTROKINETICS (EK). Both of these make use of the LBM and extend it to coarse-grain not only the solvent molecules but also ionic solutes. ELECTROHYDRODYNAMICS does so using a slip layer coupling for charged particles valid in the thin Debye layer (large salt concentration) limit, while EK explicitly treats the ionic solutes in a continuum fashion and is valid for a wide range of salt concentrations.

Tutorial Outline

To make our first steps using ELECTROKINETICS we will work on one of the few systems for which analytic solutions for the electrokinetic equations exist – the slip pore geometry with a counterion-only electrolyte. The same slit pore system is also treated in the LBM tutorial, but there, the ionic species were modeled as explicit particles. For this system, the two approaches lead to exactly the same results [1]. Differences are only becoming significant for multivalent ions, very high salt concentrations, and very high surface charges.

This tutorial is divided into three sections. The first section 2 introduces the electrokinetic equations and the analytical solution for the slit pore system, while the second section 3 deals exclusively with the simulation, its setup, and the results.

If you already know about simple diffusion-migration-advection equations, continuum electrostatics, and Navier-Stokes, then you can skip the first section.

2 Theoretical Background

2.1 The Electrokinetic Equations

In the following, we will derive the equations modeling the time evolution of the concentrations of dissolved species as well as the solvent in the standard electrokinetic model. We do so, neglecting the salt ions' contribution to the overall mass density, which allows us to treat the dynamics of the ions and the fluid separately [2]. The solvent fluid will be modeled using the Navier-Stokes equations while we use a set of diffusion-migration-advection equations for the ionic species.

Ionic Species

The description starts with the ionic species' concentrations $c_k(\boldsymbol{r}, t)$ (number density) and the associated flux densities $\boldsymbol{j}_k(\boldsymbol{r}, t)$, for which mass conservation holds

$$\partial_t c_k = -\nabla \cdot \boldsymbol{j}_k. \tag{1}$$

Here $\boldsymbol{r}$ denotes the spatial coordinate and t the time, while k enumerates the ionic species. The fluxes are caused by diffusion (due to density variations and external forces) and advection.

The advective contribution to the flux is given by

$$\boldsymbol{j}_k^{\text{adv.}} = c_k \boldsymbol{u}, \tag{2}$$

where $\boldsymbol{u}(\boldsymbol{r}, t)$ denotes the fluid velocity (advective velocity). Equation (2) models advection as a simple co-movement of the dissolved ions with the surrounding fluid. All inertial effects of the ions are neglected.

The diffusive behavior of the ions is best described in a reference frame co-moving with the local advective velocity $\boldsymbol{u}$. We assume that the species' relative fluxes instantaneously relax to a local equilibrium. This assumption allows us to derive the diffusive fluxes from a local free-energy density, which we define as

$$f\big(c_k(\boldsymbol{r})\big) = \sum_k \underbrace{k_B T c_k(\boldsymbol{r}) \left[\log\left\{\Lambda_k^3 c_k(\boldsymbol{r})\right\} - 1\right]}_{\text{ideal gas contribution}} + \underbrace{z_k e c_k(\boldsymbol{r}) \Phi(\boldsymbol{r})}_{\text{electrostatic contribution}}, \tag{3}$$

with the Λ_k the species' thermal de Broglie wavelengths, z_k their valencies, and $\Phi(\boldsymbol{r})$ the electrostatic potential. This free-energy density consists of only an ideal-gas and an electrostatic contribution. The same assumptions form the basis of Poisson-Boltzmann (PB) theory. Hence, the limitations of this model are the same as those of PB. That means this model applies to monovalent ions at low to intermediate densities and surface charges [3, 4].

The species' chemical potentials μ_k implied by the free-energy density read

$$\mu_k(\boldsymbol{r}) = \delta_{c_k} f\big(c_k(\boldsymbol{r})\big) = k_B T \log(\Lambda_k^3 c_k(\boldsymbol{r})) + z_k e \Phi(\boldsymbol{r}). \tag{4}$$

This in turn allows us to formulate first-order approximation to the thermodynamic driving force as the gradient of the chemical potential (4), which we use to define an expression for the diffusive flux

$$\begin{aligned} \boldsymbol{j}_k^{\text{diff}} &= \xi_k \left(-c_k \nabla \mu_k\right) = -k_B T \xi_k \nabla c_k - \xi_k z_k e c_k \nabla \Phi \\ &= -D_k \nabla c_k - \xi_k z_k e c_k \nabla \Phi. \end{aligned} \tag{5}$$

Here, ξ_k and D_k denote the mobility and the diffusion coefficient of species k, which are related by the Einstein-Smoluchowski relation $D_k/\xi_k = k_B T$ [5, 6].

Finally, the total number density flux combining effects of diffusion and advection reads

$$\boldsymbol{j}_k = \boldsymbol{j}_k^{\text{diff}} + \boldsymbol{j}_k^{\text{adv.}} = -D_k \nabla c_k - \xi_k z_k e c_k \nabla \Phi + c_k \boldsymbol{u}, \tag{6}$$

where the first term represents Fick's law of diffusion in the absence of an external potential, the second term gives the additional flux due to an external (in this case electrostatic) potential, and the last term introduces the influence of the motion of the underlying solvent.

Electrostatics

dynamics of the charged species in a typical micro- or nanofluidic system are slow compared to the relaxation of the electromagnetic fields. This allows us to use stationary equations to model electromagnetic effects. We further assume that the modeled species do to carry permanent magnetic dipoles and that electric currents in the system are small. Under these conditions, the full set of Maxwell's equations reduces to the Poisson equation

$$\nabla^2 \Phi = -\frac{1}{\varepsilon} \sum_k z_k e c_k = -4\pi l_{\text{B}} k_B T \sum_k z_k c_k. \tag{7}$$

Here $\varepsilon = \varepsilon_0 \varepsilon_r$ denotes the product of the vacuum permittivity ε_0 with the relative permittivity of the solvent ε_r. We have also used the Bjerrum-length

$$l_{\text{B}} = \frac{e^2}{4\pi\varepsilon k_B T}. \tag{8}$$

Finally, we have assumed that the permittivity is spatially homogeneous, since this will allow us to use efficient spectral methods to solve this equation.

Hydrodynamics

As said before, since the ionic species' contribute at most a few percent to the overall mass, we can safely approximate the overall fluid's mass by the mass of the solvent (typically water) and model the solvents velocity field $\boldsymbol{u}(\boldsymbol{r}, t)$ using the Navier-Stokes equations for an isotropic, incompressible Newtonian fluid

$$\rho\big(\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u}\big) = -\nabla p_H + \eta \boldsymbol{\nabla^2 u} + \boldsymbol{f}, \tag{9}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{10}$$

where p_H denotes the hydrostatic pressure, η the shear viscosity, ρ the density of the fluid, and $\boldsymbol{f}$ an external body force density. For the assumption of incompressibility to hold, the Mach number needs to be small – a condition that is fulfilled for micro- and nanofluidic systems with flow velocities on the order of µm/s.

In section 2.1, we assumed that the ions' velocity relative to the fluid instantaneously relaxes to a stationary state and that this stationary velocity is given by the product of their mobility and

the force exerted on them. For this state to be stationary, all the momentum transferred into the ions by the external force needs to be dissipated into the fluid immediately. From this we can conclude that the force density acting on the fluid must read

$$\boldsymbol{f} = \sum_k \boldsymbol{j}_k^{\text{diff}}/\xi_k = -\sum_k (k_B T \nabla c_k + z_k e c_k \nabla \Phi). \tag{11}$$

Summarizing, the set of electrokinetic equations we solve is given by

$$\begin{aligned}
\boldsymbol{j}_k &= -D_k \nabla c_k - \xi_k z_k e c_k \nabla \Phi + c_k \boldsymbol{u}, && (12)\\
\partial_t c_k &= -\nabla \cdot \boldsymbol{j}_k, && (13)\\
\nabla^2 \Phi &= -4\pi l_{\mathrm{B}} k_B T \textstyle\sum_k z_k c_k, && (14)\\
\rho\big(\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\big) &= -\nabla p_H + \eta \boldsymbol{\nabla^2 u} - \textstyle\sum_k (k_B T \nabla c_k + z_k e c_k \nabla \Phi), && (15)\\
\nabla \cdot \boldsymbol{u} &= 0. && (16)
\end{aligned}$$

2.2 EOF in the Slit Pore Geometry

The slit pore system depicted in Fig. 1 consists of two like charged parallel plates of infinite extent, confining a solution of water and the plates' counterions.

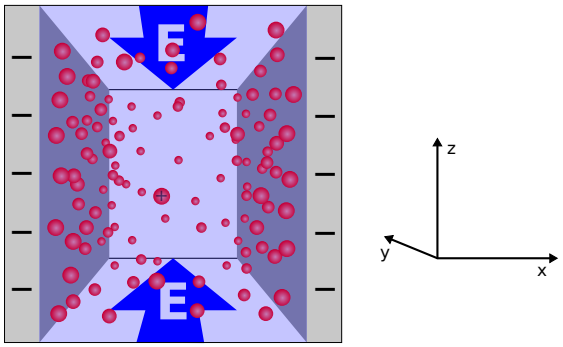


Figure 1: Slit pore system and coordinate system used for the analytical calculations.

Due to the net neutrality of the system and the translational symmetry in directions parallel to the plates, the potential outside the two plates must be constant. This means that using periodic or non-periodic boundary conditions makes no difference. As the system is in equilibrium in the normal direction, the electrokinetic eqs. (12) to (14) for this dimension reduce to the Poisson-Boltzmann equation for the electrostatic potential, which reads

$$\partial_x^2 \Phi(x) = -4\pi\, k_B T\, l_{\mathrm{B}}\, z e\, c_0 \cdot \exp\left(-\frac{z e \Phi(x)}{k_B T}\right), \tag{17}$$

where x denotes the direction normal to the plates. The constant c_0 has to be chosen such that charge neutrality is fulfilled. Multiplying by $2\partial_x \Phi(x)$ and applying the inverse chain rule further

reduces the equation to first order. Subsequent separation of variables yields the solution

$$\Phi(x) = -\frac{k_B T}{ze} \cdot \log\left[\frac{C^2}{8\pi\, k_B T\, l_\mathrm{B}} \cdot \cos^{-2}\left(\frac{zeC}{2k_B T} \cdot x\right)\right], \quad \left|\frac{zeC}{2k_B T} \cdot x\right| < \frac{\pi}{2}\,. \tag{18}$$

Refer to [1] for details on this calculation. Knowing that the counterion density c resembles a Boltzmann distribution in the potential $ze\Phi$ leads to the expression

$$c(x) = \frac{C^2}{8\pi\, k_B T\, l_\mathrm{B}} \cdot \cos^{-2}\left(\frac{zeC}{2k_B T} \cdot x\right)\,. \tag{19}$$

The constant C is determined by fixing the number of counterions or requiring the E-field to vanish outside the volume contained by the plates. Both yields

$$C \cdot \tan\left(\frac{zed}{4k_B T} \cdot C\right) = -4\pi\, k_B T\, l_\mathrm{B} \sigma\,, \tag{20}$$

where d denotes the distance between the plates and σ their (constant) surface charge density.

Applying an electrical field along one of the directions parallel to the plates does not influence the charge distribution in the normal direction, which allows us to write down the hydrodynamic equations for the parallel direction. After eliminating all terms from the the Navier-Stokes Eqations (15) which vanish due to symmetry, we are left with

$$\frac{\partial_x^2 v_y(x)}{\partial x^2} = -\frac{qEC^2}{8\, k_B T\, l_\mathrm{B}\, \eta} \cdot \cos^{-2}\left(\frac{qC}{2k_B T} \cdot x\right)\,, \tag{21}$$

which yields, by means of simple integration and the application of no-slip boundary conditions

$$v_y(x) = \frac{E}{2\pi\, l_\mathrm{B}\, \eta\, ze} \cdot \log\left[\frac{\cos\left(\frac{zeC}{2k_B T} \cdot x\right)}{\cos\left(\frac{zeC}{2k_B T} \cdot \frac{d}{2}\right)}\right]\,. \tag{22}$$

Here x stands for the direction normal to the plates and y for the direction of the E-field and parallel to the plates.

With this tutorial comes a Python script `eof_analytical.py`, which evaluates all these expressions on the same grid as is used in the simulation from section 3.

3 Simulation using ESPResSo

3.1 Setting up ESPResSo

Electrokinetics is a relatively new feature of **ESPResSo** and new functionality is still being added. This is why it is advisable to get a copy of the current **ESPResSo** master to work with. If you don't already have a working copy of the **ESPResSo** master, follow tutorial `00-building_espresso`. Enable the features `ELETROKINETICS` and `EK_BOUNDARIES` during the build process.

3.2 Mapping SI and Simulation Units

ESPResSo does not predefine any unit system. This makes it more flexible but also requires us to spend some thought on the conversion from SI units to simulation units and back. Since most first time users have trouble with this, we will go through that process in detail here.

Important to note is that **ESPResSo**'s unit system is nothing more than a rescaled variant of the SI system. All physical formulas you are used to in the SI system remain valid and you can use them to find relations between your units. Lets start by choosing a unit of length. Since we are going to deal with Debye layers with extensions of nanometers, a sensible choice is

$$[x] = 1\,\mathrm{nm}.$$

The involved energies are of the magnitude of k_BT. We will simulate our system at room temperature ($300\,\mathrm{K}$), hence we use as unit of energy

$$[E] = k_B \cdot 300\,\mathrm{K} \approx 4.14 \times 10^{-21}\,\mathrm{J}.$$

By default ESPResSo has no concept for particle masses (but the feature can be activated). That means all particle masses are assumed to be $1\,[\mathrm{m}]$, which forces us to use the particle mass as mass unit. For this simulation we use the mass of sodium ions, which is

$$[m] = 23\,\mathrm{u} \approx 3.82 \times 10^{-26}\,\mathrm{kg}.$$

For the relation

$$E = \frac{1}{2}mv^2$$

to hold, the unit of time has to be defined so that

$$[E] = [m] \cdot \frac{[x]^2}{[t]^2}.$$

From that we get the missing unit of time

$$[t] = [x] \cdot \sqrt{\frac{[m]}{[E]}} = 1\,\mathrm{nm} \cdot \sqrt{\frac{23\,\mathrm{u}}{k_B \cdot 300\,\mathrm{K}}} \approx 3.03760648 \times 10^{-12}\,\mathrm{s} \approx 3.04\,\mathrm{ps}.$$

The last unit we need is the one of charge. We choose it to be the elementary charge

$$[q] = e \approx 1.60 \times 10^{-19}\,\mathrm{C}.$$

We now have all the units necessary to convert out simulation parameters.

parameter	value (SI units)	value (simuation units)
channel width d	$50\,\mathrm{nm}$	$50\,[\mathrm{x}]$
thermal energy k_BT	$k_B \cdot 300\,\mathrm{K}$	$1\,[\mathrm{E}]$
Bjerrum length l_B	$0.7095\,\mathrm{nm}$	$0.7095\,[\mathrm{x}]$
counterion charge q	$1e$	$1\,[\mathrm{q}]$
counterion diffusion coefficient D	$2.0 \times 10^{-9}\,\mathrm{m}^2/\mathrm{s}$	$0.006075\,[\mathrm{x}]^2/[\mathrm{t}]$
solvent density ρ	$1.0 \times 10^3\,\mathrm{kg}/\mathrm{m}^3$	$26.18\,[\mathrm{m}]/[\mathrm{x}]^3$
kinematic solvent viscosity η	$1.0 \times 10^{-3}\,\mathrm{Pa\,s}$	$79.53\,[\mathrm{m}]/([\mathrm{x}][\mathrm{t}])$
external electrical field E	$2.585 \times 10^6\,\mathrm{V}/\mathrm{m}$	$0.1\,[\mathrm{E}]/[\mathrm{q}][\mathrm{x}]$

ESPResSo determines the strength of the electrostatic interactions via the Bjerrum-length l_B. That is the length for which the electrostatic interaction energy of two elementary charges equals the thermal energy

$$k_BT = \frac{e^2}{4\pi\varepsilon_0\varepsilon_r} \cdot \frac{1}{l_B}.$$

For water at $300\,\mathrm{K}$ with $\varepsilon_r = 78.54$ it is $l_B \approx 0.7095\,\mathrm{nm}$.

3.3 Setting up the slip pore system

The script for this simulation comes with this tutorial and is called `eof_electrokinetics.py`. All used commands are documented in the User's Guide in the section called "Electrokinetics".

We first set up a periodic simulation box of the desired dimensions. Note that the dimensions are of course given in simulation units.

```
1  # Initializing espresso modules and the numpy package
2  from espressomd import System, electrokinetics, shapes
3  import numpy as np
4
5  # Set the slit pore geometry the width is the non-periodic part of the geometry
6  # the padding is used to ensure that there is no field outside the slit
7
8  box_x = 6
9  box_y = 6
10 width = 50
11
```

```
12  padding = 1
13  box_z = width + 2*padding
14
15  system = System()
16  system.box_l = [box_x, box_y, box_z]
```

We then store all the parameters we calculated in section 3.2. At this point, these parameters only reside in Python variables. They will only be used by **ESPResSo** once they are being passed to the respective initialization functions.

```
19  # Set the electrokinetic parameters
20
21  agrid = 1.0
22  dt = 0.2
23  kT = 1.0
24  bjerrum_length = 0.7095
25  D = 0.006075
26  valency = 1.0
27  viscosity_dynamic = 79.53
28  density_water = 26.15
29  sigma = -0.05
30  ext_force = 0.1
```

Before we initialize the actual electrokinetics algorithm, we need to set the time step and some other parameters that are not actually used, but would otherwise lead to error messages.

```
32  # Set the simulation parameters
33
34  system.time_step = dt
35  system.cell_system.skin = 0.2
36  system.thermostat.turn_off()
37  integration_length = int(2e5)
```

We can now set up the electrokinetics algorithm. All functionality pertaining to this algorithm is available through the `electrokinetics` submodule of `espressomd`. Please note that the fluid viscosity is specified as a kinematic viscosity, which is the dynamic viscosity divided by the fluid density. The kinematic viscosity is also required if you initialize the pure lattice-Boltzmann method.

```
39  # Set up the (LB) electrokinetics fluid
40  viscosity_kinematic = viscosity_dynamic / density_water
41  ek = electrokinetics.Electrokinetics(agrid = agrid, ↩
```

```
        lb_density = density_water, viscosity = ↩
        viscosity_kinematic, friction = 1.0, T = kT, ↩
        bjerrum_length = bjerrum_length)
```

The value of the friction parameter in the previous setup command is irrelevant, since we don't include any explicit particles in our simulation, but it's needed to pass the sanity check of the LB.

Next, we set up the individual ionic species. In this case, we only set up one species of positively charged counterions. After setting up the species, we have to add it to the electrokinetics instance.

```
45  # Set up the charged and neutral species
46  density_counterions  = -2.0 * sigma / width
47  counterions = electrokinetics.Species(density=↩
        density_counterions, D=D, valency=valency, ext_force=[↩
        ext_force, 0, 0])
48
49  ek.add_species(counterions)
```

The `EKBoundary` command takes the keyword `charge_density` and the numerical charge density in simulation units as arguments. The `shape` keyword takes an instance of a shape, which is provided by the `shapes` submodule and is the same as for the `LBBoundary` command. Here we initialize two charged `Wall` boundaries. To initialize the boundaries, we have to add them to the `ekboundaries` instance of the system class. Finally, we initialize the electrokinetics algorithm with our setup by adding the electrokinetics instance as an actor to the system.

```
53  # Set up the walls confining the fluid
54  ek_wall_left = electrokinetics.EKBoundary(charge_density=↩
        sigma/agrid, shape=shapes.Wall(normal=[0, 0, 1], dist=↩
        padding))
55  ek_wall_right = electrokinetics.EKBoundary(charge_density↩
        =sigma/agrid, shape=shapes.Wall(normal=[0, 0, -1], ↩
        dist=-(padding+width)))
56
57  system.ekboundaries.add(ek_wall_left)
58  system.ekboundaries.add(ek_wall_right)
59
60  system.actors.add(ek)
```

After setting up the system, we integrate a sufficient number of time steps to relax the system into the stationary state and output the counterion density profile, the velocity profile, and the shear stress. Since this system has translational symmetry in the x- and y-direction, we iterate

over a line in the z direction and use the `species`

node

`.quantity` command, to output local quantities. You can instead also use the `electrokinetics.print_vtk_quantity` command to output the whole field at once in a Paraview compatible format.

Density and velocity are not the only fields available for output. Please refer to the User's Guide for all available options.

```
66  # Integrate the system
67  for i in range(100):
68      system.integrator.run(integration_length)
69      sys.stdout.write("\rintegration step: %03i"%i)
70      sys.stdout.flush()
71
72  print("Integration finished.")
73
74  # Output
75  position_list = []
76  density_list = []
77  velocity_list = []
78  pressure_xz_list = []
79
80  for i in range(int(box_z/agrid)):
81      if (i*agrid >= padding) and (i*agrid < box_z - ↩
           padding):
82          position = i*agrid - padding - width/2.0 + agrid↩
               /2.0
83          position_list.append(position)
84
85          # density
86          density_list.append(counterions[box_x/(2*agrid), ↩
               box_y/(2*agrid), i].density)
87
88          # velocity
89          velocity_list.append(ek[box_x/(2*agrid), box_y↩
               /(2*agrid), i].velocity[0])
90
91          # xz component pressure tensor
92          pressure_xz_list.append(ek[box_x/(2*agrid), box_y↩
               /(2*agrid), i].pressure[0,2])
93
94  np.savetxt("eof_electrokinetics.dat", np.column_stack((↩
       position_list, density_list, velocity_list, ↩
```

```
        pressure_xz_list)), header="#position ↩
        calculated_density calculated_velocity ↩
        calculated_pressure_xz")
```

With this tutorial also came a Python matplotlib script `plot.py`. If everything went well, running this script with Python from a folder containing the output files `eof_analytical.dat` and `eof_electrokinetics.dat` should produce the result shown in Figure 2.

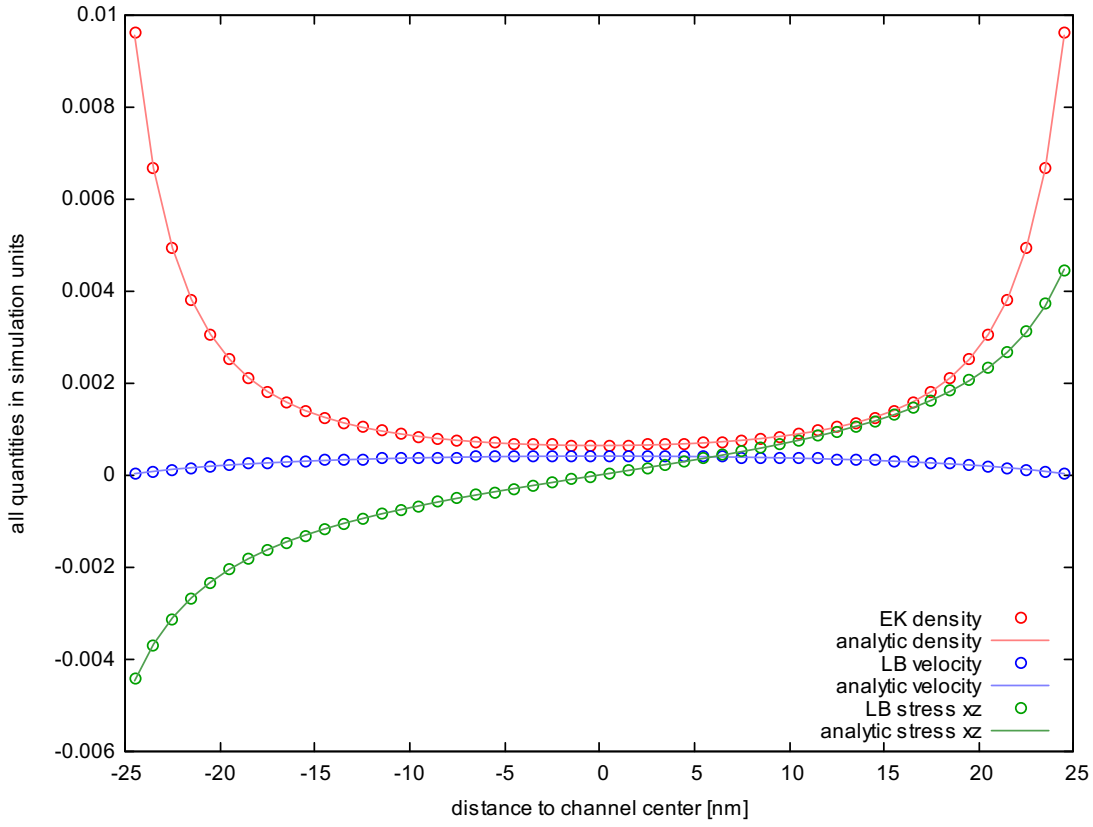


Figure 2: Profiles along the direction perpendicular to the slit pore walls for the counterion density, fluid velocity, and fluid shear stress. Parameters as chosen in Section 3.2.

References

[1] Georg Rempfer. Lattice-boltzmann simulations in complex geometries. Bachelor's thesis, Universtity of Stuttgart, Institute for Computational Physics, 2010.

[2] J. de Graaf, G. Rempfer, and C. Holm. Diffusiophoretic self-propulsion for partially catalytic spherical colloids. 14(3):272–288.

[3] Olli Punkkinen, Ali Naji, Rudolf Podgornik, Ilpo Vattulainen, and P-L Hansen. Ionic cloud distribution close to a charged surface in the presence of salt. *EPL (Europhysics Letters)*, 82(4):48001, 2008.

[4] Joost de Graaf, Niels Boon, Marjolein Dijkstra, and René van Roij. Electrostatic interactions between janus particles. *The Journal of chemical physics*, 137(10):104910, 2012.

[5] Albert Einstein. Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der physik*, 322(8):549–560, 1905.

[6] Marian Von Smoluchowski. Zur kinetischen theorie der brownschen molekularbewegung und der suspensionen. *Annalen der physik*, 326(14):756–780, 1906.