Ions in an AC Electric Field: Strong Long-Range Repulsion between Oppositely Charged Surfaces

Data S2

Paweł Jan Żuk

The Data S2 Content

The Data set S2 should contain

- `shortManual.pdf` - this file. The short manual on how to use the code;
- `electrokinFOAM` - the folder containing the OpenFOAM solver for solving the coupled Poisson-Nernst-Planck equations for two ions in solution;
- `electrokinBC` - the folder containing the no flux boundary conditions
- `oneDimCase` - the folder containing the case.

The solver and the boundary conditions were written, tested and used under `foam-ext-3.2` distribution of the OpenFOAM library.

Installation guide

To install the solver and the boundary conditions and to use them it is advised to have the `foam-ext-3.2` version of the OpenFOAM. The has not been tested under other OpenFOAM distributions. Next, it is sufficient to first enter the `electrokinFOAM` folder and execute

```
wmake
```

command as for every standard OpenFOAM utility. The place of the installation can be changed by manipulating the content of

```
electrokinFOAM/Make/files
```

file. Similarly, the installation of the boundary conditions is done. It is sufficient to enter the `electrokinBC` folder and execute the

```
wmake libso
```

command. To clean installation commands `wclean` in the `electrokinFOAM` folder and `wlcean libso` in the `electrokinBC` folder need to be executed.

electrokinFOAM

The solver `electrokinFOAM` is written to solve the Poisson-Nernst-Planck system of PDEs for the evolution of the ion concentrations n_σ ($\sigma = \pm$)

$$\frac{\partial n_\sigma}{\partial t} = \nabla \cdot D_\sigma \left(\nabla n_\sigma + \sigma \frac{e z_\sigma}{k_B T} n_\sigma \nabla \phi \right), \tag{1a}$$

under the influence of the electric field with potential ϕ

$$\Delta \phi = -\frac{e}{\epsilon} \left(z_+ n_+ - z_- n_- \right), \tag{1b}$$

with diffusion constants D_σ, ionic valency z_σ, T is the temperature, e is the elementary charge and k_B Boltzmann's constant. The electric field is described by Poisson's equation.

The solver works iteratively in the time loop. The time step can be either fixed or controlled by the analog of Courant number Co, that is calculated based on the flux of ions through the boundaries of the computational cells. We found that the optimal performance of the code is when the time step is piecewise constant. In the case of adjustable time step we chose to modify it as it takes the value outside the interval $\left(\hat{\mathrm{Co}}/2, \hat{\mathrm{Co}}\right)$, where $\hat{\mathrm{Co}}$ is the desired analog of Courant number. It can be specified as `maxCo` in the `contolDict` file.

In every time step the solver enters into the loop for solving the equations.

for i=1 to `nOuterCorrIons` **do**
  solve equation for the positive ion concentration n_+ evolution eq. (1a);
  solve equation for the negative ion concentration n_- evolution eq. (1a);
  for j=1 to `nCorrIons` **do**
    solve Poisson's equation eq. (1b);
  end for
end for

Two control values `nOuterCorrIons` and `nCorrIons` are specified as

```
PISO
{
    nOuterCorrIons   <integer>;
    nCorrIons        <integer>;
}
```

subdictionary in the `fvSolution` dictionary. For `nOuterCorrIons`= 1 the solver behaves in similar manner to the PISO algorithm for iterative solution of pressure-velocity coupling. In case `nOuterCorrIons`> 1 the solver becomes similar to PIMPLE algorithm.

The solver needs an additional dictionary `physicalProperties` in the `constant` folder specified. This dictionary contains physical constants necessary for the solution

```
epsilon0   epsilon0 [ -1 -3 4 0 0 2 0 ] <dielectric constant>;
e          e [ 0 0 1 0 0 1 0]           <elementary charge>;
nRef       nRef [ 0 0 0 0 0 0 0]        <reference concentration number n0>;
muPlus     muPlus [ -1 0 2 0 0 1 0]     <mobility coefficient of plus ion>;
muMinus    muMinus [ -1 0 2 0 0 1 0]    <mobility coefficient of plus ion>;
DPlus      DPlus [ 0 2 -1 0 0 0 0]      <diffusion coefficient of the plus ion>;
DMinus     DMinus [ 0 2 -1 0 0 0 0]     <diffusion coefficient of the minus ion>;
ZPlus      ZPlus [ 0 0 0 0 0 0 0 ]      <unsigned valency of the plus ion>;
ZMinus     ZMinus [ 0 0 0 0 0 0 0 ]     <unsigned valency of the minus ion>;
nMinimal   nMinimal [ 0 -3 0 0 0 0 0 ]  <lower boundary for the density fields>;
```

The `nRef` stands for the reference number that unit concentration has to be multiplied in order to get a real concentration value. For example in case of desired 1 mM concentration `nRef` should equal to 6.022×10^{23}. The relation between the mobility μ and the diffusion coefficient D is given by

$$\mu_{\pm} = \frac{z_{\pm} e D}{k_B T}. \tag{2}$$

The `nMinimal` is used as the lower boundary of the concentration for the calculation of the analog of Co number.

All other parameters are specified in the same way as in other OpenFOAM solvers.

electrokinBC

The no flux boundary conditions

$$\mathbf{j} = -D_{\sigma} \left(\nabla n_{\sigma} + \sigma \frac{z_{\sigma} e}{k_B T} n_{\sigma} \nabla \phi \right) = 0. \tag{3}$$

are specified using the `electrokinBC` library. After successful compilation one has to add the

```
libs ( "electrokinBC.so" );
```

line at the end of the `controlDict` dictionary. After specifying the name of the additional library, the boundary condition can be used in the following form

```
{
    type            fixedFlux;
    n               <name of the corresponding ion concentration>;
    sign            <sign of the ion charge>;
    D               <diffusion coefficient>;
    mu              <mobility coefficient>;
    gradient        uniform 0.0;
    value           calculated;
}
```

Please use the values that correspond to the values specified in the `physicalProperties` dictionary. Otherwise the results will be wrong.

oneDimCase

The `oneDimCase` is a complete example with already meshed geometry. It is sufficient to enter the folder and run

```
electrokinFOAM
```

The geometry is a non uniformly meshed realization of 1D segment. The mesh gets denser near the `plateOne` wall where no flux boundary conditions are applied. The total length of the segment is 0.5 μm. At the `plateOne` boundary we apply the no flux boundary conditions for the ions and the oscillatory electric potential. At the `plateTwo` boundary we apply the bulk concentration boundary condition for the ions and constant zero potential. The oscillatory boundary condition for the potential is defined in the `plateOne.dat` source file for the `timeVaryingUniformFixedValue` boundary conditions.

The system is filled with the 1 mM solution of $NaNO_3$ salt. Specified are the physical values of valencies, diffusion coefficients and mobility coefficients. The concentration `nRef` is equal to 6.022×10^{23}. The initial concentration fields are specified in initial condition files for the ion concentrations `0/nPlus` and `0/nMinus` to be equal 1. In the case of the divalent electrolyte like H_2SO_4 this should be in the desired ratio e.g., 1 for `nPlus` and 0.5 for `nMinus`. During the calculations the ionic concentration fields are multiplied by the `nRef`.

In addition to the files needed for the solution we also provide a exemplary `sampleDict` file for probing the fields along the line.