A thermalized electrokinetics model including stochastic reactions suitable for multiscale simulations of reaction-advection-diffusion systems

Ingo Tischler, Florian Weik, Robert Kaufmann,
Michael Kuron, Rudolf Weeber, and Christian Holm
Institute of Computational Physics, University of Stuttgart, Germany

(Dated: December 6, 2021)

Abstract

We introduce a scheme to simulate the spatial and temporal evolution of the densities of charged species, taking into account diffusion, thermal fluctuations, coupling to a carrier fluid, and chemical reactions. To this end, the diffusive fluxes in the electrokinetic model by Capuani *et al.* [1] are supplemented with thermal fluctuations. Chemical reactions are included via an additional source term in the mass balance equation. The diffusion-reaction model is then coupled to a solver for fluctuating hydrodynamics based on the lattice Boltzmann method. This combination is particularly useful for soft matter simulations, due to the ability to couple particles to the lattice-Boltzmann fluid. These could, e.g., be charged colloids or polymers, which then interact with an ion distribution. We describe one implementations based on the automatic code generation tools pystencils and lbmpy, and another one that is contained in the molecular dynamics package ESPResSo and that allows for an easy coupling of particles to the density fields. We validate our implementations by comparing to several known analytic results. Our method can be applied to coarse-grained catalysis problems as well as to many other multi-scale problems that require the coupling of explicit-particle simulations to flow fields, diffusion, and reaction problems in arbitrary geometries.

I. INTRODUCTION AND HISTORY

The lattice electrokinetics model by Capuani *et al.* [1] is an advection-diffusion model for multiple species that interact electrostatically. Hydrodynamic effects are taken into account by means of the lattice Boltzmann method [2–4]. The model is applicable to those cases where effects on the level of the individual ion can be neglected and the system can be described by continuous density fields. Since the lattice Boltzmann method can easily handle any geometry that can be grid resolved, suitable applications are found in research fields ranging from the transport of electrolytes through porous media to biomolecules and colloids in solutions containing ions [5–7], as well as the investigation of microfluidic mechanisms like pumps [8] or selective particle traps [9, 10]. An extension of the electrokinetic model to moving colloids in binary fluid flows has also been achieved recently [11].

The electrokinetic model can be extended to include chemical reactions by adding source terms to the mass balance equation [12]. These lead to the conversion of different species into each other as governed by the reaction rates. In this way, the model can be used to analyze diffusion or flow-driven reactions [13] and to simulate active particles [14].

The standard electrokinetic model does not include thermal fluctuations. While this is not an issue for many applications that treat systems on larger scales, e.g. in the field of microfluidics, in some cases the fluctuations will play an important role. These include fluctuation-induced instabilities [15], pattern formation [16, 17], stochastic processes in cell biology [18], transport properties [19, 20], and many processes that happen on the nanoscale where the relative significance of fluctuations is large. For chemical reactions, thermal fluctuations can have a significant impact on the reaction rate if the order of the reaction is not 1: the presence of thermal fluctuations will accelerate or slow down these reactions [21]. Lastly, fluctuations in the electrokinetic model have to be taken into account when coupling it to other thermalized models, such as a fluctuating lattice Boltzmann method [22, 23] or a particle-based simulation in which the particles undergo Brownian motion. If the electrokinetic solver would not include thermal fluctuations, it would act as an energy sink, continually drawing energy from the other coupled thermalized models.

To overcome these limitations, Dean [24] added a stochastic fluctuation term to the flux in the electrokinetic equations, which can be used in combination with different models like charged dissipative particle dynamic [25], electrolyte solutions [26, 27] or the fluctuating immersed boundary method [28]. In the present article, we combine the above methods to arrive at a coupled solver for fluctuating electrokinetics, fluctuating hydrodynamics, and chemical reactions. Donev *et al.* have developed a similar method based on fluctuating hydrodynamics (FHD) [12, 26, 27]. They couple the diffusive species to an incompressible Navier-Stokes fluid, which they solve via a finite-volume method. We want to expand on this previous work by solving the advection using a lattice Boltzmann method. This method has the advantage that coupling colloidal particles as well as hard and soft moving boundaries are supported. We discuss two implementations of the method. One is based on an automatic code generation approach using pystencils [29] and lbmpy [30] that allows for a fast prototyping of these models. The other one is part of the molecular dynamics (MD) software package ESPResSo [31], which allows an easy coupling to explicit particle MD simulations. It will also serve as the basis for future work on coupling colloidal particles subject to Brownian motion.

The paper is structured as follows: in section II we give a detailed description of the model. In section III we discuss the discretization scheme. In section IV we go into detail concerning the implementations. Section V describes how our implementations are validated. We conclude with a summary and outlook in section VI.

II. THE ELECTROKINETIC MODEL

In this section, we will first present a summary of the electrokinetic model and then describe how one can introduce thermal fluctuations. The electrokinetic model describes the diffusion and advection behaviour of N species, in the following indexed by k. The density of each species as a function of position and time is denoted by $n_k(\vec{r}, t)$. The corresponding density fluxes are $\vec{j}_k(\vec{r}, t)$.

A. Athermal model

The flux has two contributions, diffusion and advection:

$$\vec{j}_k = \vec{j}_k^{\text{diff}} + \vec{j}_k^{\text{adv}}. \tag{1}$$

We require mass conservation

$$\frac{\partial n_k}{\partial t} = -\nabla \cdot \vec{j}_k. \tag{2}$$

To obtain the diffusive contribution, we write down the free energy as a functional of density

$$F\left[n_k(\vec{r})\right] = \sum_k \underbrace{k_\mathrm{B}T n_k(\vec{r})\left(\log\left(\Lambda_k^3 n_k(\vec{r})\right) - 1\right)}_{\text{ideal gas contribution}} + \underbrace{q_k n_k(\vec{r})\Phi(\vec{r})}_{\text{electrostatics}} \tag{3}$$

with the Boltzmann constant k_B, temperature T, de Broglie wavelength Λ, charge of a particle q_k and the electrostatic potential Φ. The chemical potential μ_k is obtaining by taking the functional derivative of the free energy with respect to particle number:

$$\mu_k(\vec{r}) = \frac{\delta F\left[n_k\right]}{\delta n_k}. \tag{4}$$

This allows us to write the diffusive flux as

$$j_k^{\mathrm{diff}} = -\nu_k n_k \nabla \mu_k, \tag{5}$$

where ν_k is the mobility of species k. We thus have

$$\begin{aligned} \vec{j}_k^{\mathrm{diff}} &= -k_\mathrm{B}T\nu_k \nabla n_k - \nu_k q_k n_k \nabla\Phi \\ &= -D_k \nabla n_k - \nu_k q_k n_k \nabla\Phi. \end{aligned} \tag{6}$$

$D_k = \nu_k k_\mathrm{B} T$ is the diffusion coefficient fulfilling the Einstein-Smoluchowski relation. In summary, the diffusive flux consists of Fickian diffusion and an electrostatic driving force.

Let us now consider the advective flux. It is due to the motion of the solvent fluid. At the same time, forces on the immersed species lead to a driving force on the fluid. This can be captured via the Navier-Stokes equations

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\vec{\nabla}\vec{u}\right) = -\vec{\nabla}p + \eta\vec{\nabla}^2\vec{u} + \vec{f}, \tag{7}$$

$$\frac{\partial \rho}{\partial t} + \vec{\nabla}\cdot\rho\vec{u} = 0. \tag{8}$$

Here, η is viscosity, $\vec{u}$ flow velocity, p the pressure, and $\vec{f}$ a driving force density. We assume that any force acting on the immersed species is transferred instantly onto the fluid, hence

$$\vec{f} = -\sum_k \left(k_\mathrm{B}T\nabla n_k + q_k n_k \nabla\Phi\right) = \sum_k \frac{1}{\nu_k}\vec{j}_k^{\mathrm{diff}}. \tag{9}$$

Similarly, we assume that the immersed species move with exactly the same velocity as the underlying fluid, neglecting any inertial effects. Hence, the advective flux is

$$\vec{j}_{\mathrm{adv}} = n_k\vec{u}. \tag{10}$$

The electrostatic potential is obtained by solving the Poisson equation

$$\Delta\Phi = \frac{1}{\epsilon}\sum_k q_k n_k = -4\pi l_{\mathrm{B}} k_{\mathrm{B}} T \sum_k q_k n_k, \tag{11}$$

where $l_{\mathrm{B}} = \frac{e^2}{4\pi\epsilon k_{\mathrm{B}} T}$ is the Bjerrum length with ϵ being the dielectric permittivity of the solvent.

Chemical reactions can be incorporated into eq. (2) by adding a source term

$$R_k = s_k \gamma = s_k K \prod_i n_i^{\alpha_i}, \tag{12}$$

to the right-hand side. It contains the reaction rate γ, which is expressed in terms of the reaction constant K and the stoichiometric coefficients $\vec{s} = \{-a, -b, \ldots, c, d, \ldots\}$. The latter come from the chemical reaction equation $a\mathrm{A} + b\mathrm{B} + \cdots \leftrightharpoons c\mathrm{C} + d\mathrm{D} + \ldots$. The α_i define the reaction order, stating how much the reaction rate depends on the concentration of the different species, and for most chemical reactions the order of the educt species is $i \in \{\text{educts}\}, \alpha_i = s_i$, while the reaction does not depend on the product concentration and hence $i \in \{\text{products}\}, \alpha_i = 0$.

B. Thermal fluctuations

Let us now discuss the introduction of thermal fluctuations. For this purpose, we follow the derivation in ref. [24] and begin with the over-damped Langevin equation for N particles experiencing a single-particle potential $\psi(r)$,

$$\dot{\vec{r}}_i(t) = \vec{\eta}_i(t) - \vec{\nabla}_i \psi(\vec{r}_i, t), \tag{13}$$

where $\eta_i(t)$ is an uncorrelated Gaussian random force obeying

$$\langle \eta_i^\mu(t) \eta_j^\nu(t') \rangle = 2D\delta_{ij}\delta_{\mu\nu}\delta(t-t'), \quad \langle \vec{\eta}_i(t) \rangle = 0, \tag{14}$$

with the superscripts indicating Cartesian components.

Following ref. 24 we will derive an equation for the global density

$$n(\vec{r}, t) = \sum_{i=1}^{N} n_i(\vec{r}, t) = \sum_{i=1}^{N} \delta(\vec{r}_i(t) - \vec{r}), \tag{15}$$

where $n_i(\vec{r}, t) = \delta(\vec{r}_i(t) - \vec{r})$ is the density function of a single particle i.

With f being an arbitrary function, the definition of the single particle density $n_i(\vec{r},t)$ allows us to define

$$f(\vec{r}_i(t)) = \int_{\mathbb{R}^3} \mathrm{d}^3 r\, n_i(\vec{r},t) f(\vec{r}). \tag{16}$$

We can now analyze the difference of the classical and the stochastic time derivative which will lead us to a continuous time derivative of n. The classical derivative of f is

$$\frac{\mathrm{d}}{\mathrm{d}t} f(\vec{r}_i) = \int_{\mathbb{R}^3} \mathrm{d}^3 r\, \frac{\partial n_i(\vec{r},t)}{\partial t} f(\vec{r}). \tag{17}$$

For the stochastic Itô derivative [32] of f, we expand equation 16 over the next time step δt. To do this we utilize Itô's lemma for a function $f(B_t,t)$ with a Brownian process B_t:

$$\mathrm{d}f = \left(\frac{\partial f}{\partial t} + \vec{\mu}_t \frac{\partial f}{\partial x} + \frac{\sigma_t^2}{2} \frac{\partial^2 f}{\partial x^2} \right) \mathrm{d}t + \sigma_t \frac{\partial f}{\partial x} \mathrm{d}B_t. \tag{18}$$

Here, $\vec{\mu}_t$ is the drift velocity and σ_t is the variance of the Brownian process. The time derivative of the Brownian process is given by $\dot{\vec{B}}_t = \vec{\eta}(t)/\sqrt{2\gamma k_\mathrm{B} T}$. With this we obtain

$$\begin{aligned} \frac{\mathrm{d}f(\vec{r}_i)}{\mathrm{d}t} &= \underbrace{\frac{\partial f}{\partial t}}_{=0} + \vec{\nabla}_i f(r_i) \left(\vec{\mu}_t + \sigma_t \frac{\mathrm{d}\vec{B}_t}{\mathrm{d}t} \right) + \frac{\sigma_t^2}{2} \nabla_i^2 f(r_i) & (19) \\ &= \int \mathrm{d}^3 r f(\vec{r}) \Big(\vec{\nabla}_i \cdot (n_i(\vec{r},t) \vec{\eta}_i(t)) \\ &\quad - \vec{\nabla}_i (n_i(\vec{r},t)) \cdot \vec{\nabla}_i \psi(\vec{r}_i,t) + D \nabla_i^2 n_i(\vec{r},t) \Big) . & (20) \end{aligned}$$

Subtracting the classical (equation 17) from the stochastic derivative (equation 20) and summing over all particles i, we arrive at

$$\frac{\partial n(\vec{r},t)}{\partial t} = \xi(\vec{r},t) - \vec{\nabla} \cdot (n(\vec{r},t) \nabla \psi(\vec{r},t)) + D \nabla^2 n(\vec{r},t), \tag{21}$$

where the noise term $\xi(\vec{r},t)$ is redefined as

$$\xi(\vec{r},t) = \sum_{i=1}^{N} \vec{\nabla} \cdot (n_i(\vec{r},t) \eta_i(t)) \tag{22}$$

to contain $n_i(\vec{r},t)$. This redefined noise term has the correlation function

$$\langle \xi(\vec{r},t) \xi(\vec{r}',t') \rangle = 2D\delta(t-t') \sum_{i=1}^{N} \vec{\nabla}_r \cdot \vec{\nabla}_{r'} (n_i(\vec{r},t) n_i(\vec{r}',t)). \tag{23}$$

The correlation function can be rewritten using the properties of the Dirac delta function yielding

$$\langle \xi(\vec{r},t)\xi(\vec{r}',t')\rangle = 2D\delta(t-t')\vec{\nabla}_r \cdot \vec{\nabla}_{r'}(\delta(\vec{r}-\vec{r}')n(\vec{r},t)). \tag{24}$$

We want to express the noise term 22 without the usage of a summation. To that end we define a different noise field, which is statistically identical. This is achieved by having the same correlation function. The new global noise field with that exact property turns out to be

$$\xi'(\vec{r},t) = \vec{\nabla} \cdot (\vec{\eta}(\vec{r},t)\sqrt{n(\vec{r},t)}) \tag{25}$$

with the global white noise field $\vec{\eta}$ characterized by

$$\langle \eta^{\mu}(\vec{r},t)\eta^{\nu}(\vec{r}',t')\rangle = 2D\delta^{\mu\nu}\delta(t-t')\delta(\vec{r}-\vec{r}'), \quad \langle \vec{\eta}(\vec{r},t)\rangle = 0. \tag{26}$$

With $\xi'(\vec{r},t)$ we can rewrite equation 21 as

$$\frac{\partial n(\vec{r},t)}{\partial t} = \vec{\nabla} \cdot \left(n(\vec{r},t)\nabla\psi(\vec{r},t) + D\nabla n(\vec{r},t) + \sqrt{n(\vec{r},t)}\vec{\eta}(\vec{r},t) \right). \tag{27}$$

This is a more general version of equation 6 with an additional noise term. While this describes the influence thermal fluctuations towards diffusion we can also add thermal fluctuations to the chemical reactions terms, as was originally done by Gillespie [33]. For this derivation we assume that chemical reactions are Poisson processes. This holds true under the condition that the substrate density changes happening during a reaction in the time interval $[t, t+\Delta t]$ are small enough so that the influence of this density change on the probability of the single reactions is negligible. This renders the different reaction processes to be independent from each other. This condition can be maintained by choosing a sufficiently small time step Δt. With that we can write the chemical equation as:

$$\gamma = K\prod_i n_i^{\alpha_i} \tag{28}$$

$$n_k(t+\Delta t) = n_k + s_k\mathcal{P}(\gamma(\vec{r},t),\Delta t) \tag{29}$$

with $\mathcal{P}(\gamma(\vec{r_i},t),\Delta t)$ being a Poisson random variable denoting the number of reactions that occur within the time-span Δt. Under the assumption that there are multiple reactions happen per time step $\mathcal{P}(\gamma(\vec{r_i},t),\Delta t) \gg 1$ this Poisson random variable converges to a normal distributed random variable $\mathcal{N}(\gamma(\vec{r},t)\Delta t, \gamma(\vec{r},t)\Delta t)$, where $\mathcal{N}(m,\sigma^2)$ is a random normal distribution with mean m and variance σ^2. This assumption basically adds an upper and a lower limit for the values that the time step can assume. Furthermore, this normal distribution can also be expressed via a Gaussian $\mathcal{N}(0,1)$ with mean $m = 0$ and variance $\sigma^2 = 1$ that simplifies the expression to:

$$n_k(t+\Delta t) = n_k + s_k\gamma(\vec{r},t)\Delta t + s_k\sqrt{\gamma(\vec{r},t)\Delta t}\mathcal{N}(0,1). \tag{30}$$

III. DISCRETIZATION

Let us now discretize the fluctuating electrokinetic equations such that they may be be solved numerically on a lattice. We will be using the same finite volume discretization as Rempfer *et al.* [9], who improved the model of Capuani *et al.* [1] to reduce numerically caused flux artifacts. Recall that the continuous electrokinetic equations are

$$\begin{aligned} \frac{\partial n_k}{\partial t} &= -\nabla\cdot\vec{j}_k, \\ \vec{j}_k &= \underbrace{-D_k\nabla n_k}_{\vec{j}^{\text{diff}}}\underbrace{-\nu_k q_k n_k\nabla\Phi}_{\vec{j}^{\text{pot}}}\underbrace{-\sqrt{n_k}\vec{\eta}_k}_{\vec{j}^{\text{fluc}}}\underbrace{+n_k\vec{v}}_{\vec{j}^{\text{adv}}}. \end{aligned} \tag{31}$$

We will be using two regular cubic grids, one for the densities and a second, staggered one, for the fluxes. The regular grid will store the mean density of the region in space that it covers. The flux is defined on the staggered grid, which is shifted by $\frac{a_i}{2}$ in all directions compared to the density grid. With that the density exchange of two neighbouring cells is characterized by the flux located in between. Here, a_i is the lattice constant in the direction i, which is defined by the grid stencil. There is some freedom in choosing the discretization scheme. Our choice was to be consistent with the thermal fluxes. Approximating $\vec{\nabla}\cdot\vec{j}_k$ with finite differences allows us to write

$$\vec{\nabla}\cdot\vec{j}_k(\vec{x},t) \approx \sum_{i=1}^{q}\frac{1}{a_i A_0}(j_{k,i}(\vec{x}+\frac{\vec{a_i}}{2},t) - j_{k,i}(\vec{x}-\frac{\vec{a_i}}{2},t)), \tag{32}$$

where $\vec{a_i}$ defines the vector to the nearest grid neighbours and $j_{k,i}$ represents the flux to the given grid direction $\vec{a_i}$. A_0 is a normalization factor of the used grid stencil. This is needed for grids, where the fluxes are not linearly independent. For D2Q9, we get $A_{0,D2Q9} = 1+\sqrt{2}$. For D3Q19 and D3Q27, we have $A_{0,D3Q19} = 1+2\sqrt{2}$ and $A_{0,D3Q27} = 1+2\sqrt{2}+\frac{4}{3}\sqrt{3}$, respectively. We now look at the different contributions to the flux, starting with the diffusive flux:

$$\vec{j}^{\text{diff}}_{k,i}(\vec{x}+\frac{\vec{a_i}}{2},t) = -D\vec{\nabla}n_k(\vec{x}+\frac{\vec{a_i}}{2},t). \tag{33}$$

We use finite differences to approximate $\vec{\nabla} n_k(\vec{x} + \frac{\vec{a_i}}{2}, t)$, which yields

$$\vec{\nabla} n_k(\vec{x} + \frac{\vec{a_i}}{2}, t) \approx \frac{\vec{a_i}}{a_i^2}(n_k(\vec{x} + \vec{a_i}, t) - n_k(\vec{x}, t)). \tag{34}$$

The contribution of the electrostatic potential is treated in a similar fashion, yielding

$$\vec{j}_{k,i}^{\,\text{pot}}(\vec{x} + \frac{\vec{a_i}}{2}, t) = \nu_k q_k n_k(\vec{x} + \frac{\vec{a_i}}{2}, t)\nabla\Phi(\vec{x} + \frac{\vec{a_i}}{2}, t). \tag{35}$$

To approximate the density between the grid nodes, we use the arithmetic mean of the two adjacent nodes in the direction $\vec{a_i}$, which then yields

$$\vec{j}_{k,i}^{\,\text{pot}}(\vec{x} + \frac{\vec{a_i}}{2}, t) \approx -\frac{\vec{a_i}}{a_i^2}\nu_k q_k \left(\frac{n_k(\vec{x} + \vec{a_i}, t) + n_k(\vec{x}, t)}{2}(\Phi(\vec{x} + \vec{a_i}, t) - \Phi(\vec{x}, t))\right). \tag{36}$$

Calculating the advective flux, we rely on the assumption that the density of particles in the small volume represented by a node is homogeneous. The advective velocity displaces this volume and transfers it to neighbouring cells. Figure 1 illustrates this scheme in two dimensions.

Lastly the distribution due to the fluctuation are described as

$$\begin{aligned}\vec{j}_{k,i}^{\,\text{fluc}}(\vec{x} + \frac{\vec{a_i}}{2}, t) &= \sqrt{n_k(\vec{x} + \frac{\vec{a_i}}{2}, t)}\eta_k(\vec{x} + \frac{\vec{a_i}}{2}, t) \\ &\approx \sqrt{\frac{n_k(\vec{x} + \vec{a_i}, t) + n(\vec{x}, t)}{2}}\,\eta_k(\vec{x} + \frac{\vec{a_i}}{2}, t).\end{aligned} \tag{37}$$

Since we know the statistical properties of the noise field $\eta_k(\vec{x} + \frac{\vec{a_i}}{2}, t)$ from equation 26, we can express it using a Gaußian white noise $\vec{\mathcal{W}}$ as

$$\vec{\eta}_k(\vec{x}, t) = \sqrt{2D}\vec{\mathcal{W}}(t). \tag{38}$$

Using an explicit first-order time stepping scheme and considering that the time integral over the white noise is $\int_t^{t+\Delta t} \vec{\mathcal{W}}(t')dt' = \sqrt{\Delta t}\vec{\mathcal{W}}(t)$ yields

$$n_k(\vec{x}, t + \Delta t) = n_k(\vec{x}, t) - \Delta t \vec{\nabla} \cdot (\vec{j}_k^{\,\text{pot}}(\vec{x}, t) + \vec{j}_k^{\,\text{diff}}(\vec{x}, t) + \frac{1}{\sqrt{\Delta t}}\vec{j}_k^{\,\text{fluc}}(\vec{x}, t)). \tag{39}$$

Applying the discretization of the flux gradient (equation 32) to the fluctuation flux we need to normalize the variance of the flux and not the flux itself. This, however, leads to a length scaling factor a_i and a stencil factor A_0 to appear under the square root

$$\vec{\nabla} \cdot \vec{j}_k^{\,\text{fluc}}(\vec{x}, t) \approx \sum_{i=1}^{q} \frac{1}{\sqrt{a_i A_0}}(j_{k,i}^{\text{fluc}}(\vec{x} + \frac{\vec{a_i}}{2}, t) - j_{k,i}^{\text{fluc}}(\vec{x} - \frac{\vec{a_i}}{2}, t)). \tag{40}$$

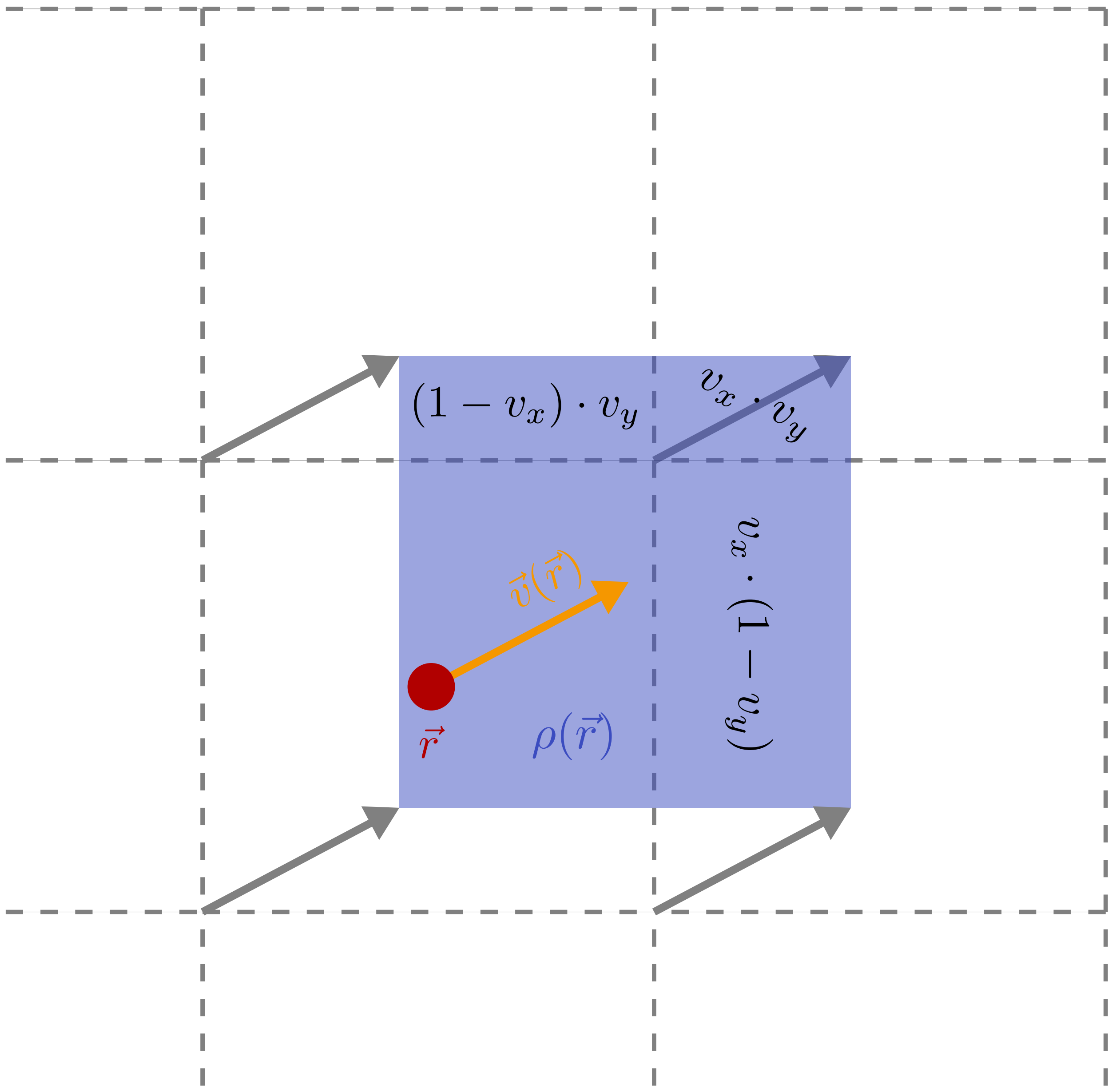

FIG. 1: Scheme for the advection on a two dimensional grid with spacing a. The dashed lines indicate the boundaries between cells with the lattice nodes in their centre. The flux to the right neighbour would be given by $n(\vec{r})v_x(1-v_y)\frac{a^2}{\Delta t}$. Image taken from ref. 34.

Since the fluctuating reactions equation (30) are already in a discretized form, no additions need to be made there.

IV. IMPLEMENTATION

In practice, the model is implemented by coupling three solvers: a finite-volume solver for the diffusion-advection equation [1, 35], a Fourier-based solver for obtaining the electrostatic potential from the charge distribution via the Poisson equation [35], and a lattice Boltzmann hydrodynamic implementation. The simulation package ESPResSo [31, 36] contains all necessary components. As they are largely based on regular grids and matrix-vector operations, all algorithms are well suited for acceleration on graphics processors (GPUs). ESPResSo hence employs such accelerated versions [35, 37].

For instructive purposes, we also provide an implementation written in Python for the advection-diffusion and for the Poisson solver in this contribution. They can be used to validate the model, but can also easily be extended for other purposes. We use the pystencils package as backend for the finite volume diffusion advection solver, and NumPy/SciPy for a Fourier-based Poisson solver. The Fourier-approach is highly efficient and requires very little custom code, but in its simple form only works in conjunction with periodic boundary conditions. While the discretization of section III has been performed manually in the ESPResSo implementation, pystencils uses sympy to define stencil operations and generates the discretization scheme automatically. During this code generation process one can specify the target platform, i.e. whether the code shall be executed in the CPU, GPU or via OpenMP. It is also possible to change the floating point precision, which one can use to optimize the balance between performance and numerical accuracy.

The full algorithm, including electrokinetics, reactions and lattice Boltzmann involves the following steps

1. Calculate total charge from per-species densities and valencies
2. Obtain the electrostatic potential for the charge distribution by solving the Poisson equation via the Fourier solver
3. Calculate the fluxes, including thermal fluctuations, using Eq. 31
4. Based on the fluxes, add external forces to the coupled lattice Boltzmann fluid (Eq. 9)
5. Add sources and sinks to the fluxes based on chemical reactions (Eq. 12)
6. Apply any boundary conditions to the densities and fluxes

7. Solve the continuity equation (Eq. 2)
8. Propagate the lattice Boltzmann fluid by one step
9. Advect the densities according to the lattice Boltzmann fluid's velocities. This is done using a volume-of-fluid approach, more formally known as *corner-transport upwind scheme* [1, 38].

While the underlying equations are straight forward, there are some details that need to be discussed. For the derivation of the advection-diffusion method, it is assumed that the particle density per cell is large enough to be considered as a continuum. However, for some examples, like the ion distributions in nanoporous media, one would need a very fine grid resolution to resolve the Debye layer and to render the geometry. This in turn would result in a small average density per gird cell ($\approx 10 - 20/a^3$), and it it is possible that locally some cells may reach a density below $n_k(\vec{r_i}) < 1/a^3$. If one would then calculate the amplitude of the fluctuations as stated in equation 40, the resulting fluctuation could be larger than the density in the cell, which would mathematically lead to a negative density. This is unphysical and also leads to a numerical breakdown in the next time step. To circumvent this issue, we scale down the amplitude of the fluctuations smoothly to be less than the local density. For that we define a smoothed heaviside function:

$$H(n) = \begin{cases} 0 & n < 0 \\ n & 0 < n < 1 \\ 1 & 1 < n \end{cases} , \tag{41}$$

which we use to rescale the average density of equation 37:

$$\vec{j}^{\text{fluc}}_{k,i}(\vec{x} + \frac{\vec{a_i}}{2}, t) = \sqrt{\frac{n_k(\vec{x} + \vec{a_i}, t) + n(\vec{x}, t)}{2}} H(n_k(\vec{x} + \vec{a_i}, t)) H(n(\vec{x}, t)) \eta_k(\vec{x} + \frac{\vec{a_i}}{2}, t). \tag{42}$$

The same is applied to the fluctuating reactions 30.

To generate the random noise, we choose the Philox pseudo-random number generator, which has good performance on CPU and GPU [39]. The Philox generator is counter-based, hence it does not have an internal state. Instead, pseudo-random numbers are generated from a few deterministic numbers such as the time step counter and the coordinates of the lattice cell for which a flux is to be calculated. It is of importance that the random numbers used

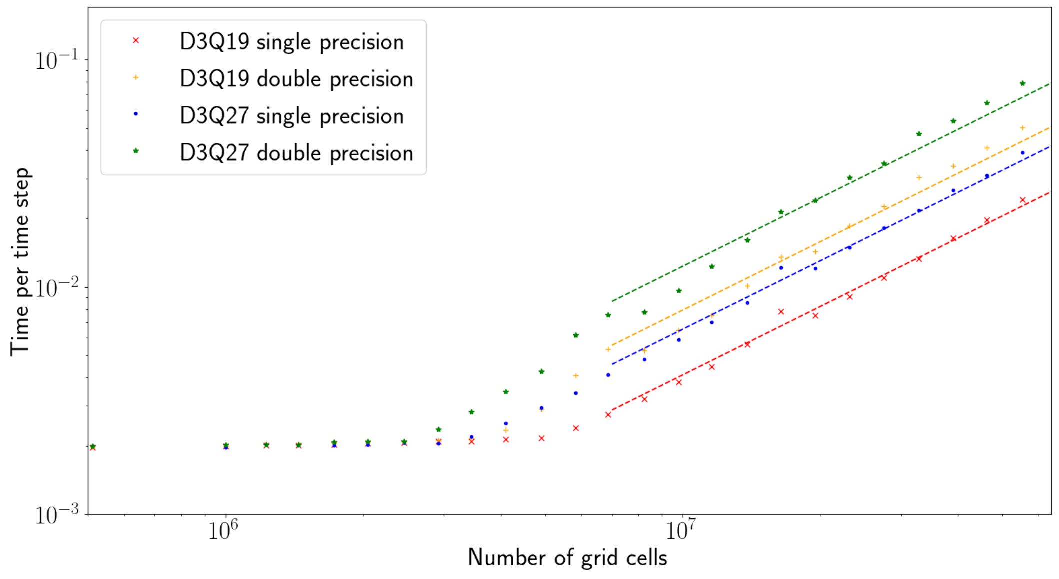

FIG. 2: Computation time per time step on GPU for different stencils and different floating point precisions.

in the boundary cells and the corresponding ghost layer are the same, otherwise the fluxes would not be equal, and the density would not be conserved. Here, key-based generators are particularly useful in parallel programming, as the same random numbers can be generated at both sides of the periodic boundary without the need for communication.

To benchmark the implementation, we measured the computation time per time step for the D3Q19 and D3Q27 stencil with single (4-byte) and double (8-byte) precision on a GPU (fig 2). For this we chose a system with only one species and no charges. The test system is run on a NVIDIA Tesla V100 as provided of the bwunicluster. The times are measured from the pystencils implementation and include the time the python script needs to start the kernels each time step, and due the slow start-up of the kernels this computation time is roughly equal for small systems. Excluding the kernel initiation time, one can see that this method scales linearly with system size. The different stencils also show that larger data processing results in slower simulation speeds. On the other hand it it is also evident that utilizing single-precision floating point arithmetic speeds up the simulation by a factor of 1.21, however we generally do not recommend running EK in single precision. The D3Q19 kernel is also faster then the D3Q27 by a factor of 1.57.

Let us now look at the memory bandwidth usage of the kernels: there are three kernels that need to be run every time step and for each of them the data of the cells needs to be transported from the memory to the processing unit and back. The three kernels do calculations of the fluxes, updating the density based on the fluxes due to the continuity equation, and synchronizing the density over the periodic boundary, respectively. Summing up the data that needs to be transported for each cell, we get 15 floating point numbers for the flux calculation kernel (1 for the density, 3 for the velocity, one for the potential and 10 for the D3Q19 staggered fluxes), 11 for the continuity kernel (1 for the density and 10 for the fluxes) and 1 for the synchronizing kernel. So for every cell on every time step, we need a throughput of 54 floating point numbers. For a system of 320^3 cells running in double precision, this corresponds to 14.2GB of data that need to be transported each time step. The NVIDIA Tesla V100 GPU used for this benchmark provides a memory bandwidth of 900GB/s. This should give us around 64 time steps per second. This is a very optimistic estimate because not all data needed for the calculations are on the local cell, but also on its neighboring cells. If this data is not present in the cache at that point, it needs to be transferred as well. Assuming, that every time when data from a neighboring cell is being accessed, this data also needs to be transported to the processor. For the flux calculation we need the information of the density of the adjacent cells, which adds nine floating point numbers to the data throughput per cell. For the continuity equation we also need the nine fluxes that are not stored in the local cell. So all in all with both transport directions we need a throughput of 90 floating point numbers per cell per time step. With this assumption, we arrive at a maximum of 38 time steps per second. Our measured rate for this test system is 33.0Hz, which is 87% of the theoretical limit.

The implementation in pystencils is very instructive and easy to use, as it is closer to the mathematical description of the algorithm than a hand-written implementation. Moreover, it will form the basis of diffusion-advection-reaction simulations in the upcoming versions of ESPResSo. The integration with a molecular dynamics package makes it possible to combine electrokinetic simulations with, e.g., colloids and polymers represented as particles. There are mainly three options to choose from. The first one is the direct coupling of point particles via interpolation of the forces it receives from the fluid and the species and *vice versa*. While this method is convenient and efficient, the accuracy and the discretization artifacts strongly depend on the interpolation scheme and the grid spacing used. The second way is to use raspberry particles [40], which are basically a collection of point particles to represent a single larger particle. Using the interpolation scheme with these raspberries, the accuracy increases due to the better description of the volume of the particle, however this comes with an increase in computational cost. The final method used in ESPResSo is to represent the particles as a moving boundary condition. While the principle behind this were developed by Ladd [41] for a LB fluid, it was further enhanced by Kuron *et al.* [7] to be applied to the electrokinetic method. This approach comes at even more computational cost, but the reward is an accurate description of the near and the far field of the fluid around the particles. This has been shown to be suitable for investigating (self-)electrophoresis problems [42].

V. VALIDATION

In this section, we demonstrate the correctness of our implementations using several scenarios for which the results are either known analytically or where they can be obtained using alternative methods.

A. Density distribution inside of a reaction slit pore system

Fig. 3 shows the concentration profile of two species in a slit pore system. On one wall the conversion of species A to species B is being catalysed, while the reverse happens on the other side. Such a system can then be described as

$$R_A(x) = \delta\left(x - \frac{d}{2}\right) k_B n_B(x) - \delta\left(x - \frac{d}{2}\right) k_A n_A(x) \tag{43}$$

$$R_B(x) = -R_A(x) \tag{44}$$

$$\frac{\partial n_A(x,t)}{\partial t} = D\frac{\partial^2 n_A(x,t)}{\partial x^2} + R_A(x) \tag{45}$$

$$\frac{\partial n_B(x,t)}{\partial t} = D\frac{\partial^2 n_B(x,t)}{\partial x^2} + R_B(x), \tag{46}$$

where $R_i(x)$ is the local reaction rate of species i, and k_A and k_B being the reaction rate constants at the different sides of the slit pore. Because we are investigating this system in equilibrium we can set the time derivatives to zero, which results in

$$D\frac{\partial^2 n_A(x)}{\partial x^2} = -R_A(x) \tag{47}$$

$$D\frac{\partial^2 n_B(x)}{\partial x^2} = -R_B(x). \tag{48}$$

Solving this source-drain-diffusion system yields a linear density profile for the steady-state solution:

$$n_A(x) = -\frac{2 \cdot n_0}{d + D \cdot \left(\frac{1}{k_A} + \frac{1}{k_B}\right)} \left(x - \frac{d}{2} + \frac{D}{k_A}\right) \tag{49}$$

where n_0 is the initial density, d the width of the slit pore, D the diffusion constant, and k_A and k_B are the reaction rates of the reaction $A \to B$ and $B \to A$.

The simulated system shows that the fluctuation have a significant impact on the density profile, however, averaging the density profile over 10000 uncorrelated time steps, yields a good agreement with the predicted density profile.

B. Ideal gas and Coulomb gas density distribution

Let us now turn to situations where thermal fluctuations are of importance. To determine whether the fluctuation model results in the correct physical behavior, we calculate the density distribution of an ideal gas. To this end we take a system of N non-interacting particles contained in a volume V. The probability that a sub-volume $v \subset V$ encloses n particles is given by

$$P(n) = \binom{N}{n} \left(\frac{v}{V}\right)^n \left(\frac{V-v}{V}\right)^{N-n}. \tag{50}$$

In the limit of $N \to \infty$ and $v \ll V$ this equation turns into the Poisson distribution

$$P(n) = \frac{\bar{n}^n e^{-\bar{n}}}{n!}, \tag{51}$$

where $\bar{n} = N\frac{v}{V}$. We compare the simulation data obtained from the fluctuating electrokinetic model against this equation. Additionally, we also compare the data to the density histogram obtained via a molecular dynamics simulation of non-interacting particles based on the Langevin equation. The results are displayed in figure 4, which show that the MD simulation and the electrokinetic model simulation are in good agreement with the theoretical prediction.

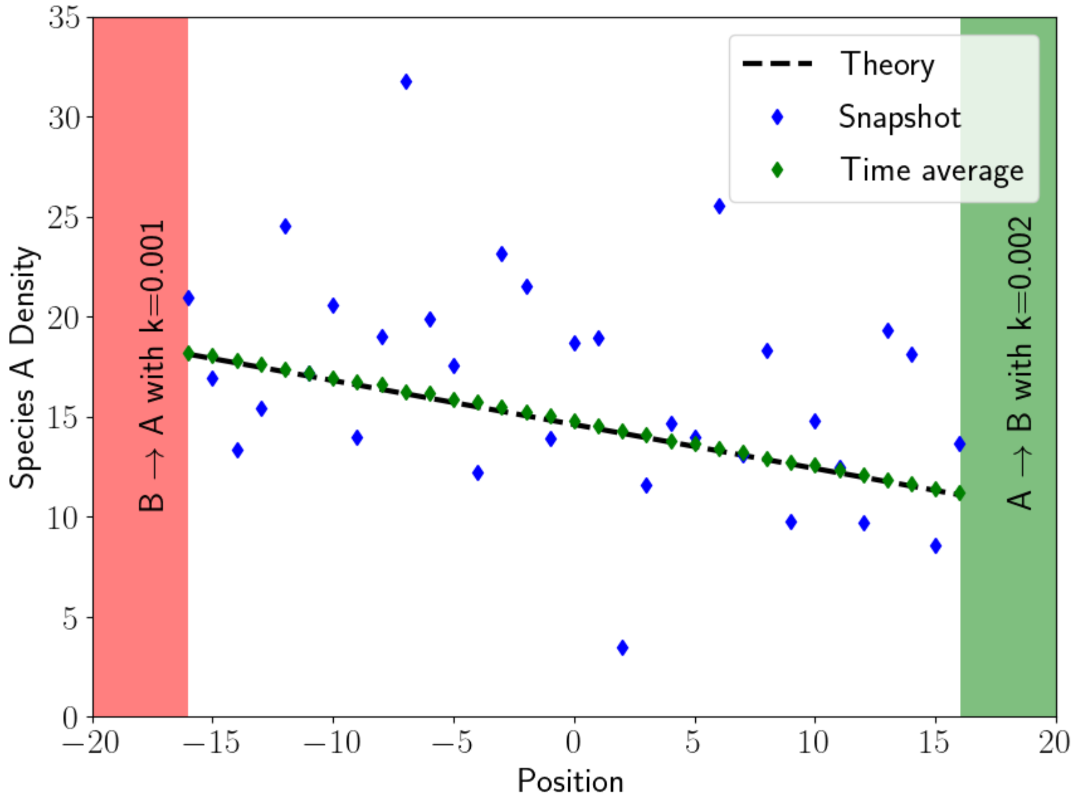

FIG. 3: Density distribution of a slit pore system, where on either side the opposite reaction takes place. For the time average 10000 uncorrelated time steps were used.

Next, we demonstrate that the electrostatic solver works in conjunction with thermal fluctuations. To this end, we examine the density histogram of a Coulomb gas containing an equal amount of positively and negatively charged ions. Since an analytical prediction is not available, we compared the electrokinetic simulation against a molecular dynamics simulation of charged particles. In the MD simulation, the charges have to be prevented from coming too close to each other. This is accomplished using a hard-sphere like Weeks-Chandler-Anderson potential [43]. The results of those simulation can be found in figure 5. While the electrokinetic model provides a good approximation towards the MD simulation, is has a slightly larger variance, which likely originates from the fact that the molecular dynamics simulation contains excluded volume interaction, whereas the electrokinetic model does not.

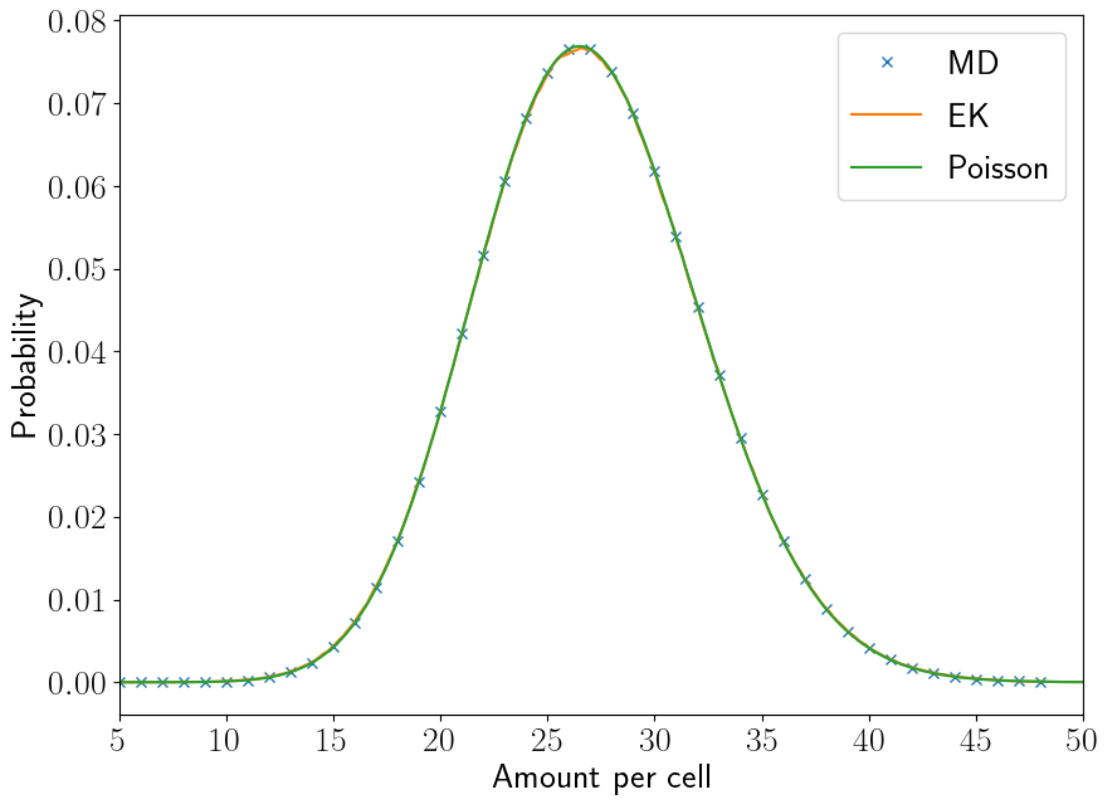

FIG. 4: Comparison of the density histogram of an ideal gas for a fluctuating electrokinetic simulation (orange), a MD simulation (blue), and the predicted theoretical distribution (green).

C. Effect of thermal fluctuations on the rate of chemical reactions

While the fluctuations do not change the equilibrium state of a system, they can have an influence on the dynamics of sufficiently complex systems. As an example, we consider the case of a reactive system where the reaction rate depends nonlinearly on the concentration of the species. In other words, the reaction order is not equal to one.

For example, let us look at the reaction of

$$2A \to B \tag{52}$$

We can then define the rate equation as

$$\frac{\mathrm{d}\rho_B(t)}{\mathrm{d}t} = -\frac{1}{2}\frac{\mathrm{d}\rho_A(t)}{\mathrm{d}t} = k\rho_A(t)^2, \tag{53}$$

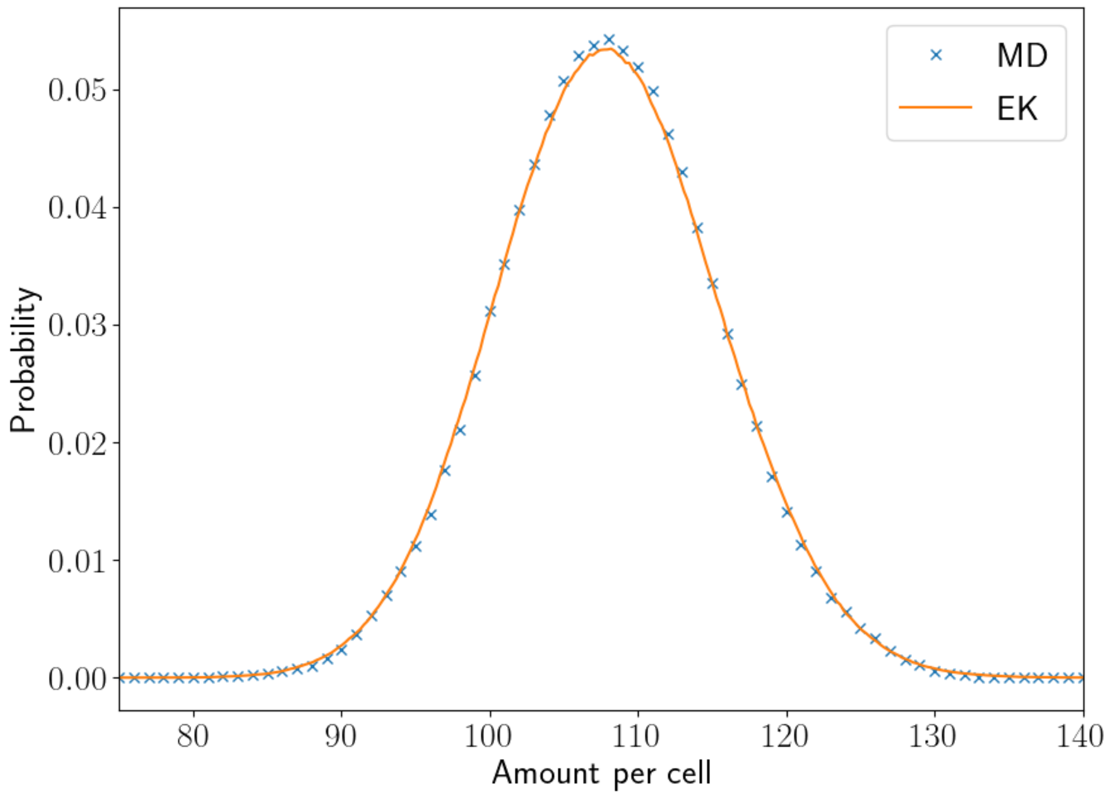

FIG. 5: Comparison of the density histogram for a Coulomb gas for the fluctuating electrokinetic simulation (orange) and the MD simulation (blue).

where k is the reaction rate constant. With the initial concentration of $\rho_A(0) = \rho_0$ and $\rho_B(0) = 0$, the product density is obtained as

$$\rho_B(t) = \frac{\rho_0}{2}\left(1 - \frac{1}{2kt\rho_0 + 1}\right). \tag{54}$$

However, when fluctuations take place, the density-dependent term $\rho_A(t)^2$ of equation 53 needs to be replaced by its expectation value $\langle\rho_A(t)^2\rangle$. Under the assumption that at any time during the reaction, ρ_A follows an ideal gas distribution, and approximating the Poisson distribution (Eq. 51) by a Gaussian distribution with mean $\langle\rho_A\rangle$ and variance $\sqrt{\langle\rho_a\rangle}$, the expectation value is given by

$$\langle\rho_A^2\rangle = \langle\rho_a\rangle^2 + \langle\rho_A\rangle. \tag{55}$$

Inserting this into Eq. 53 and solving the ODE yields a product density of

$$\widetilde{\rho}_B(t) = \frac{\rho_0}{2}\left(1 - \frac{1}{\mathrm{e}^{2kt}\left(\rho_0 + 1\right) - \rho_0}\right), \tag{56}$$

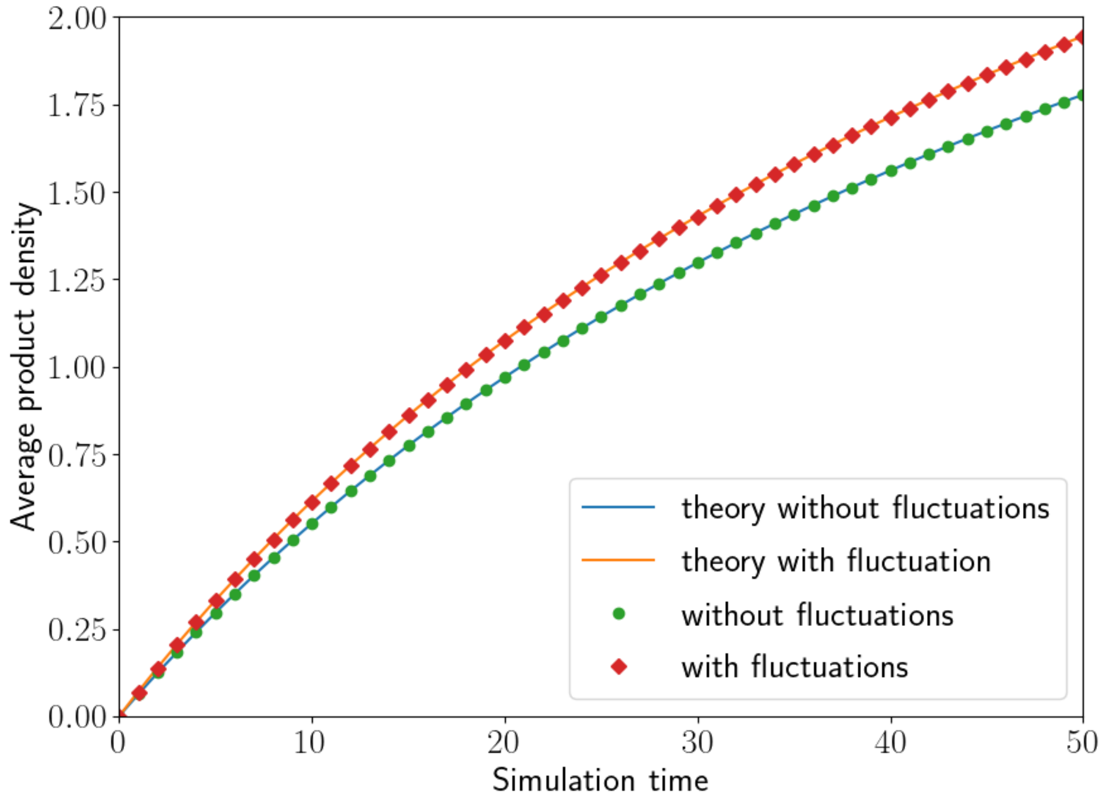

FIG. 6: Reaction product density plotted over time. The reaction is of the type $2A \to B$, where the reaction rate was set to $r \propto [A]^2$. The initial density was set to be $\rho_A(0) = 8$. The theory curves are from equations 54 and 56.

which differs from eq. 53. A plot of those product densities is displayed in figure 6. It can be seen that with fluctuations the amount of product B created over time is larger than without the fluctuations. Both cases are in good agreement with the theoretical predictions.

D. Reaction-advection-diffusion with fluctuations

To demonstrate that all of the aforementioned algorithms can be applied together, we have set up a model reaction-advection-diffusion system. It consists of a channel system in which the reaction $2A + B \to C$ takes place. We divided the inlet of the substrates into 3 equal parts. The upper and lower one are used for injecting species A, and in the middle

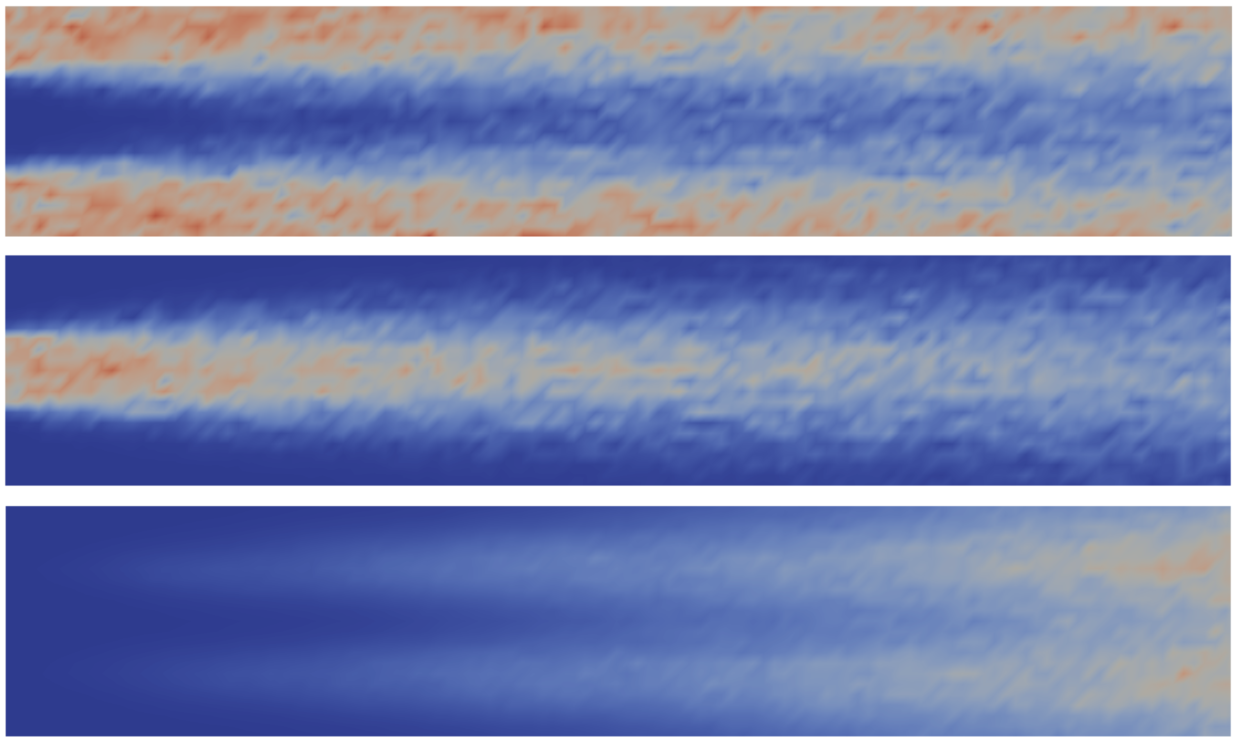

FIG. 7: Reactant and product densities in a channel with fluid driving them from left to right. The reaction taking place has the form $2A + B \to C$. The upper image shows the density of species A, the middle one that of species B and the bottom one is the product density C. As inlet and outlet boundary conditions we used constant density and velocity boundary condition.

part of the channel species B is injected.

The fluid advects the species through the channel, and the diffusion leads to a broadening of the density profile while the substrates move along the channel. In the growing area where species A and B overlap, the reaction can take place and the product C is created. The fluctuations add noise to all 3 density profiles. An image of this system is displayed in figure 7. This shows a broadening of the density profiles of species A and B along the flow direction due to diffusion. Species C can only be created at the interfaces between species A and B, however, due to the broadening of the density profiles in the later part of the channel, the intersection between those species increases, which also leads to more product being created. The fluctuations can be observed in all three species profiles.

VI. CONCLUSION AND OUTLOOK

Fluctuations play an important role in many dynamical processes on the nanoscale, and depending on the system under consideration, it may be necessary to include these fluctuation to have consistent physical description of reaction-advection-diffusion processes. We discussed in this article how to combine mesh-based solvers for thermalized electrokinetics, chemical reactions, and fluctuating hydrodynamics. Apart from the physical consistency, we demonstrated that thermal fluctuations need to be added to the electrokinetic solver to correctly reproduce the chemical reactions rates when the reaction order is not unity. Furthermore, when using an unthermalized electrokinetic model coupled to a thermalized lattice Boltzmann solver, the thermal energy of the lattice Boltzmann fluid would be depleted by the unthermalized electrokinetic solver. A similar depletion can be expected when a thermalized molecular dynamics simulation is coupled to an unthermalized electrokinetic or hydrodynamic solver.

We furthermore described two implementations of the reaction-advection-diffusion algorithms. One consists of manually written GPU solvers that are included as part of the simulation package ESPResSo. A second implementation is based on automatic code generation using pystencils and lbmpy, and is provided as a jupyter notebook in the data repository [44]. This can be used for research scenarios in various charged soft matter problems or reactive flow problems. In particular, we intend to apply this method to catalytic systems in mesoporous material, where the transport and diffusion of the reactants towards and away from a catalyst can be studied.

ACKNOWLEDGEMENT

This work was funded by the German Research Foundation (DFG) via SFB 1333 project number 358283783 and the cluster of excellence EXC 2075 (SimTech) project number 390740016. Furthermore the authors acknowledge support by the state of Baden-Württemberg through bwHPC.

DATA AVAILABILITY AND SUPPLEMENTARY MATERIAL

The data that support the findings of this study will be openly available in DaRUS - Uni Stuttgart, https://doi.org/10.18419/darus-2258.

[1] F. Capuani, I. Pagonabarraga, and D. Frenkel, The Journal of Chemical Physics **121**, 973 (2004).

[2] G. R. McNamara and G. Zanetti, Physical Review Letters **61**, 2332 (1988).

[3] X. He and L.-S. Luo, Physical Review E **56**, 6811 (1997).

[4] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, *The Lattice Boltzmann Method: Principles and Practice* (Springer International Publishing, Cham, 2017).

[5] I. Pagonabarraga, B. Rotenberg, and D. Frenkel, Physical Chemistry Chemical Physics **12**, 9566 (2010).

[6] B. Rotenberg and I. Pagonabarraga, Molecular Physics **111**, 827 (2013).

[7] M. Kuron, G. Rempfer, F. Schornbaum, M. Bauer, C. Godenschwager, C. Holm, and J. de Graaf, The Journal of Chemical Physics **145**, 214102 (2016).

[8] H. Daiguji, Y. Oka, and K. Shirono, Nano Letters **5**, 2274 (2005).

[9] G. Rempfer, S. Ehrhardt, N. Laohakunakorn, G. B. Davies, U. F. Keyser, C. Holm, and J. de Graaf, Langmuir **32**, 8525 (2016).

[10] G. Rempfer, S. Ehrhardt, C. Holm, and J. de Graaf, Macromolecular Theory and Simulations **26**, 1600051 (2017).

[11] N. Rivas, S. Frijters, I. Pagonabarraga, and J. Harting, The Journal of Chemical Physics **148**, 144101 (2018).

[12] C. Kim, A. Nonaka, J. B. Bell, A. L. Garcia, and A. Donev, The Journal of Chemical Physics **149**, 084113 (2018).

[13] S. Molins, D. Trebotich, C. I. Steefel, and C. Shen, Water Resources Research **48** (2012), 10.1029/2011wr011404.

[14] J. de Graaf, G. Rempfer, and C. Holm, IEEE Transactions on NanoBioscience **14**, 272 (2015).

[15] D. A. Kessler and H. Levine, Nature **394**, 556 (1998).

[16] A. Lemarchand and B. Nowakowski, EPL (Europhysics Letters) **94**, 48004 (2011).

[17] A. K. Bhattacharjee, K. Balakrishnan, A. L. Garcia, J. B. Bell, and A. Donev, The Journal of Chemical Physics **142**, 224107 (2015).

[18] D. Fange and J. Elf, PLoS Computational Biology **2**, e80 (2006).

[19] A. Donev, J. B. Bell, A. de la Fuente, and A. L. Garcia, Journal of Statistical Mechanics: Theory and Experiment **2011**, P06014 (2011).

[20] A. Donev, A. Nonaka, Y. Sun, T. Fai, A. Garcia, and J. Bell, Communications in Applied Mathematics and Computational Science **9**, 47 (2014).

[21] C. Kim, A. Nonaka, J. B. Bell, A. L. Garcia, and A. Donev, The Journal of Chemical Physics **146**, 124110 (2017).

[22] B. Dünweg, U. D. Schiller, and A. J. C. Ladd, Physical Review E **76**, 36704 (2007).

[23] B. Dünweg, U. D. Schiller, and A. J. C. Ladd, Computer Physics Communications **180**, 605 (2009).

[24] D. S. Dean, Journal of Physics A **29**, L613 (1996).

[25] M. Deng, F. Tushar, L. Bravo, A. Ghoshal, G. Karniadakis, and Z. Li, arXiv preprint arXiv:2107.05733 (2021).

[26] J.-P. Péraud, A. Nonaka, A. Chaudhri, J. B. Bell, A. Donev, and A. L. Garcia, Physical Review Fluids **1**, 074103 (2016).

[27] A. Donev, A. J. Nonaka, C. Kim, A. L. Garcia, and J. B. Bell, Physical Review Fluids **4** (2019), 10.1103/physrevfluids.4.043701.

[28] D. R. Ladiges, A. Nonaka, K. Klymko, G. C. Moore, J. B. Bell, S. P. Carney, A. L. Garcia, S. R. Natesh, and A. Donev, Physical Review Fluids **6** (2021), 10.1103/physrevfluids.6.044309.

[29] M. Bauer, J. Hötzer, D. Ernst, J. Hammer, M. Seiz, H. Hierl, J. Hönig, H. Köstler, G. Wellein, B. Nestler, and U. Rüde, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Association for Computing Machinery, New York, 2019).

[30] M. Bauer, H. Köstler, and U. Rüde, Journal of Computational Science **49**, 101269 (2021).

[31] F. Weik, R. Weeber, K. Szuttor, K. Breitsprecher, J. de Graaf, M. Kuron, J. Landsgesell, H. Menke, D. Sean, and C. Holm, European Physical Journal Special Topics **227**, 1789 (2019).

[32] B. Øksendal, *Stochastic Differential Equations* (Springer Berlin Heidelberg, 2003).

[33] D. T. Gillespie, **113**, 297 (2000).

[34] G. Rempfer, *A Lattice based Model for Electrokinetics*, Master's thesis, University of Stuttgart (2013).

[35] G. Rempfer, G. B. Davies, C. Holm, and J. de Graaf, The Journal of Chemical Physics **145**, 044901 (2016).

[36] H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm, Computer Physics Communications **174**, 704 (2006).

[37] D. Röhm, K. Kratzer, and A. Arnold, in *High Performance Computing in Science and Engineering '13*, edited by W. E. Nagel, D. H. Kroener, and M. M. Resch (Springer International Publishing, 2013) pp. 33–52.

[38] P. Colella, Journal of Computational Physics **87**, 171 (1990).

[39] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw (ACM Press, 2011).

[40] V. Lobaskin and B. Dünweg, New Journal of Physics **6**, 54 (2004).

[41] A. J. Ladd, Journal of Fluid Mechanics **271**, 311 (1994).

[42] M. Kuron, P. Kreissl, and C. Holm, Accounts of Chemical Research **51**, 2998 (2018).

[43] J. D. Weeks, D. Chandler, and H. C. Andersen, The Journal of Chemical Physics **54**, 5237 (1971).

[44] I. Tischler, C. Holm, R. Weeber, M. Kuron, F. Weik, and R. Kaufmann, "Replication data of c6 group for: "a thermalized electrokinetics model including stochastic reactions suitable for multiscale simulations of reaction-advection-diffusion systems"," (2021).