A Lattice based Model for Electrokinetics

Master's Thesis

Georg Rempfer

March 1, 2013

Supervisor:	Prof. Dr. Christian Holm
	Institute for Computational Physics
	University of Stuttgart
Co-Examiner:	Prof. Dr. Günter Wunner
	1st Institue for Theoretical Physics
	University of Stuttgart

The image on the title page shows a snapshot from a simulation of a drop of ink, spreading in a turbulent flow around an obstacle.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe bzw. unerlaubte Hilfsmittel angefertigt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, die den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe, dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen gegenstand eines anderen Prüfungsverfahrens gewesen ist, dass ich sie weder vollständig noch in Teilen bereits veröffentlich habe und, dass der Inhalt des elektronischen Exemplars mit dem des Druckexemplars übereinstimmt.

Stuttgart, 1. März 2013

Georg Rempfer

Contents

1 Introduction

Soft matter as a research topic extends over fields from a multitude of disciplines. Biological systems are nearly exclusively composed of soft matter. Nearly everything that animals eat is considered soft matter. Large parts of chemistry deal with soft matter, such as the whole field of polymers. Many materials, expecially modern ones, are soft matter. Due to the wide spectrum of systems traditionally considered to be soft matter, it is impossible to find a concise and precise criterion for what exactly falls within that category. One characteristic that the majority of soft matter systems share is that neither energy, nor entropy completely dominates their dynamics.

Similarily wide spread are the applications of soft matter systems. The most important application is probably the discovery and subsequent refinement of plastics, a material cheaper, easier to process, and more flexible in its application than any before. Also noteworthy are liquid crystals, and more recently e-ink displays, and of course, DNA sequencing using gel electrophoresis.

Compared to other fields in physics, soft matter is notorious for the absence of powerful analytical methods to describe these systems. The underlying reason for this is the lack of symmetry, due to the entropically induced randomness. Statistical mechanics offer a framework often applicable to soft matter systems, however, this usually only includes powerful tools for systems in thermodynamic equilibrium.

The lack of analytical tools makes soft matter systems prime candidates for computer simulations. While in the past, due to performance limitations, computer simulations of soft matter systems were limited to very crude models or very small system sizes, the continuously increased availability of high performance computing platforms for academic research and the advent of commodity hardware with considerable computational power has lead to the widespread development of software tools for modeling to soft matter systems and often freely available under open source licenses [5].

One such tool is ESPResSo, the **E**xtensible **S**imulation **P**ackage for **Res**earch on **So**ft matter [2]. It is a software package for **m**olecular **d**ynamics simulations, at this point in time mainly developed by Prof. Dr. Christian Holm's research group at the Institute for Computational Physics at the University of Stuttgart. Some of its core features are a number of efficient algorithms for calculating electrostatic potentials in systems with one-, two-, and three-dimensional periodicity, as well as the possibility of making use of massively parallel hardware architectures like modern super computers or GPUs (**G**raphics **P**rocessing **U**nits) [11, 28, 29].

Molecular dynamics simulations are a powerful tool for the investigation of soft matter systems. They are usually based on the calculation of particle trajectories as described by classical mechanics. A picture that, at least at room temperature, remains valid down to the micrometer and for many important systems, such as DNA, many proteins, electrolyte solutions, etc., even the nanometer scale.

The focus of ESPResSo's application lies on so-called coarse-grained molecular dynamics simulations, which combine whole groups of particles and model their interactions in some effective way, as opposed to all-atom simulations, which, as the name suggests, explicitly integrate the trajectory of every single atom in the system. This approach drastically reduces the degrees of freedom and with it the computational complexity, allowing for the simulation of systems on much larger length and time scales, than possible with current computers and more detailed models.

In the past, this approach comprised so-called bead-spring models, which combine groups of particles into a bead and the interaction of those beads through effective pair potentials, a method often applied in the study of polymer dynamics and lipid membranes. Another typical representative of this approach is the DPD (**D**issipative **P**article **D**ynamics) fluid model [16], which combines solvent particles into soft spheres of a significantly bigger radius and models the effective interactions using frictional and conservative interactions, to some extent recovering hydrodynamic effects.

In more recent years, however, the coupling of particle simulations with lattice based continuum models has been recognized as another way of modeling certain parts of soft matter systems with reduced resolution while maintaining the necessary microscopic information in others. One particular example of this kind is the lattice-Boltzmann method [7], a method based on a discretization of the Boltzmann equation, describing the evolution of the phase space density of fluid particles. It reproduces the full Navier-Stokes equations in certain limits typically valid in soft matter systems, and has advantages over other cutting edge approaches to numerically treat those equations, such as linear scaling and the possibility for relatively simple parallelization. The successful application of this method to problems like polyelectrolyte electrophoresis [11] and electroosmotic flow [27] has spawned the development of a multitude of similar schemes and extensions [14, 4].

While the replacing of explicit solvent particles with such a mesoscopic method reduces the number of degrees of freedom dramatically and thereby speeds the simulation up significantly or allows for the inclusion of hydrodynamic effects in situations where they have been completely neglected in the past. It does not however solve the problem of having to resolve a wide spectrum of different time and length scales. This constitutes a particularily pressing problem in the simulation of soft matter systems comprising particles of very different sizes, such as ions and colloids in colloidal electrophoresis. A method implicitly modeling the effects of ionic species on charged macro molecules by using a lattice-Boltzmann with a modified coupling scheme, developed by Hickey et. al. [14] and implemented in ESPResSo, provides a powerful demonstration of what time and length scales are reachable with very little computational effort, if one only resolves the time and length scales on which the dynamics of the macro molecules take place. Unfortunately this method only reproduces correct dynamics in the limit of infinitely small Debye-length.

To be able to simulate such systems efficiently, without restrictions on the Debye-length, I have implemented a method that treats the dynamics of both, the neutral and charged components of an electrolyte solution, using a continuum description. One such method particularily well suited for this task is the method proposed by Capuani, Pagonabarraga and Frenkel in 2004 [4], since it relies on a lattice-Boltzmann implementation which is extremely fast. Another benefit is that it operates on a regular cubic grid relying mostly on local interactions (apart from the long-ranged electrostatic interactions that is). This combination of properties makes this method, just like the lattice-Boltzmann method, an ideal candidate for exploiting the computational power provided by modern **G**raphics **P**rocessing **U**nits.

The following chapters will derive the mathematical model underlying this method from statistical mechanics, introduce the different mathematical and numerical schemes employed, document the implementation of the method, and verify the resulting dynamics by comparing them to analytical calculations. In addition to that, a number of alternative methods, deviating from the ones suggested in the original publication, as well as possible extensions to the model will be presented.

2 The Model

In my experience, any simulation method for a particular physical phenomena is comprised of three distinct layers of decreasing abstraction – distinct in the sense that each layer depends only on the previous one and a change in one layer can only require changes in the layers below it, but not above.

The first layer consists of the mathematical derivation of the governing equations from first principles. While the laws of nature should in principle reflect every known detail of any physical phenomena, the derived equations should not account for any effect, insignificant to the desired application, in order to minimize the necessary effort in the two subsequent layers. Even disregarding the different possibilities of neglecting certain effects, the choice of governing equations is in general not unique. Most often there are several mathematically equivalent formulations that represent the same facts, such as differential and integral formulations as well as formulations based on minimization principles. Each possibility, however, comes with a different set of known methods to solve the resulting equations.

The second layer consists of exactly one of those solution methods – a mathematical step-by-step recipe of how to obtain a solution for the chosen governing equations. The important considerations on this level are stability, runtime, scaling, and suitability for parallelization. Some methods might be mathematically very elegant but useless for implementation on a computer, due to excessive requirements of memory or computation time in the desired application, or due to instabilities under limited numerical precision.

The third and final layer consists of an implementation of the chosen method for solving the governing equations. The focus on this level lies on the optimization of the program for the given computer architecture and the trade off between efficiency and the ability to maintain and extend the program. Especially in an academic environment, characterized by massive personnel fluctuations, a faster low-level implementation usually comes with unjustifyable time requirements in becoming accustomed to the program.

Following this rationale, I will derive the governing equations in the following section 2.1 and summarize its properties in section 2.2. The subsequent sections 3 and 4 contain descriptions of the particular methods used to obtain solutions to these equations and the concrete implementation of those methods. The model as presented in this thesis was first published by Capuani, Pagonabarraga and Frenkel in their 2004 paper [4].

2.1 The Electrokinetic Equations

The central quantities in the continuum description are densities and fluxes of the coarse-grained particle species, each individually identified by an index k. For each species we define a scalar number density field $\rho_k(\vec{r}, t)$ and the respective flux density $\vec{j}_k(\vec{r}, t)$. For a given flux density, the time evolution of the corresponding density field is completely determined by mass conservation

$$\partial_t \rho_k = -\nabla \cdot \vec{j}_k \ . \tag{2.1}$$

The fluxes in turn are caused by two different effects – diffusion and advection. The advective contribution to the flux is given by the product of the species' density ρ_k with the advective velocity $\vec{v}$

$$\vec{j}_k^{\,\mathrm{adv.}} = \rho_\mathrm{k} \vec{v} \ . \tag{2.2}$$

This definition assumes that the particles comprising a species instantaneously assume the advective velocity, thereby neglecting inertial effects of individual particles, a situation known as the overdamped limit in a langevin simulation. This is valid on the time scales of the continuum model, which are much bigger than the time scale of the ballistic dynamics of individual particles.

The diffusive behaviour is best described in a reference frame co-moving with the local advective velocity $\vec{v}$. In this reference frame, the species' relative fluxes relax to a local equilibrium much faster than the time scale of the continuum model. This allows one to derive the diffusive fluxes using statistical mechanics. In a fashion closely related to density functional theory, we define a local free energy density

$$f(\rho_k(\vec{r})) = \underbrace{\sum_k k_\mathrm{B} T \, \rho_k(\vec{r}) \left[\log\left\{ \Lambda_k^3 \, \rho_k(\vec{r}) \right\} - 1 \right]}_{\text{ideal gas contribution}} + \underbrace{q_k \rho_k(\vec{r}) \, \Phi(\vec{r})}_{\text{electrostatic contribution}} \ , \tag{2.3}$$

consisting of an ideal gas part and a contribution to the internal energy coming from electrostatic interactions. Here k_B denotes the Boltzmann constant, T the temperature, Λ the thermal de Broglie wavelength, q_k the charge of a single particle of species k and $\Phi(\vec{r})$ the electrostatic potential at position $\vec{r}$.

In density functional theory, one would now go on to defining the total free energy functional through this free energy density. Minimizing this functional leads to the equilibrium distribution of the charged species. We however want to use this expression to derive the diffusive fluxes, which we need to model the transient behaviour of the system. Since the fluxes relative to the advective velocity are purely diffusive, they resemble a drift towards local equilibrium and can be expressed through a thermodynamic driving force. The respective thermodynamic potential (i.e. the chemical potential μ_k) is determined by the given free energy density through

$$\mu_k(\vec{r}) = \frac{\partial f(\rho_k)}{\partial \rho_k} = k_{\mathrm{B}}T \, \log(\Lambda_k^3 \, \rho_k(\vec{r})) + q_k \, \Phi(\vec{r}) \; , \tag{2.4}$$

which makes the diffusive flux

$$\begin{aligned} \vec{j}_k^{\,\mathrm{diff.}} &= \nu_k \cdot \rho_k \left[-\nabla \mu_k \right] = -k_{\mathrm{B}}T \, \nu_k \, \nabla \rho_k - \nu_k \, q_k \rho_k \, \nabla \Phi \\ &= -D_k \nabla \rho_k - \nu_k \, q_k \rho_k \, \nabla \Phi \; , \end{aligned} \tag{2.5}$$

where the mobility ν_k and the diffusion coefficient D_k of species k fulfill the Einstein-Smoluchowski relation $D/\nu = k_{\mathrm{B}}T$. This comes as no surprise, as it is exactly the same as the traditional diffusion equation, with the first term representing Fick's law of diffusion in the abscence of an external potential and the second term giving the additional flux due to an external potential.

The preceding derivations yield the following equations governing the propagation of the densities

$$\vec{j}_k = \vec{j}_k^{\,\mathrm{diff.}} + \vec{j}_k^{\,\mathrm{adv.}} = -D_k \nabla \rho_k - \nu_k \, q_k \rho_k \, \nabla \Phi + \rho_{\mathrm{k}} \vec{v} \; , \tag{2.6}$$

$$\partial_t \rho_k = - \nabla \cdot \vec{j}_k \; , \tag{2.1}$$

depending on the electrostatic potential Φ and the advective velocity $\vec{v}$.

The dynamics of charged species generally involve Maxwell's equations, whose application in this case would yield a model similar to those used in plasma physics. For soft matter systems, however, drastic simplifications are well justified. Due to the slow dynamics and low current densities in electrolytic solutions, one can neglect magnetic and electrodynamic effects completely, as long as the involved species are not magnetic. In this case the only remaining quantity, the electric field, is given by Maxwell's equation of electrostatics

$$\nabla \cdot \left(\varepsilon \vec{E} \right) = \sum_k q_k \rho_k \; , \tag{2.7}$$

where ε is the permittivity of the medium. While the case of spacially varying permittivity ε has interesting applications in soft matter systems, I want to restrict the model to constant permittivity for now, since the currently implemented method of solution as presented in section 3.3 depends on this property. An alternative method and its implementation without this restriction is presented in section 4.4. Expressing the electrostatic field through a potential $\vec{E} = -\nabla\Phi$, while taking advantage of ε being constant, said potential is determined by Poisson's equation

$$\Delta\Phi = -\frac{1}{\varepsilon}\sum_k q_k\rho_k = -4\pi\, l_\mathrm{B}\, k_\mathrm{B}T \sum_k q_k\rho_k \;, \tag{2.8}$$

where $\frac{q_o}{4\pi\,\varepsilon\, k_\mathrm{B}T} = l_\mathrm{B}$, the so-called Bjerrum length – the distance for which the interaction energy of two unit charges equals the thermal energy $k_\mathrm{B}T$.
Leaving out the contribution of advection, the governing equations state

$$\begin{aligned} \vec{j}_k &= -D_k\nabla\rho_k - \nu_k\, q_k\rho_k\, \nabla\Phi \;, & (2.5)\\ \partial_t\rho_k &= -\,\nabla\cdot\vec{j}_k \;, & (2.1)\\ \Delta\Phi &= -4\pi\, l_\mathrm{B}\, k_\mathrm{B}T \sum_k q_k\rho_k \;. & (2.8) \end{aligned}$$

In thermodynamic equilibrium, all net fluxes in the diffusion equation vanish, leaving us with

$$D_k\nabla\rho_k = -\nu_k\, q_k\rho_k\, \nabla\Phi \;, \tag{2.9}$$

for which the solution is given by the well known Boltzmann distribution

$$\rho_k(\vec{r}) = \rho_k(\vec{r}_0)\cdot\exp\left(-\frac{q_k[\Phi(\vec{r}) - \Phi(\vec{r}_0)]}{k_\mathrm{B}T}\right) \;. \tag{2.10}$$

Requiring self-consistency of the electrostatic potential Φ and the densities ρ_k with respect equations (2.8) and (2.10) yields the Poisson-Boltzmann equation

$$\Delta\Phi = -4\pi\, l_\mathrm{B}\, k_\mathrm{B}T \sum_k q_k\rho_k(\vec{r}_0)\cdot\exp\left(-\frac{q_k[\Phi(\vec{r}) - \Phi(\vec{r}_0)]}{k_\mathrm{B}T}\right) \;. \tag{2.11}$$

This shows that using this model, we will be able to recover the equilibrium densities from Poisson-Boltzmann theory as the stationary states in our model. Also it demonstrates that, despite describing transient behaviour, the constraints of this model are the same as for Poisson-Boltzmann theory, since both are based on the same underlying statistical mechanics description and make the same assumptions. Those assumptions are, firstly that smooth densities resulting in a smooth potential instead of one with singularities at every discrete charges position are a suitable description, and secondly that all non-ideal contributions to the local free energy but the electrostatic one are negligible, most notably, there is no excluded volume. Those assumptions restrict the validity of the model to monovalent ions at moderate densities.

The last thing missing is a model for the advective velocity $\vec{v}$. Since the main application of the resulting algorithm is going to be electrolytic solutions, we would like the mixture of the species k to exert hydrodynamic behaviour. Momentum conservation for a continuum fluid, expressed by the Navier-Stokes equations, states

$$\frac{D\vec{v}}{Dt}\rho = -\nabla p + \nabla\cdot \overset{\leftrightarrow}{T}\,, \tag{2.12}$$

where ρ is the density of the fluid, D denotes the material derivative (i.e. the derivative in a locally co-moving reference frame), p denotes the pressure, $\overset{\leftrightarrow}{T}$ the deviatoric stress and $\vec{f}$ an external body force. Expressing the acceleration on the left hand side in a stationary reference frame, and expressing the density of the fluid in terms of the densities of the individual species $\rho = \sum_k \rho_k$, we arrive at

$$\left(\frac{\partial\vec{v}}{\partial t} + \vec{v}\cdot\vec{\nabla}\vec{v}\right)\sum_k \rho_k = -\nabla p + \nabla\cdot \overset{\leftrightarrow}{T}\;. \tag{2.13}$$

Typically one would have to impose a separate constraint for conservation of mass. In this case this is unneccessary though, since the fluid density is given by the individual species' densities, for which equation (2.1) already ensures this condition. Under the assumptions that the fluid is isotropic and compressible, that the only deviatoric stresses arise from sheering and compression, and that the sheer stress is proportional to the sheer rate (i.e. the fluid is newtonian), the equations further simplify to

$$\left(\frac{\partial\vec{v}}{\partial t} + \vec{v}\cdot\vec{\nabla}\vec{v}\right)\sum_k \rho_k = -\nabla p + \eta\vec{\Delta}\vec{v} + (\eta/3 + \eta_{\mathsf{b}})\nabla(\nabla\cdot\vec{v})\,, \tag{2.14}$$

with the sheer viscosity η, the bulk viscosity η_{b} and the vector Laplacian $\vec{\Delta}$ – in Cartesian coordinates a component wise scalar Laplacian. In order for the model to be consistent, we need to prescribe the pressure as it results from the free energy density (2.3)

$$p = \sum_k \rho_k(\vec{r})\mu_k(\vec{r}) - f(\rho_k(\vec{r})) \; . \tag{2.15}$$

When applying the gradient, one has to pay attention to the fact that f is only a function of the densities ρ_k

$$\begin{aligned}\nabla p(\vec{r}) &= \sum_k \rho_k(\vec{r})\nabla\mu_k(\vec{r}) + \sum_k \mu_k(\vec{r})\nabla\rho_k(\vec{r}) - \sum_k \underbrace{\frac{\partial f(\rho_k)}{\partial \rho_k}}_{=\mu_k} \cdot \nabla\rho_k(\vec{r}) \\ &= \underbrace{k_\mathrm{B}T\sum_k \nabla\rho_k}_{\text{ideal gas part}} \quad + \quad \underbrace{q_k\rho_k\nabla\Phi}_{\text{electrostatic contribution}} \; . \end{aligned} \tag{2.16}$$

This results was also to be expected from a less technical perspective, since equation (2.13) represents conservation of momentum and therefore has to take into account all forces. By constrution, other than the purely hydrodynamic effects, our model only contains forces present in an ideal gas and the electrostatic force, which correspond to the two terms in the derived expression for the pressure gradient. With this, we finally arrive at the equation for the evolution of the advective velocity and complete our model

$$\left(\frac{\partial \vec{v}}{\partial t} + \vec{v}\cdot\vec{\nabla}\vec{v}\right)\sum_k \rho_k = -k_\mathrm{B}T\sum_k \nabla\rho_k - q_k\rho_k\nabla\Phi + \eta\vec{\Delta}\vec{v} + (\eta/3 + \eta_\mathsf{b})\nabla(\nabla\cdot\vec{v}) \; . \tag{2.17}$$

2.2 Summary of the Model Properties

The previous section 2.1 contains the derivation of the governing equations of the model. In summary, those are given by

$$\frac{\partial \rho_k}{\partial t} = -\nabla \cdot \vec{j}_k \ , \tag{2.1}$$

$$\vec{j}_k = -D_k \nabla \rho_k - \nu_k \, q_k \rho_k \, \nabla \Phi + \rho_\mathrm{k} \vec{v} \ , \tag{2.6}$$

$$\Delta \Phi = -4\pi \, l_\mathrm{B} \, k_\mathrm{B} T \sum_k q_k \rho_k \ , \tag{2.8}$$

$$\left(\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \vec{\nabla} \vec{v} \right) \sum_k \rho_k = -k_\mathrm{B} T \sum_k \nabla \rho_k - q_k \rho_k \nabla \Phi + \eta \vec{\Delta} \vec{v} + (\eta/3 + \eta_\mathsf{b}) \nabla (\nabla \cdot \vec{v}) \ . \tag{2.17}$$

They define relations between the following observables

ρ_k	the number density of the particles of species k,
$\vec{j}_k$	the number density flux of the particles of species k,
Φ	the electrostatic potential,
$\vec{v}$	the advective velocity,

and input parameters

D_k	the diffusion constant of species k,
ν_k	the mobility of species k,
q_k	the charge of a single particle of species k,
l_B	the Bjerrum-length,
$k_\mathrm{B}T$	the thermal energy given by the product of Boltzmann's constant k_B and the temperature T,
η	the dynamic viscosity of the fluid,
$\eta_\mathbf{b}$	the bulk viscosity of the fluid.

The temperature T, and diffusion constants D_k and mobilities ν_k of individual species are linked through the Einstein-Smoluchowski relation $D_k/\nu_k = k_\mathrm{B}T$.
This set of equations, combining diffusion-advection, electrostatics, and hydrodynamics is conventionally referred to as the *Electrokinetic Equations*.

The derivation as given in the previous section 2.1 required a number of assumptions and simplifications, which put some broad borders on the applicability of the model.

- On the coarse time and length scale of the model, the dynamics of the particle species can be described in terms of smooth density distributions and potentials as opposed to the microscale where highly localized densities cause singularities in the potential.

  In most situations, this restricts the application of the model to species of monovalent ions, since ions of higher valency typically show strong condensation and correlation effects – the localization of individual ions in local potential minima and the subsequent correlated motion with the charges causing this minima.

- Only the entropy of an ideal gas and electrostatic interactions are accounted for. In particular, there is no excluded volume.

  This restricts the application of the model to monovalend ions and moderate charge densities. At higher valencies or densities, overcharging and layering effects can occur, which lead to non-monotonic charge densities and potentials, that can not be covered by a mean-field model such as Poisson-Boltzmann or this one.

  Even in salt free systems containing only counterions, the counterion densities close to highly charged objects can be overestimated when neglecting excluded volume effects. Decades of the application of Poisson-Boltzmann theory to systems of electrolytic solutions, however, show that those conditions are fulfilled for monovalent salt ions (such as sodium chloride or potassium chloride) at experimentally realizable concentrations.

- Electrodynamic and magnetic effects play no role. Electrolytic solutions fulfill those conditions as long as they don't contain magnetic particles.

- The diffusion coefficient is a scalar, which means there can not be any cross-diffusion. Additionally, the diffusive behaviour has been deduced using a formalism relying on the notion of a local equilibrium. The resulting diffusion equation, however, is known to be valid also far from equilibrium.

- The temperature is constant throughout the system.

- The density fluxes instantaneously relax to their local equilibrium values. Obviously one can not extract information about processes on length and time scales not covered by the model, such as dielectric spectra at frequencies, high enough that they correspond to times faster than the diffusive time scales of the charged species.

3 Computational Methods

This section introduces the individual methods used to solve the equations (2.1) and (2.6), which account for diffusive and advective mass transport, Poisson's equation (2.8) for determining the electrostatic potential, and the Navier-Stokes equations (2.17) representing conservation of momentum.

To solve the Navier-Stokes equations (2.17), we use the so-called **L**attice-**B**oltzmann method. ESPResSo already contains two separate implementations of the method [2]. One applies a domain decomposition scheme and uses the **M**essage **P**assing **I**nterface to be able to carry out simulations on large clusters of CPUs, connected through a network. The other one is based on NVIDIA's CUDA framework and runs on a single GPU [29]. Since this model for electrokinetics is supposed to extend the existing GPU LB method originally developed by Dominic Röhm in his diploma thesis [28], special care has to be taken to ensure the compatibility of the other methods with this implementation. An introdcution to the LB method is given in section 3.1.

The electrostatics problem represented by Poisson's equation (2.8) is treated using a **F**inite-**D**ifference scheme in conjunction with an FFT based method to solve the resulting system of equations, based on idea presented by Gonsalves in [9]. This approach deviates from the one proposed in a publication by Capuani et al. [4] who suggest an FD solver based on **S**uccessive **O**ver-**R**elaxation as described in an earlier publication by Horbach and Frenkel [17].

For the calculation of diffusive and advective mass transport, modeled by the equations (2.1) and (2.6), we implement a scheme based on the calculation of fluxes between nodes of the regular cubic grid used by the other two solvers. The diffusive contribution to the flux is obtained using a finite-difference scheme to obtain the necessary gradients, taking into account nearest and next-nearest neighbors in order to avoid lattice artifacts in transient processes. The advective flux contribution is calculated from a displacement scheme by assuming a constant density in a lattice cell, moving with the local advective velocity and calculating its intersection with neighboring cells after one time step. This method is simply the method suggested in [4].

3.1 Lattice-Boltzmann

Historically, the development of the lattice-Boltzmann method dates back to the second half of the 1980s, with the so-called lattice gas automata, a simplistic lattice model for dilute gas dynamics. This model consists of a regular cubic grid comprising nodes carrying populations of gas molecules with a finite number of possible velocities. These velocities are chosen such that they transport a particle from one node to one of its neighbors in one time step, a process called streaming. In between streaming steps, a scheme of swapping the populations velocities is applied in order to model collisions. Initially, each velocity population on a node could only consist of one molecule. To overcome lattice artifacts introduced by this limitation, this was changed so that the local velocity populations are represented by a positive real valued density, which eventually led to the development of the LB method. In 2001, a comprehensive introduction to LB following this historical development was given by Succi [32].

It was later discovered that the lattice-Boltzmann equation can also formally be derived as a discrete lattice approximation to the continuum Boltzmann equation for fluids, providing a sound mathematical basis to the method. Further analysis based on the Chapman-Enskog expansion shows that the LB equation is equivalent to the full Navier-Stokes equations for low Mach numbers, or in other terms, small density variations. In this limit it reproduces the equation of state of an ideal gas. Dünweg gave a comprehensive analysis of the LB method and many of its extensions relevant to soft matter simulations in 2009 [7]. A good overview of the state of the development and its possible applications was given by Succi in 2010 [33].

In kinetic gas theory, the Boltzmann equations describes the dynamics of particles by means of their phase space distribution, given by the phase space distribution function $f(\vec{x}, \vec{p}, t)$, denoting the number density of particles at a position $\vec{x}$ with momentum $\vec{p}$ at time t. The evolution of this density is given by

$$\frac{d}{dt} f(\vec{r}, \vec{p}, t) = \left(\frac{\vec{p}}{m} \cdot \nabla_{\vec{r}} + \vec{F} \cdot \nabla_{\vec{p}} + \frac{\partial}{\partial t} \right) f(\vec{r}, \vec{p}, t) = C[f](\vec{r}, \vec{p}, t) \tag{3.1}$$

where the connection between velocity and momentum is given by $\frac{d\vec{x}}{dt} = \vec{p}/m$, where m the mass of a single particle, $\vec{F}$ is the externally applied force, and C is the collison operator, that accounts for changes in the distribution function due to collisions. In principle, the collision operator depends on the miscroscopic detail of the particles' collisions, bringing back the whole complexity of many particle dynamics. In practice, one makes an assumption about the shape of this operator, for the purposes of deriving the **L**attice-

Boltzmann **E**quation, one assumes that the local momentum distribution linearly decays towards the equilibrium distribution for the given local average velocity $\vec{v}_{\text{meso}}$ and average local density ρ_{meso}, with a relaxation time τ, yielding the Bhatnagar-Gross-Krook collison operator

$$C[f](\vec{r},\vec{p},t) = \frac{1}{\tau}\left\{f^{\text{eq}}(\vec{p},\rho_{\text{meso}},\vec{v}_{\text{meso}}) - f(\vec{r},\vec{p},t)\right\} \ . \tag{3.2}$$

Since the equilibrium distribution of momenta f^{eq} is known to be a Maxwell-Boltzmann distribution, this leaves no unknowns. Discretizing equation (3.1), using a regular cubic grid and a finite number of discrete velocities chosen as such that they transport particles from one lattice node to one of its neighbors in one time step results in

$$f_i(\vec{r}+\vec{c}_i\Delta t, t+\Delta t) = f_i(\vec{r},t) + \frac{1}{\tau}\left\{f_i^{\text{eq}} - f_i(\vec{r},t)\right\} + \Psi_i \ , \tag{3.3}$$

where f_i denotes the densities of the particle populations assuming the velocity $\vec{c}_i$, and Ψ_i models the density change of of the population with velocity $\vec{c}_i$ due to external forces. The mesoscopic values for the density and velocity can be recovered from the microscopic particle populations through

$$\rho = m\sum_i f_i \tag{3.4}$$

$$\vec{v} = \frac{m}{\rho}\sum_i f_i\vec{c}_i \tag{3.5}$$

Using those, the lattice equilibrium distribution is given by a second order expansion of the continuum Maxwell-Boltzmann distribution

$$f_i^{\text{eq}} = a_i\rho\left(1 + \frac{\vec{v}\cdot\vec{c}_i}{c_s^2} + \frac{(\vec{v}\cdot\vec{c}_i)^2}{2c_s^4} - \frac{v^2}{2c_s^2}\right) \ , \tag{3.6}$$

with prefactors a_i and the speed of sound c_s depending on the specific lattice geometry. Lattice geometries in LB are typically characterized by an identifier of the shape DmQn. This stands for a regular cubic lattice in m dimensions with n discrete velocities, connecting a node with its neighbors. In order to minize the computational effort, a minimal choice of the number of discrete velocities is desired. In order to avoid lattice artifacts, at least 18 velocities connecting a node to its nearest and next nearest neighbors are necessary. For reasons given in detail in [7], a better choice is the D3Q19 model, whiches grid is depicted in figure 3.1.

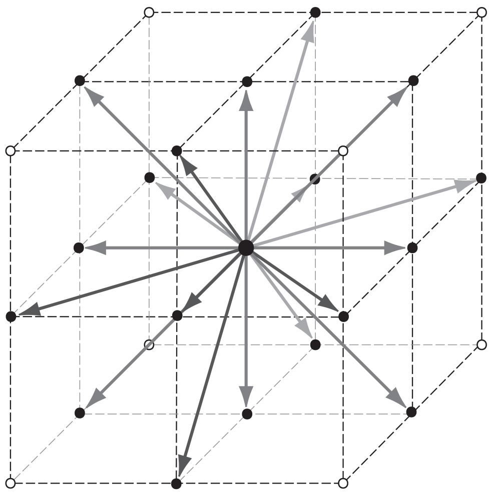

Figure 3.1: One node from a D3Q19 LB grid, depicting the 19 discrete velocities, connecting the node with itself, its nearest, and next nearest neighbors. Image inspired by Kai Grass [11].

One problem of the above choice of the collision operator is, that it only contains one free parameter, which does not suffice to set the fluid's sheer and bulk viscosity individually. The overcome this limitation, one generalizes the relaxation rate $1/\tau$ to a linear operator, resulting in the so-called **M**ulti-**R**elaxation **T**ime collison operator. In the discretized case, this linear operator can be expressed through a matrix. The LBE then states

$$f_i(\vec{r}+\vec{c}_i\Delta t, t+\Delta t) = f_i(\vec{r},t) + \sum_k L_{ik}\left\{f_k^{\mathrm{eq}} - f_k(\vec{r},t)\right\} + \Psi_i\ , \tag{3.7}$$

How L operates is best described in a base consisting of vectors corresponding to the mesoscopic physical quantities. Those vectors can be constructed in such a way, that they are orthogonal, and are referred to as *modes* and the respective space is called *mode space*. In accordance with [7], the modes can be chosen so that

$$\begin{aligned}
&\rho = m_0, \\
&j_x = m_1 c, \quad \pi_{xx} = (m_0+m_4+m_5)c^2/3, \quad && \pi_{xy} = m_7 c^2, \\
&j_y = m_2 c, \quad \pi_{yy} = (2m_0+2m_4-m_5+3m_6)c^2/6, \quad && \pi_{yz} = m_8 c^2, \\
&j_z = m_3 c, \quad \pi_{zz} = (2m_0+2m_4-m_5-3m_6)c^2/6, \quad && \pi_{zx} = m_9 c^2,
\end{aligned} \tag{3.8}$$

where m_0 to m_9 are the contributions of the first nine modes to the local nodes' populations' densities, ρ is the mesoscopic density as in (3.4), $\vec{j} = m\vec{v}$ the momentum density related to the mesoscopic velocity from (3.5), $\overleftrightarrow{\pi}$ the stress tensor with diagonal bulk and off-diagonal sheer components. Here c denotes the ratio of grid spacing to time step, the maximum possible velocity of particles in this model. The remaining nine mode space components m_{10} to m_{18} do not correspond to any mesoscopic quantities and are therefore called ghost modes.

One can now conveniently define the relaxation rates in mode space. Since the modes m_0 to m_3 correspond to the mesoscopic density and momentum, which are conserved during collisions, their relaxation rate is set to 0. Isotropy of the fluid dictates, that there are only two distinct relaxation rates, γ_b and γ_s, for the three bulk stress components and the three sheer stress components, respectively. To save computation time, the ghost modes are not relaxed. The relaxation rates are related to the bulk and sheer viscosity through

$$\eta_\text{b} = \frac{\Delta t\, \rho c_s^2}{3} \cdot \frac{1+\gamma_\text{b}}{1-\gamma_\text{b}}, \tag{3.9}$$

$$\eta_\text{s} = \frac{\Delta t\, \rho c_s^2}{2} \cdot \frac{1+\gamma_\text{s}}{1-\gamma_\text{s}}, \tag{3.10}$$

where Δt is the time step. The term Ψ_i in eq. (3.7), representing the influence of an external force has to fulfill two conditions

$$\sum_i \Psi_i = 0, \tag{3.11}$$

$$\sum_i \Psi_i \vec{c}_i = \vec{f}\,\Delta t, \tag{3.12}$$

where the first one ensures conservation of mass and the second one represents the proper momentum change due to the external volume force density $\vec{f}$.

Analysis of the LB method with the MRT collison operator as presented above, by means of the Chapman-Enskog expansion as presented in detail by Dünweg [7] yields, that the method reproduces flow consistent with

$$\rho\left(\dot{\vec{v}} + \vec{v}\cdot\vec{\nabla}\vec{v}\right) = -c_s^2\nabla\rho + \eta\vec{\Delta}\vec{v} + \left(\tfrac{1}{3}\eta + \eta_{\mathrm{b}}\right)\nabla(\nabla\cdot\vec{v}) + \vec{f}\,, \tag{3.13}$$

in the continuum limit and for low Mach numbers, which means, for small density variations. Choosing $c_s^2 = k_{\mathrm{B}}T$ makes this equivalent to the desired Navier-Stokes equations modeling flow of an isotropic Newtonian fluid with the equation of state of an ideal gas

$$\rho\left(\dot{\vec{v}} + \vec{v}\cdot\vec{\nabla}\vec{v}\right) = -\nabla p + \eta\vec{\Delta}\vec{v} + \left(\tfrac{1}{3}\eta + \eta_{\mathrm{b}}\right)\nabla(\nabla\cdot\vec{v}) + \vec{f}\,. \tag{3.14}$$

By identifying nodes on the boundaries of the cubic grid with the corresponding nodes on the opposite side, one can trivially implement periodic boundary conditions. Other boundary conditions, such as fixed velocity or fixed pressure, are more complex to implement. In addition to that, boundaries with vanishing velocity, so-called no-slip boundary conditions with more complex geometries are often required to model rigit objects.

One simple way to realize this is by introducing so-called bounce-back nodes in the grid. Instead of streaming populations along their respective velocities as dictated by the LBE equation (3.7), they reverse the populations velocities and transfer then back to the originating node as shown in figure 3.2. This mimics the existence of particle populations on the bounce-back node, with reversed velocities, relative to the node streaming populations onto said bounce-back node. Linear interpolations between the two nodes then produces a vanishing mesoscopic velocity at a point between the two nodes. For planar walls and flow with translational symmetry in directions parallel to the wall, the point of vanishing velocity lies exactly in the center between regular and bounce-back nodes. In all other cases the this position depends on the boundary geometry, and the flow profile.

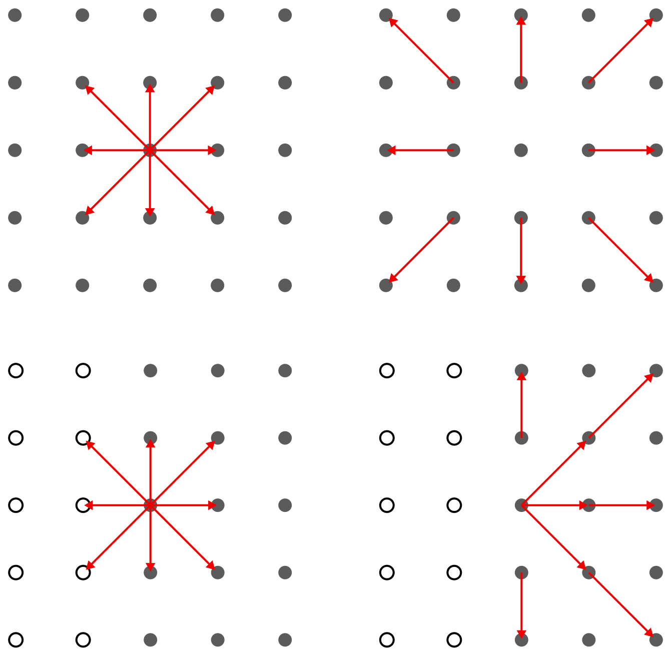

Figure 3.2: Illustration of the streaming of particle populations, identified by their velocities (red arrows) in a two dimensional grid of LB nodes (grey circles).
Top: Populations of the center node in the unmodified scheme before (*left*) and after (*right*) streaming.
Bottom: Populations of the center node with the bounce-back scheme before (*left*) and after (*right*) streaming. Empty circles represent bounce-back nodes.

3.2 Summary of the Lattice-Boltzmann Method

The lattice-Boltzmann method is a mesoscopic method of modeling hydrodynamics, which is based on a discretized version of the Boltzmann equation, which describes the dynamics of particles by means of their phase space density. This lattice-Boltzmann equations reads

$$f_i(\vec{r} + \vec{c}_i \Delta t, t + \Delta t) = f_i(\vec{r}, t) + \sum_k L_{ik} \left\{ f_k^{\textsf{eq}} - f_k(\vec{r}, t) \right\} + \Psi_i \ . \tag{3.7}$$

It defines relations between the following observables

f_i	the number density of particles assuming the velocity $\vec{c}_i$,
$\vec{r}$	the position of a lattice node on a regular cubic grid,
$\vec{c}_i$	the velocities transporting particle populations from one lattice node to its neighbors in one time step, indexed by i,
t	the time,
Δt	the discrete time step,
$\overset{\leftrightarrow}{L}$	a matrix, diagonal in mode space, such that locally mass and momentum are conserved and the stress tensor relaxes to the equilibrium value for the local mesoscopic density and velocity,
$f_k^{\textsf{eq}}$	the equilibrium value for the density of the population k for a given local mesoscopic density and velocity,
Ψ_i	the change in density of population i due to external forces.

Modes are the components of the vector containing all densities of the local populations, represented in a base whose base vectors correspond to the mesoscopic density, components of the velocity, and components of the stress tensor on the local node. The matrix $\overset{\leftrightarrow}{L}$ is diagonal in this base and has eigenvalues 0 for the density and momentum components, such that these are conserved. The bulk stress and sheer stress components share one eigenvalue each, which are smaller than 1, which results in a relaxation towards the equilibrium value determined by $f^{\textsf{eq}}$. Their values are related to the bulk and sheer viscosity η_b and η_s through

$$\eta_\mathsf{b} = \frac{\Delta t\,\rho c_s^2}{3} \cdot \frac{1+\gamma_\mathsf{b}}{1-\gamma_\mathsf{b}}, \tag{3.9}$$

$$\eta_\mathsf{s} = \frac{\Delta t\,\rho c_s^2}{2} \cdot \frac{1+\gamma_\mathsf{s}}{1-\gamma_\mathsf{s}}, \tag{3.10}$$

where c_s is the speed of sound, $\rho = \sum_i f_i$ the mesoscopic density, and γ_b and γ_b are the relaxation parameters of the bulk and sheer stresses.

The local equilibrium distribution is a second order expansion of a Maxwell-Boltzmann distribution in the momenta, given by

$$f_i^\mathsf{eq} = a_i \rho \left(1 + \frac{\vec{v}\cdot\vec{c}_i}{c_s^2} + \frac{(\vec{v}\cdot\vec{c}_i)^2}{2c_s^4} - \frac{v^2}{2c_s^2} \right) , \tag{3.6}$$

with lattice geometry dependent prefactors a_i and the local mesoscopic velocity $\vec{v}$. A typical choice with one of the smalles possible sets of discrete velocities is the so-called D3Q19 lattice (3 dimensions, 19 discrete velocities). The local populations consist of a resting component and components being transported to nearest and next nearest neighbors in the streaming step.

This model is proven to reproduce the Navier-Stokes equations in the continuum limit for low Mach number – or in other terms, for low density variations. Since it is based on purely local interactions, it exhibits linear scaling and is relatively simple to parallelize using domain decomposition.

3.3 Finite-Difference Electrostatics

According to the model, the electrostatic interactions are described in terms of an electrostatic potential Φ, which can be obtained from the charge distribution, given by the distributions of the individual species, through Poisson's equation

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial y^2}\right)\Phi(x,y,z)=-4\pi\, l_\mathrm{B}\, k_\mathrm{B}T\sum_k q_k\rho_k(x,y,z)\ , \tag{3.15}$$

where x, y, and z are the individual components of a vector in cartesian coordinates. This consitutes a single linear, elliptic, **P**artial **D**ifferential **E**quation in three dimensions. In the original paper proposing this model, the authors suggest using **F**inite **D**ifferences to discretize the equation and then use a **S**uccessive **O**ver-**R**elaxation scheme to solve the resulting system of linear equations [4]. This combination is known to be a numerically stable method to solve Poisson's equation and relatively easy to implement. While other methods such as the **F**inite **E**lement **M**ethod yield more accurate results with fewer grid points and have advantages in treating complex boundary geometries, they can not display their advantages in this situation. Due to the coupling with the lattice-Boltzmann method, which operates on a regular cubic grid, it would be highly disadvantageous to deviate from this, since it would require additional interpolation, introducing additional error and complexity. For cubic grids and linear basis functions however, the resulting system of equations for the FEM method is the same as for finite differences. Additionally, we use the method not to calculate the electrostatic potential for a single setup but in every time step of a simulation. For a FEM simulation to actually produce the higher accuracy and convenient boundary implementation, one would have to create an optimized grid and derive the respective system of equations in every step. This overhead, that is completely absent in a finite difference method on a regular grid, would quickly make the whole simulation unfeasable.

According to the finite difference scheme, one approximates the partial derivatives of a function $f(x)$ by the finite differences between points on a regular grid with spacing h. In the simplest case of doing so, in a symmetric fashion, only taking into account nearest neighbors, this results in

$$\frac{\partial}{\partial x}f(x)\approx\frac{f(x+h/2)-f(x-h/2)}{h}\qquad\forall\, x\in\mathbb{Z}\cdot h\quad . \tag{3.16}$$

Repeatedly applying this scheme to equation (3.15) yields

$$\begin{aligned}
-4\pi\, h^2\, l_\mathrm{B}\, k_\mathrm{B}T \sum_k q_k \rho_k(x,y,z) \quad = \quad & \Phi(x+h,y,z) + \Phi(x-h,y,z) \\
+ \; & \Phi(x,y+h,z) + \Phi(x,y-h,z) \\
+ \; & \Phi(x,y,z+h) + \Phi(x,y,z-h) \\
& - 6\Phi(x,y,z)\;.
\end{aligned} \tag{3.17}$$

The right hand side of this equation represents a so-called 7-point stencil, in this case the simplest symmetric stencil representing a finite-difference approximation to the Laplacian. Additionally, we need to define boundary conditions. Since the lattice-Boltzmann implementation prescribes periodic boundary conditions, we do the same and require

$$\Phi(0,y,z) = \Phi(L_x,y,z),\ \Phi(x,0,z) = \Phi(x,L_y,z),\ \Phi(x,y,0) = \Phi(x,y,L_z), \tag{3.18}$$

$$(x,y,z) \in (\mathbb{Z}\cdot h)^3 \cap [0,L_x]\times[0,L_y]\times[0,L_z]\;. \tag{3.19}$$

To actually solve this system of linear equations, several methods were considered, including **S**uccessive **O**ver-**R**elaxation, **C**onjugate **G**radients, **M**ulti-**G**rid, and spectral methods based on Fourier transformations. Since CUDA is a relatively new framework, very few high level scientific libraries are available for it. Even more limited is the number of those which are compatible with ESPResSo's open source license. What does exist are a set of lower level libraries included in the CUDA toolkit, implementing **F**ast **F**ourier **T**ransformations and **B**asic **L**inear **A**lgebra **S**ubprograms optimized for fully populated and sparse matrices [25]. The possibility of an MG implementation was eliminated due to the relatively high complexity of the implementation, that can not be justified by the limited potential for speedup of the overall algorithm. This is because at typical system sizes, the lattice-Boltzmann and link-flux implementations take up most of the calculation time.

At the time of this writing, there existed one CUDA based open source implementation of an iterative solver for sparse systems of linear equations, so an implementation of the electrostatics solver based on this CG method and an implementation of a spectral method based on the discrete FFT included in the CUDA toolkit were chosen for evaluation. Due to drawbacks described in detail in section 4.4, the FFT solver was chosen for the final system and its function will be derived in this section.

Let $L_x = h\cdot N_x$, $L_y = h\cdot N_y$, and $L_z = h\cdot N_z$ be the side lengths of our grid with grid spacing h and let $\vec{r} = h\cdot(x,y,z)$ denote a grid point identified by the indices x, y, and z. Then the unitary and discrete Fourier transform $\hat{f}(\vec{k})$ of a function $f(\vec{r})$ defined on said grid is given by

$$\hat{f}(\vec{k}) = \frac{1}{\sqrt{N_x N_y N_z}} \sum_{x=0}^{N_x-1} \sum_{y=0}^{N_y-1} \sum_{z=0}^{N_z-1} e^{-2\pi i \left(\frac{k_x x}{N_x} + \frac{k_y y}{N_y} + \frac{k_z z}{N_z} \right)} f(\vec{r}) \; , \tag{3.20}$$

where $\vec{k} = (k_x, k_y, k_z)$ is defined on a grid with side lengths N_x, N_y, and N_z. The back transform is given by

$$f(\vec{r}) = \frac{1}{\sqrt{N_x N_y N_z}} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} \sum_{k_z=0}^{N_z-1} e^{2\pi i \left(\frac{k_x x}{N_x} + \frac{k_y y}{N_y} + \frac{k_z z}{N_z} \right)} \hat{f}(\vec{k}) \; . \tag{3.21}$$

Due to the periodicity of the complex exponential, the functions are instrinsically periodic, if considered on a domain that extends beyond the given grids. Also, if the function is real valued in real space $f(x) \in \mathbb{R}$, its Fourier transform exhibits a Hermitian symmetry

$$\hat{f}(N_x - k_x, N_x - k_x, N_x - k_x) = \hat{f}(k_x, k_x, k_x)^* \; , \tag{3.22}$$

reducing the computational effort to calculate it by a factor of two.

As every FD stencil, the stencil used here as defined in equation (3.17) consists of a linear combination of the given function, shifted by different multiples of the lattice spacing. A shift in real space corresponds to a multiplication with a scalar in Fourier space, as demonstrated here for a shift $(a, b, c) \in \mathbb{Z}^3$

$$\begin{aligned} &f(x+a, y+b, z+c) \\ &\quad = \frac{1}{\sqrt{N_x N_y N_z}} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} \sum_{k_z=0}^{N_z-1} e^{2\pi i \left(\frac{k_x (x+a)}{N_x} + \frac{k_y (y+b)}{N_y} + \frac{k_z (z+c)}{N_z} \right)} \hat{f}(\vec{k}) \\ &\quad = \frac{1}{\sqrt{N_x N_y N_z}} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} \sum_{k_z=0}^{N_z-1} e^{2\pi i \left(\frac{k_x x}{N_x} + \frac{k_y y}{N_y} + \frac{k_z z}{N_z} \right)} \left[e^{2\pi i \left(\frac{k_x a}{N_x} + \frac{k_y b}{N_y} + \frac{k_z c}{N_z} \right)} \hat{f}(\vec{k}) \right] \; . \end{aligned} \tag{3.23}$$

Applying this lemma to the individual terms of the finite difference stencil in equation

(3.17) and expressing the charge distribution by its Fourier transform as well yields

$$\sum_{k_x=0}^{N_x-1}\sum_{k_y=0}^{N_y-1}\sum_{k_z=0}^{N_z-1} e^{2\pi i\left(\frac{k_x x}{N_x}+\frac{k_y y}{N_y}+\frac{k_z z}{N_z}\right)}\left[-4\pi\, h^2\, l_{\mathrm{B}}\, k_{\mathrm{B}}T\,\hat{\rho}(\vec{k})\right] = \sum_{k_x=0}^{N_x-1}\sum_{k_y=0}^{N_y-1}\sum_{k_z=0}^{N_z-1} e^{2\pi i\left(\frac{k_x x}{N_x}+\frac{k_y y}{N_y}+\frac{k_z z}{N_z}\right)} \left[\left\{e^{2\pi i\left(\frac{k_x}{N_x}\right)}+e^{-2\pi i\left(\frac{k_x}{N_x}\right)}+e^{2\pi i\left(\frac{k_y}{N_y}\right)}+e^{-2\pi i\left(\frac{k_y}{N_y}\right)}+e^{2\pi i\left(\frac{k_z}{N_z}\right)}+e^{-2\pi i\left(\frac{k_z}{N_z}\right)}-6\right\}\hat{\Phi}(\vec{k})\right], \tag{3.24}$$

with the charge density $\rho = \sum_k q_k \rho_k$ and its Fourier transform $\hat{\rho}$. Since the discrete Fourier transformation is a base transformation with a orthonormal base, relation (3.24) has to be fulfilled for each $\vec{k}$ individually. This way, we gain knowledge about the decoupled system of linear equations in Fourier space, which are, using the relation $e^{ix}+e^{-ix} = 2\,cos(x)$

$$\hat{\rho}(\vec{k}) = \frac{\cos\left(\frac{2\pi k_x}{N_x}\right)+\cos\left(\frac{2\pi k_y}{N_y}\right)+\cos\left(\frac{2\pi k_z}{N_z}\right)-3}{-2\pi\, h^2\, l_{\mathrm{B}}\, k_{\mathrm{B}}T}\,\hat{\Phi}(\vec{k})\,. \tag{3.25}$$

Several observations can be made at this point. Firstly, if $\hat{\rho}(0,0,0) \neq 0$, the potential is not defined, more specifically, it becomes infinite. Since

$$\hat{\rho}(0,0,0) = \frac{1}{\sqrt{N_x N_y N_z}}\sum_{k_x=0}^{N_x-1}\sum_{k_y=0}^{N_y-1}\sum_{k_z=0}^{N_z-1}\rho(\vec{r}) \tag{3.26}$$

is proportional to the average charge density, this case corresponds to system containing a net charge, for which the electrostatic potential in periodic boundary conditions should indeed diverge. One way to circumvent this effect is to enforce $\hat{\rho}(0,0,0) = 0$, which is equivalent to adding a homogeneous background charge density to neutralize the system.

Secondly, expanding (3.24) to second order around $h = 0$ while setting $n_x = N_y = N_z = L/h$ and $\vec{k}_{\text{cont.}} = (k_x,k_y,k_z)/L$ yields the correct expression for the corresponding continuum problem in Fourier space

$$\hat{\rho}(\vec{k}_{\text{cont.}}) = \frac{\pi\vec{k}^2_{\text{cont.}}}{l_{\mathrm{B}}\,k_{\mathrm{B}}T}\,\hat{\Phi}(\vec{k}_{\text{cont.}})\,. \tag{3.27}$$

Using equation (3.24), we can construct a direct method for solving Poisson's equation in the given geometry, which consists of applying a discrete Fourier transformation to the charge density $\rho(\vec{r})$, yielding its representation in Fourier space $\hat{\rho}(\vec{k})$, then using (3.24) to gain the Fourier transform of the electrostatic potential $\hat{\Phi}(\vec{k})$

$$\hat{\Phi}(\vec{k}) = \frac{-2\pi\, h^2\, l_{\mathrm{B}}\, k_{\mathrm{B}}T}{\cos\left(\frac{2\pi k_x}{N_x}\right) + \cos\left(\frac{2\pi k_y}{N_y}\right) + \cos\left(\frac{2\pi k_z}{N_z}\right) - 3}\, \hat{\rho}(\vec{k}),\ \vec{k} \neq 0 \tag{3.28}$$

$$\hat{\Phi}(0) = 0\,, \tag{3.29}$$

and subsequent back transformation of $\hat{\Phi}(\vec{k})$ to gain the desired real space representation of the electrostatic Potential $\Phi(\vec{r})$.

A naive implementation of the Fourier transformation, based on the definitions (3.20) and (3.21) would scale like $O(n^2)$ where n is the number of grid points. For this particular transform, however, there are a number of algorithms that exhibit $O(n \cdot \log n)$ scaling with a small prefactor, called **F**ast **F**ourier **T**ransform algorithms. They essentially all take advantage of the fact that the transform of a function defined on a grid can be reconstructed, with linear complexity, from transforms of the function on subgrids consisting of every i$^{\text{th}}$ element, with i defined per direction. This division is repeated for the sub grids as well, until the transform is small enough, so that it can be computed with negligible effort. The depth of this recursion is $\log n$, therefore the overall scaling is $O(n \cdot \log n)$. The details depend somewhat on the specific implementation.

3.4 Summary of the Electrostatics Method

The electrostatic potential Φ is defined by the charge density ρ through Poisson's equation

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial y^2}\right) \Phi(x, y, z) = -4\pi\, l_\mathrm{B}\, k_\mathrm{B} T \sum_k q_k \rho_k(x, y, z) \ . \tag{3.15}$$

To avoid additional interpolation, the method uses a grid of the same regular cubic geometry as the lattice-Boltzmann method and approximates the Laplacian using a finite-difference approximation with a 7-point stencil. The resulting system of linear equations is solved using a Fourier transformation based approach, taking advantake of the fact that systems of linear equations resulting from finite-difference formulations decouple in Fourier space, consisting of the following steps:

1. Calculate the Fourier space representation of the $\hat{\rho}(\vec{k})$ charge density $\rho(\vec{r})$ using an FFT algorithm.
2. Divide this by a factor only depending ok $\vec{k}$,

   $$\hat{\Phi}(\vec{k}) = \frac{-2\pi\, h^2\, l_\mathrm{B}\, k_\mathrm{B} T}{\cos\left(\frac{2\pi k_x}{N_x}\right) + \cos\left(\frac{2\pi k_y}{N_y}\right) + \cos\left(\frac{2\pi k_z}{N_z}\right) - 3}\, \hat{\rho}(\vec{k}) \ , \tag{3.28}$$

   which represents the finite-difference approximation to the Laplacian using a 7-point stencil (c.f. previous section 3.3 for the derivation). This yields the Fourier space representation of the electrostatic potential $\hat{\phi}(\vec{k})$. This term is not defined for $\vec{k} = 0$. To fix this, we set

   $$\hat{\Phi}(0) = 0 \ , \tag{3.29}$$

   which corresponds to adding a homogeneous neutralizing background charge in real space. Charge neutrality is a neccessary condition for there to be a non-diverging solution.
3. Calculate the electrostatic potential in real space $\Phi(\vec{r})$ by applying an FFT back transform to $\hat{\Phi}(\vec{k})$.

This method has several advantages, which are:

- It is a direct method and therefore has no convergence problems. It also is stable under limited numerical precision, which can be an issue for other methods such as the CG method, especially when working on architectures with only single precision as in this application.

- The scaling of $O(n \cdot \log n)$ is superior of that of SOR with $O(n^{1+1/3} \cdot \log n)$ and CG with $O(n^{1+1/6} \cdot \log n)$ [15]. For the typical system sizes in this application, the $\log n$ contribution is neglibly small and due to the small prefactor, the method even outperforms other, linearily scaling ones such as MG and fast multipole methods, at those small grid sizes.
- It is very simple to implement and only relies on an FFT implementation, which is readily available for virtually all platforms.
- Given that a parallelized FFT implementation is available, parallelization of the method is trivial, since it only involves the point wise multiplication of two scalar fields.

Those advantages do come at a cost, though. The applicability is limited in the sense that

- The FFT based method to solve the finite-difference system only works if the original PDE is linear with constant coefficients. In our application, linearity is a given, but the need for constant coefficients restricts us to systems consisting of a medium with spacially invariant electrical permittivity. While the permittivity does not appear explicitly in equation (3.15), it is contained within the Bjerrum-length l_{B}.
- The boundary conditions are limited to a few very simple cases. In this method as with most Fourier transformation based methods, boundary conditions are implicitly accounted for through the choice of base functions in the transformation. The exponentials used here implement periodic boundaries. Other possibilities are Dirichlet and Neumann boundary conditions using sine and cosine transformations. Boundary conditions defined on any geometry other than two opposite sides of a cuboid are impossible.

Both of those limitations can be resolved using an iterative solver instead of the FFT based one to solve the system of equations resulting from the finite-difference discretization. For comparison, one such method is discussed in section 4.4.

3.5 Link-Flux Diffusion-Advection

As described in section 2, the propagation of the particle densities ρ_k is determined by the continuity equation

$$\frac{\partial \rho_k}{\partial t} = -\nabla \cdot \vec{j}_k \, , \tag{2.1}$$

with the particle density fluxes given by the diffusion-advection equation

$$\vec{j}_k = \underbrace{-D_k \nabla \rho_k - \nu_k \, q_k \rho_k \, \nabla \Phi}_{\text{diffusive contribution}} \; + \; \underbrace{\rho_\mathrm{k} \vec{v}}_{\text{advective contribution}} \, , \tag{2.6}$$

depending on the electrostatic potential Φ and the advective velocity $\vec{v}$, determined by Poisson's equation (2.8) and the Navier-Stokes equations (2.17). The methods discussed in the following are simply those proposed in the original publication introducing this method to solve the electrokinetic equations [4].

In order to discretize these equations using the same regular cubic grid as for the methods used to determine the electrostatic potential Φ and the advective velocity $\vec{v}$ (as introduced in section 3.1 and 3.3), we employ two different schemes for the diffusive contribution and advective contribution to the particle flux, resprectively. As demonstrated by Wagner [34], the choice of compatible discretization schemes for diffusive and advective dynamics, in applications combining the two, is crucial for the supression of spurious fluxes. In a physical system in equilibrium, both the advective velocity and all fluxes must vanish. Numerical schemes with incompatibly chosen discretizations, however, often exhibit stationary states in which the diffusive and advective fluxes don't vanish individually, but cancel each other. These remaining fluxes are the so-called spurious fluxes. Capuani et. al. show that for the scheme as will be presented in the following, spurious fluxes only exist on the order of the machine precision.

If you consider a lattice node to represent the cubic volume reaching half way to its neighbors and further assume that the particle density is constant within one such cell, then a discrete equivalent of the continuity equation (2.1) can be deduced by means of Gauss' theorem

$$\frac{\partial}{\partial t} \int_V \rho_k \, dV = - \int_{\partial V} \langle \vec{j}_k, \hat{\vec{n}} \rangle \, dA \, , \tag{3.30}$$

with V the aforementioned volume and its surface ∂V, and $\hat{\vec{n}}$ a unit vector normal to said surface pointing outwards. In order to avoid anisotropies in the transient dynamics, stemming from the geometry of the underlying lattice, we have to take at least nearest and next nearest neighbors into account. Using an explicit first order time stepping scheme, this yields

$$n_k(\vec{r}, t + \Delta t) = n_k(\vec{r}, t) - \Delta t \cdot A \sum_i j_{ki}(\vec{r}, t) \; , \tag{3.31}$$

where $n_k(\vec{r}, t)$ is the number of particles of species k on the lattice node at position $\vec{r}$ and time t. $j_{ki}(\vec{r})$ denotes the flux of particles of species k from the node at location $\vec{r}$ to its neighbor at $\vec{r} + \vec{c}_i$ at time t. For the sake of simplicity, we choose the area A, which one of the discrete fluxes covers, to be the same for all fluxes of a node.

A discretized expression for the diffusive part of the flux can be obtained by using finite-differences to evaluate the gradients on the grid

$$j_{ki}(\vec{r}, t) = D_k \cdot \frac{n_k(\vec{r}, t) - n_k(\vec{r} + \vec{c}_i, t)}{|\vec{c}_i|} + \nu_k \, q_k \cdot \frac{n_k(\vec{r} + \vec{c}_i, t) + n_k(\vec{r}, t)}{2} \cdot \frac{\Phi(\vec{r} + \vec{c}_i) - \Phi(\vec{r})}{|\vec{c}_i|} \tag{3.32}$$

Considering the second moment of a particle distribution initially located at a node $\vec{r}$, after one time step, and requiring it to match the correct continuum value

$$\sum_{\Delta\vec{r}} (\Delta\vec{r})^2 \, n(\vec{r} + \Delta\vec{r}, \Delta t) = 2 \, d \, D \, \Delta t \; , \tag{3.33}$$

where D is the diffusion constant and d the dimensionality, yields a value for for the area covered by one of the discrete fluxes, of $A = 1 + 2\sqrt{2}$.

Capuani et. al. propose a equivalent formulation of the diffusive contribution in the flux term (2.6) derived in section 2.1, of the shape

$$\vec{j}_k = -D_k \, e^{-\frac{q_k \Phi}{k_\mathrm{B} T}} \, \nabla \left(\rho_k \, e^{\frac{q_k \Phi}{k_\mathrm{B} T}} \right) \; , \tag{3.34}$$

which results in a discretized expression for the fluxes between nodes

$$j_{ki}(\vec{r}, t) = D_k \cdot \frac{e^{-\frac{q_k \Phi(\vec{r})}{k_\mathrm{B} T}} + e^{-\frac{q_k \Phi(\vec{r} + \vec{c}_i)}{k_\mathrm{B} T}}}{2} \cdot \frac{n_k(\vec{r}, t) \, e^{\frac{q_k \Phi(\vec{r})}{k_\mathrm{B} T}} - n_k(\vec{r} + \vec{c}_i, t) \, e^{\frac{q_k \Phi(\vec{r} + \vec{c}_i)}{k_\mathrm{B} T}}}{|\vec{c}_i|} \; , \tag{3.35}$$

which has the advantage that it becomes exactly zero in equilibrium, helping to eliminate spurious fluxes. What has not been reported by Capuani et. al. is, that this formulation is only strictly equivalent to the original one in the continuum limit, and that it introduces higher order contributions in the discrete case. For simplicity, consider the flux of a homogeneously distributed species

$$j_i = \frac{Dn}{2\,|\vec{c}_i|} \cdot \left(e^{-\frac{q\Phi(\vec{r})}{k_\mathrm{B}T}} + e^{-\frac{q\Phi(\vec{r}+\vec{c}_i)}{k_\mathrm{B}T}} \right) \left(e^{\frac{q\Phi(\vec{r})}{k_\mathrm{B}T}} - e^{\frac{q\Phi(\vec{r}+\vec{c}_i)}{k_\mathrm{B}T}} \right) . \tag{3.36}$$

By setting $\Phi(\vec{r}+\vec{c}_i) = \Phi(\vec{r}) + \delta_{\vec{c}_i}\Phi(\vec{r})$, this can be expressed as

$$j_i = -\frac{Dn}{|\vec{c}_i|} \cdot \sinh\left(\frac{q\delta_{\vec{c}_i}\Phi(\vec{r})}{k_\mathrm{B}T} \right) , \tag{3.37}$$

from which one recovers the correct expression by first order expansion of the hyperbolic sine. As demonstrated in section 5.3, this can lead to noticable differences in nonequilibrium situations.

The advective flux is determined using a scheme that is consistent with the assumption that the density of particles is homogeneous in the volume represented by a grid node. The advective velocity then displaces this volume and the fluxes have to be chosen such, that the the number of particles occupying the parts of the displaced volume that now intersects with the neighboring grid cells, are transferred to the respective neighboring cell. Figure 3.3 illustrates this scheme in two dimensions and gives the respective intersecting volume ratios.

Boundary conditions can be implemented by requiring fluxes connecting boundary nodes to be set to a prescribed value, in application most often zero.

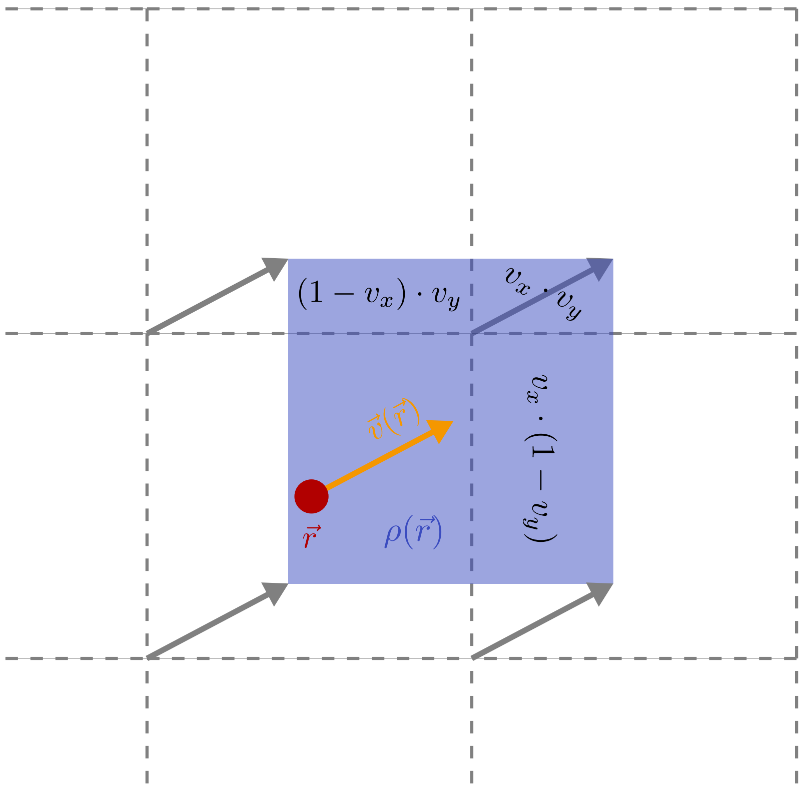

Figure 3.3: Two dimensional illustration of the scheme used to calculate the advective fluxes. In its discretized shape, the flux to the neighbor on the right for example, would be given by $\rho(\vec{r}) \cdot v_x \cdot (1 - v_y) \cdot \frac{h^2}{\Delta t}$, where h is the lattice spacing and Δt is the time step. Dashed lines depict the boundaries between the volumina associated with individual lattice nodes.

4 Implementation

At the time of this writing, considerable effort has been undertaken to port the computationally more expensive parts of ESPResSo to GPUs – a trend that started with Dominic Röhm's implementation of the lattice-Boltzmann method, using NVIDIA's CUDA framework [28, 29]. This implementation, with its 50-fold speedup over an equivalent implementation for CPUs using MPI for parallelization, showed not only that lattice based algorithms with local interactions are perfect candidates for GPU based implementations, but also that the development effort is significantly reduced, compared to CPU implementations using MPI. This is due to the overhead of complex and fault-prone communication schemes, which need to be employed in distributed memory architectures present in all modern super computers.

In addition, more and more computational power is available in the shape of clusters of GPUs, a trend that will most certainly continue, considering the lower initial costs and energy consumption per FLOP for GPUs.

For those reasons, this implementation of the electrokinetics solver is based on NVIDIAs CUDA framework. The following sections will introduce the relevant CUDA features and discuss the special considerations neccessary to develop efficient CUDA programs.

4.1 The CUDA Archtitecture

The NVIDIA **C**ompute **U**nified **D**evice **A**rchitecture [25] is based on an accelerator model. This means that programs taking advantage of the computational power of a CUDA capable **G**raphics **P**rocessing **U**nit are still executed by the CPU of the host computer, but offload certain computationally intensive tasks to the GPU. The transfer of the input data to the GPU and the subsequent transfer of the results back from the GPU happens over the PCIe (**P**eripheral **C**omponent **I**nterconnect **E**xpress) bus.

Internally, the hardware architecture of a CUDA capable device adheres to the **S**ingle **I**nstruction **M**ultiple **D**ata model, which means that the device executes the same instructions on a large number of computation cores, each working with another set of data. This makes the GPU a vector processor, as opposed the CPU, which is a scalar processor, only executing one instruction with a single set of data at a time[1].

The main difference between the core of a CUDA device and the core of a multicore CPU is that a big number of CUDA cores share their control structures, whereas each core from a multicore CPU has its own. This enables the CPU to run instructions from different programs in parallel, while the GPU is restricted to running the same instructions on all cores of a so-called **S**treaming **M**ulti**P**rocessor. The actual number of cores per SMX and the number of SMXs in a CUDA device varies between hardware versions. At the time of this writing, a cutting-edge CUDA capable GPU, such as the GeForce GTX 680 contains 8 SMXs with 192 cores each, which makes a total of 1536 cores.

Up till recently, a severe limitation for GPU computing was the small amount of memory available on GPUs. Recent CUDA capable devices, however, are available with up to 6 GB of memory. For a typical GPU computing application, the bottlenecks have become the transfer of data to and from the device and, during the computation, the limited memory bandwith and latency.

To mitigate the effects of the limited data transfer rate on the PCIe bus, NVIDIA introduced the so-called streamed processing, in which several tasks, memory transfers and calculations, can be arranged in streams for subsequent execution. By having several streams processed asynchronously with the execution of the host program, data transfers in one stream can be overlapped with calculations from another stream and calculations on the CPU.

Similarily, latencies of the GPUs memory can be hidden by spawning a number of threads, much bigger than the number of cores, so that an SMX can execute another set of threads while the first one is waiting for data from, or to the so-called global memory. The CUDA hardware is specifically optimized to execute the neccessary content switches with only minimal overhead. For read accesses, the steadily increasing amount of cache in an SMX has alleviated the problem significantly. Additionally, CUDA allows the developer to control the cache contents manually by allowing them to allocate parts of it as so-called shared memory.

One problem, which occurs with every parallel computation architecture is the existence of race conditions that occur in many applications. These conditions exist because a memory read and subsequent write can not be executed within one cycle. Whenever two threads try to read, modify, and write using the same data in memory, the modifications of the first thread's write, will be overwritten by the second, and therefore lost.

[1] Many CPUs implement technologies, which give them vector processing capabilities as well, but compared to GPUs, these are very limited.

The typical solution for this problem is a lock architecture that allows to read and lock a memory segment from accesses until the respective write occurs. CUDA implements this feature in a number of so-called atomic operations.

4.2 The CUDA Programming Model

The CUDA toolkit includes the `nvcc` compiler, which processes C- and C++-code with a number of CUDA specific extensions. The following example shows parts of a bare bone implementation of the FFT based finite-difference electrostatics solver introduced in section 3.3. The scheme consists of only a Fourier transformation, a multiplication in Fourier space, and a Fourier back transformation. Its simplicity makes it a prime example of an almost minimal CUDA accelerated program.

```
#define PI_FLOAT 3.14159265358979323846264338327f
```

Since some older CUDA capable devices can not process double precision floating point numbers, one has to make sure to define all floating point numerical constants explicitly as floats. Otherwise it can happen that the compiler promotes intermittent results in calculations to type double, which causes the compilation for those devices to fail.

```
__global__ void createGreensFunc(cufftReal* greensfunc, ↩
    unsigned int Nx, unsigned int Ny, unsigned int Nz, ↩
    float h) {
  for(int i = blockDim.x*blockIdx.x+threadIdx.x; i < Nz ↩
      * Ny * (Nx/2+1); i += gridDim.x*blockDim.x) {
    if(i == 0)
      greensfunc[i] = 0.0f;
    else
    /* ... */
  }
}
```

This function precomputes the Green's function for the whole grid and stores it in the device memory. Due to the great computational power and the memory limitations, the tradeoff between computations and memory is usually much more in favour of computations, in this case the evaluation of several trigonometric functions makes the solution using precomputation faster.

The keyword `__global__` defines a function to be called from the CPU but executed on the GPU. Such functions are referred to as kernels and they always need to be of type `void`. The global variables `threadIdx`, `blockIdx`, and `blockDim` identify a specific thread. Threads are organized in blocks, whose size is given by `blockDim`. Those blocks of threads have three dimensions but in this example, the y- and z-extension is set to one. Analogously, blocks can be arranged in a three dimensional grid, whose dimensions are given by `gridDim`.

A common method to apply an operation to a number of grid points in parallel is to construct a linear index from this information and use it to access an array, as done here with the index i. A single thread can then either loop over a portion of the array or there can be as many threads as array entries. The former allows for more architecture specific optimizations while the latter performs on average better on different configurations.

```
__global__ void multiplyGreensFunc(cufftComplex* data, cufftReal* greensfunc, unsigned int N) {
  i = blockDim.x*blockIdx.x+threadIdx.x;

  if(i < N) {
    data[i].x *= greensfunc[i];
    data[i].y *= greensfunc[i];
  }
}
```

This is an example of the second method. Since the integer grid- and block-dimensions most often do not result in the exact number of threads necessary, one always needs to check whether the calculated index lies within the desired bounds.

```
int main(int argc, char** argv) {
  /* ... */
  cudaMalloc((void**) &data_dev, sizeof(cufftComplex)*Nz*Ny*(Nx/2+1));
  cudaMallocHost((void**) &data_host, sizeof(cufftComplex)*Nz*Ny*(Nx/2+1));
```

```
    /* ... */
```

The function `cudaMalloc` allows one to allocate device memory. As opposed to its CPU equivalent, it does not return a pointer but expects to recieve a reference for it.

```
    createGreensFunc<<<14,32*32>>>(greensfunc_dev, Nx, Ny,↩
        Nz, h);
```

This demonstrates how to invoke a kernel call. The additional construct `«<gridDim,blockDim»>` specifies the number of threads and their arrangement in blocks.

```
    /* ... */
    cudaMemcpy(data_dev, data_host, sizeof(cufftComplex)*↩
       Nz*Ny*(Nx/2+1), cudaMemcpyHostToDevice);
```

This copies data from the host to the device memory. Analogously, specifying `cudaMemcpyHostToDevice` does the reverse.

```
    cufftPlan3d(&plan_fft, Nz, Ny, Nx, CUFFT_R2C);
    cufftSetCompatibilityMode(plan_fft, ↩
       CUFFT_COMPATIBILITY_NATIVE);
    cufftPlan3d(&plan_ifft, Nz, Ny, Nx, CUFFT_C2R);
    cufftSetCompatibilityMode(plan_ifft, ↩
       CUFFT_COMPATIBILITY_NATIVE);

    cufftExecR2C(plan_fft, (cufftReal*) data_dev, data_dev↩
       );
```

Detailed information about library functions included in the CUDA toolkit, like the FFTs used here, can be found in the CUDA documentation included in the toolkit or at [25].

```
    multiplyGreensFunc<<<Nz*Ny*(Nx/2+1)/(32*32)↩
       +1,32*32>>>(data_dev, greensfunc_dev, Nz*Ny*(Nx↩
       /2+1));
    /* ... */
```

In this case, the kernel will need one thread per array entry. Therefore the block- and grid-dimensions are chosen such that the number of threads is at least as big as the number of array entries. The usage of more than one block is required since CUDA limits the number of threads per block to 32^2.

This example illustrates how easily the enormous computational power of modern GPUs can be used for scientific computations using the CUDA framework. For a detailed documentation about the more advanced infrastructure using streams to asynchronously transfer data between host and device, please consult the diploma thesis of Dominic Röhm, who implemented it [28].

4.3 Electrokinetics Solver

The integration loop of the routine is structured as follows

1. Integrate diffusion-advection scheme to propagate particle densities.
   For each species:
   a) Clear the fluxes from the previous species' propagation.
   b) Calculate the fluxes for the current species.
   c) Clear all fluxes connecting boundary nodes.
   d) Propagate densities according to the calculated fluxes.
2. Calculate the electrostatic potential
3. Integrate the LB

The integration of the diffusion-advection scheme happens sequencially for each of the different species. This allows up one to only allocate memory for a single flux field, instead of one per species.

The order of integration in the outer loop has the disadvantage that the electrostatic potential needs to be calculated separately before the first integration step, but it allows the user to output the current state of the system between two integration steps, including the electrostatic field consistent with the current particle distribution.

To avoid redundancies causing unnecessary memory consumption, the boundary system present in the LB implementation is used to determine which nodes are boundaries.

Boundaries can also carry charge, a feature not previously implemented in ESPResSo (unless realized using fixed charged particles). To avoid additional memory consumption for those charges, they are realized placing densities of the first one of the charged species in the wall. The elimination of all fluxes connecting boundary nodes ensures that this boundary charge distribution stays fixed.

One more feature of the implementation is that it consistently avoids the use of conditional statements. CUDA does not prohibit their use, but since a large number of CUDA cores shares their control logic, all branches of a conditional statement, which are reached by one of those cores, need to be executed sequentially. The cores for which a particular condition does not apply are then just running empty. For most parts of the implementation, like neighbor index calculations for the diffusive fluxes, conditional statements could easily be replaced by more general mathematical formulas. For the advective fluxes this posed a more serious challenge. The solution of this problem consists of determining the direction of the advective velocity by using the sign bit of its components

```
    di[0] = 1 - signbit(dx[0]);
    di[1] = 1 - signbit(dx[1]);
    di[2] = 1 - signbit(dx[2]);
```

and the relative overlap of the displaced node volume with its neighbors

```
    dx[0] = fabs(dx[0]);
    dx[1] = fabs(dx[1]);
    dx[2] = fabs(dx[2]);
```

Mathematical expressions for all index calculations and physical quantities, related to the advective fluxes could then be determined using those values.

4.4 Conjugate-Gradient Electrostatics solver

Using the linear indexing scheme

$$i = N_y N_x z + N_x y + x \ , \tag{4.1}$$

for the three-dimensional regular cubic grid with side lengths N_x, N_y, and N_z, results in a band matrix with only 13 bands for a 7-point stencil representing the Laplacian with periodic boundary conditions.

Such a sparsely populated system of linear equations can efficiently be solved by iterative methods in conjunction with algebra routines optimized for sparse systems. Solvers based on conjugate gradients form such an approach. A good introduction to the topic was given by Shewchuk in 1994 [30]. An implementation of this method, called CUSP, based on the CUDA framework was published by Dalton and Bell [6] under an open source license. Using this library, a complete electrostatics solver can be implemented with very little effort, since only the matrix and right hand side of the system need to be assembled and passed to the solver as shown below.

```
cusp::dia_matrix<int,ValueType,DevMem> A(n*n-1, n*n-1, 5*(n*n-1)-4, ↩
    9);
cusp::array1d<ValueType,DevMem> b(n*n-1, 0.0);

A.diagonal_offsets[0] = 0; //self
A.diagonal_offsets[1] = -1; //left
A.diagonal_offsets[2] = n-1; //left_folded
A.diagonal_offsets[3] = 1; //right
A.diagonal_offsets[4] = -(n-1); //right_folded
A.diagonal_offsets[5] = -n; //up
A.diagonal_offsets[6] = (n*n - n) % (n*n); //up_folded
A.diagonal_offsets[7] = +n; //down
A.diagonal_offsets[8] = (n - n*n) % (n*n); //down_folded

for(int i = 0; i < n*n-1; i++) {
  A.values(i,0) = -4.; //self

  if(i != 0) //left
    if((i+1)%n == 0)
      A.values(i,2) = 1.;
    else
      A.values(i,1) = 1.;

  if(i != n-2) //right
    if((i+2)%n != 0) A.values(i,3) = 1.;
    else A.values(i,4) = 1.;

  if(i != n-1) //up
    if(i < n) A.values(i,6) = 1.;
```

```
      else A.values(i,5) = 1.;

    if(i != n*n-n-1) //down
      if(i < n*n-n) A.values(i,7) = 1.;
      else A.values(i,8) = 1.;
  }

  /* set right hand side b */
  cusp::array1d<ValueType,DevMem> x(n*n-1, 0.);
  cusp::default_monitor<ValueType> monitor(b, 10*n*n, 1.e-10*(n*n-1));
  cusp::krylov::cg(A, x, b, monitor);
```

This method has the advantage over the FFT based solver introduced in section 3.3, that it is capable of treating systems with spacially varying permittivity. A drawback is the higher runtime, which might however be mitigated in a molecular dynamics simulation, since the solution from the previous time step can be used as a good initial approximation for the solution during the next time step. How strong this effect is in this particular application will be subject of future investigation.

The main argument against the implementation of this method in ESPResSo is insufficient maintaining of the library. Over the course of this thesis, NVIDIA released two new versions of the CUDA toolkit, which led to incompatibilities between the toolkit and the CUSP library, which were only resolved months later. Circumstances like this are obviously not acceptable for production software like ESPResSo.

4.5 Benchmark of the FFT Electrostatics Solver

The FFT based finite-difference solver for Poisson's equation as described in 3.3 and 3.4 is a method that strongly deviates from the one suggested by Capuani et. at. [4], it could be replaced by another method with relatively little structural changes to the implementation, and it is also interesting as a standalone application. For those reasons I want to assess its performance individually. Precise values can be obtained by timing the execution of every kernel individually, using the timing routines provided by the CUDA toolkit. I did so on a machine that was not occupied with any processes that make use of the GPU. This procedure minimizes the influence of the operating systems scheduling of other processes utilizing the CPU and the thereby caused delay of kernel executions. Repeated runs show, that the timings obtained this way consistently yield the same results.

A typical timing routine is implemented in the following way

```
float time = 0.0, time_tmp;
CUDAEvent_t start, stop;

CUDAEventCreate(&start);
CUDAEventCreate(&stop);
```

```
CUDAEventRecord(start, 0);

if(cufftExecR2C(plan_fft, (cufftReal*) data_dev, ↩
    data_dev) != CUFFT_SUCCESS) {
  fprintf(stderr, "ERROR: Unable to execute FFT plan\n")↩
      ;
  return 1;
}

CUDAEventRecord(stop, 0);
CUDAEventSynchronize(stop);
CUDAEventElapsedTime(&time_tmp, start, stop);

time += time_tmp;
```

```
CUDAEventDestroy(start);
CUDAEventDestroy(stop);
```

The first and last part contain the allocation and deallocation of the variables used in the timing. The middle part times the execution of, in this case, a real-to-complex FFT. `CUDAEventRecord` markes a position in the execution of the GPU stream, to store the time by which it gets reached in the event variable given as the first parameter. The second parameter gives a stream. The standalone version of the solver uses only the default stream 0. `CUDAEventRecord` is an asynchronous operation, therefore the CPU would reach `CUDAEventElapsedTime` at a time when the neccessary information is not yet available in the events start and stop. `CUDAEventSynchronize` provides a barrier to prevent this. Finally, `CUDAEventElapsedTime` calculates the time in milliseconds between the `start` and `stop` event and stores it in `time_tmp`, which gets added to the total execution time stored in `time`. The total execution times for different grid sizes on a NVIDIA GeForce GTX 680 are given in figure 4.1.

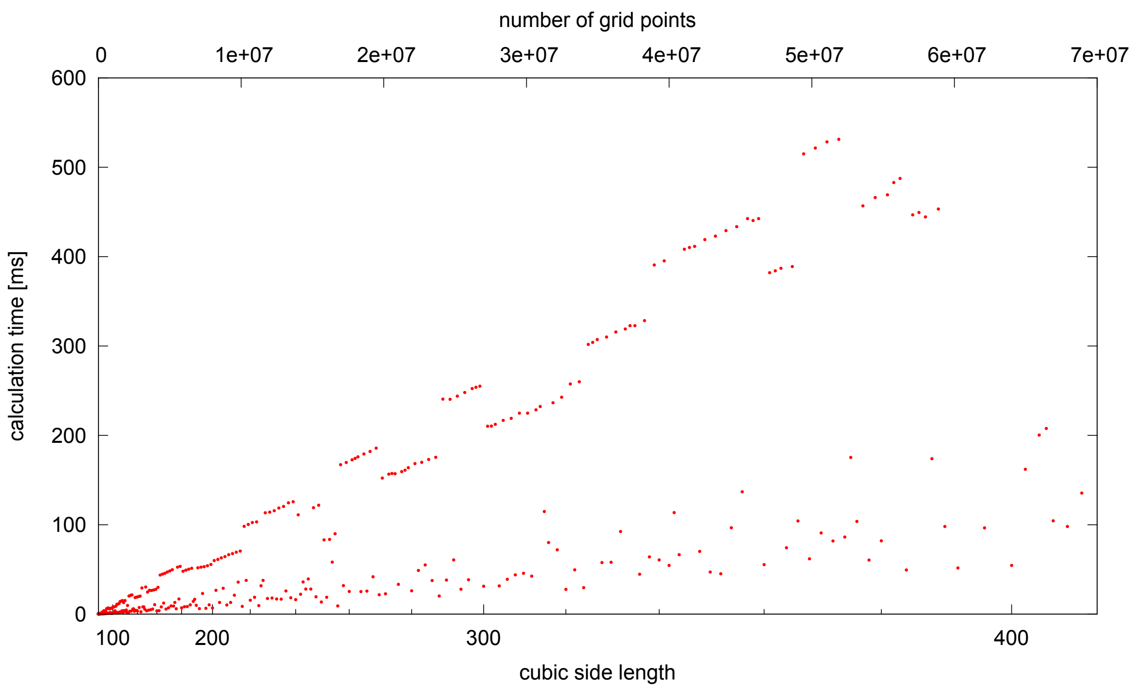

Figure 4.1: Necessary calculation time for a single run of the Poisson solver as described in section 3.3 and 3.4 on a NVIDIA GeForce GTX 680. The lower x-axis shows the side length of the cubic grid while the upper x-axis shows the total number of grid points of the cubic grid used.

The odd scaling behaviour comes from the fact that the FFT is a divide and conquer algorithm as briefly described in the end of section 3.3. The process of recursively dividing the problem can only be applied without any additional overhead for side lengths of the transformation domain, which are composed of certain prime factors. The CUDA FFT implements this for the prime factors 2, 3, 5, and 7. The run time is then mostly determined by the number of prime factors other than 2, 3, 5, or 7 in the side length of the, in this case, cubic domain. The upper envelope in this scatter plot therefore corresponds to the scaling for prime numbered side lengths, whereas the lower envelope is determined by the scaling of optimal grids with side lengths only consisting of the prime factors 2, 3, 5, and 7. For production runs, the scaling of the optimal grids is the relevant one, since for grid of side lengths between 0 and 400, the distribution of optimal grid sizes is dense enough, so that one can limit himself to using only those.

For architectural reasons, independent of the amount of memory available, the CUDA FFT is limited to transforms of sizes up to $128 \cdot 10^6$ float elements [?]. This corresponds to cubic grids with side lengths of up to 500. Since transforms with non-optimal grid sizes consume more resources, those will fail at even smaller grids.

The conclusions from these measurements are: The electrostatics solver operating according to the principles described in section 3.3 and implemented using the NVIDIA CUDA framework, can be applied to systems of sizes up to 400^3 (for specifically selected grid sizes up to 512^3) and exhibits approximately linear scaling in this range. On a current high end graphics card such as the NVIDIA GeForce GTX 680, it achieves calculation times of under $50\,\mathrm{ms}$ even for the biggest of those grid sizes, if the side lengths are composed of prime factors 2, 3, 5, and 7, exclusively.

5 Verification

5.1 Poisson-Boltzmann Theory

As discussed in section 2.1, Poisson-Boltzmann theory can be recovered as the stationary state solution for the electrokinetic equations with vanishing advective velocity. In the current implementation of the electrostatics solver, this can easily be achieved by leaving out the integration step for the lattice-Boltzmann solver. To verify the correctness of the implementation of the link-flux diffusion solver as discussed in section 3.5, and the FFT based finite-difference solver for the electrostatic interactions as described in section 3.3 and 3.4, we simulate two systems which can be treated analytically.

The first one of those systems consists of two like charged parallel plates, confining their counterions. 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, and thus having periodic or non-periodic boundary conditions makes no difference in this case. As discussed in detail in a previous investigation of this system [27], the Poisson-Boltzmann equation for the elecrostatic potential Φ reduces to an ordinary differential equation of the shape

$$\frac{\partial_x^2 \Phi(x)}{\partial x^2} = -4\pi\, k_\mathrm{B}T\, l_\mathrm{B}\, q\, \rho_0 \cdot e^{-\frac{q\Phi(x)}{k_B T}} \;, \tag{5.1}$$

where x denotes the direction normal to the plates and ρ_0 an arbitrary constant, that has to be chosen such that charge neutrality is fulfilled. The remaining variables are the same as those defined in section 2. Multiplying with $2\frac{\partial\Phi(x)}{\partial x}$ and applying the inverse chain rule further reduces the equation to first order. Subsequent separation of variables yields the solution

$$\Phi(x) = -\frac{k_\mathrm{B}T}{q} \cdot \log\left[\frac{C^2}{8\pi\, k_\mathrm{B}T\, l_\mathrm{B}} \cdot \cos^{-2}\left(\frac{qC}{2k_\mathrm{B}T} \cdot x\right)\right] , \quad \left|\frac{qC}{2k_\mathrm{B}T} \cdot x\right| < \frac{\pi}{2} \;. \tag{5.2}$$

Knowing that the density ρ resembles a Boltzmann distribution in the potential $q\Phi$ leads to the desired result for the counterion density

$$\rho(x) = \frac{C^2}{8\pi\, k_{\mathrm{B}}T\, l_{\mathrm{B}}} \cdot \cos^{-2}\left(\frac{qC}{2k_{\mathrm{B}}T} \cdot x\right) \ . \tag{5.3}$$

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

$$C \cdot \tan\left(\frac{qd}{4k_{\mathrm{B}}T} \cdot C\right) = -4\pi\, k_{\mathrm{B}}T\, l_{\mathrm{B}}\sigma \ , \tag{5.4}$$

where d is the distance between the plates and σ the constant surface charge density on the plates.

Simulation results of a system consisting of two parallel plates carrying a surface charge density of $\sigma = -0.1$ at a distance of $d = 50$ lattice spacings and a single species of monovalent ions neutralizing the wall charge are depicted in figure 5.1. For convenience the thermal enery $k_{\mathrm{B}}T$, as well as the Bjerrum-length l_{B} were chosen to be unity in simulation units. For information about how to map these units to the SI or any other real world unit system, consult [27, sec. 4.1]. The stationary state is reached after roughly $t = 30$ simulation time units exhibits perfect agreement with the analytical prediction based on Poisson-Boltzmann theory.

The diffusion constant D has no influence on the stationary state distribution but determines the speed at which this distribution is reached. In this simulation it was chosen to be $D = 3.5$ and close to the upper stable limit of the link-flux algorithm, to speed up the convergence process. For the time step used, in this case $\Delta t = 0.1$, and lattice spacing $h = 1$, the upper stable limit for the diffusion constant is $D_{\mathsf{max}} = 4.4$.

The variation in the electrostatic potential across the channel drops from $\delta\Phi|_{t=0} \geq 16.3$ for the initial homogeneous charge distribution to $\delta\Phi|_{t\geq 30} \geq 3.8$ for the stationary distribtuion. This puts the system well within the nonlinear regime where the Debye-Hückel approximation, linearizing the exponential in the Poisson-Boltzmann equation 5.2, would lead to a significant underestimation of the charge density close to the walls.

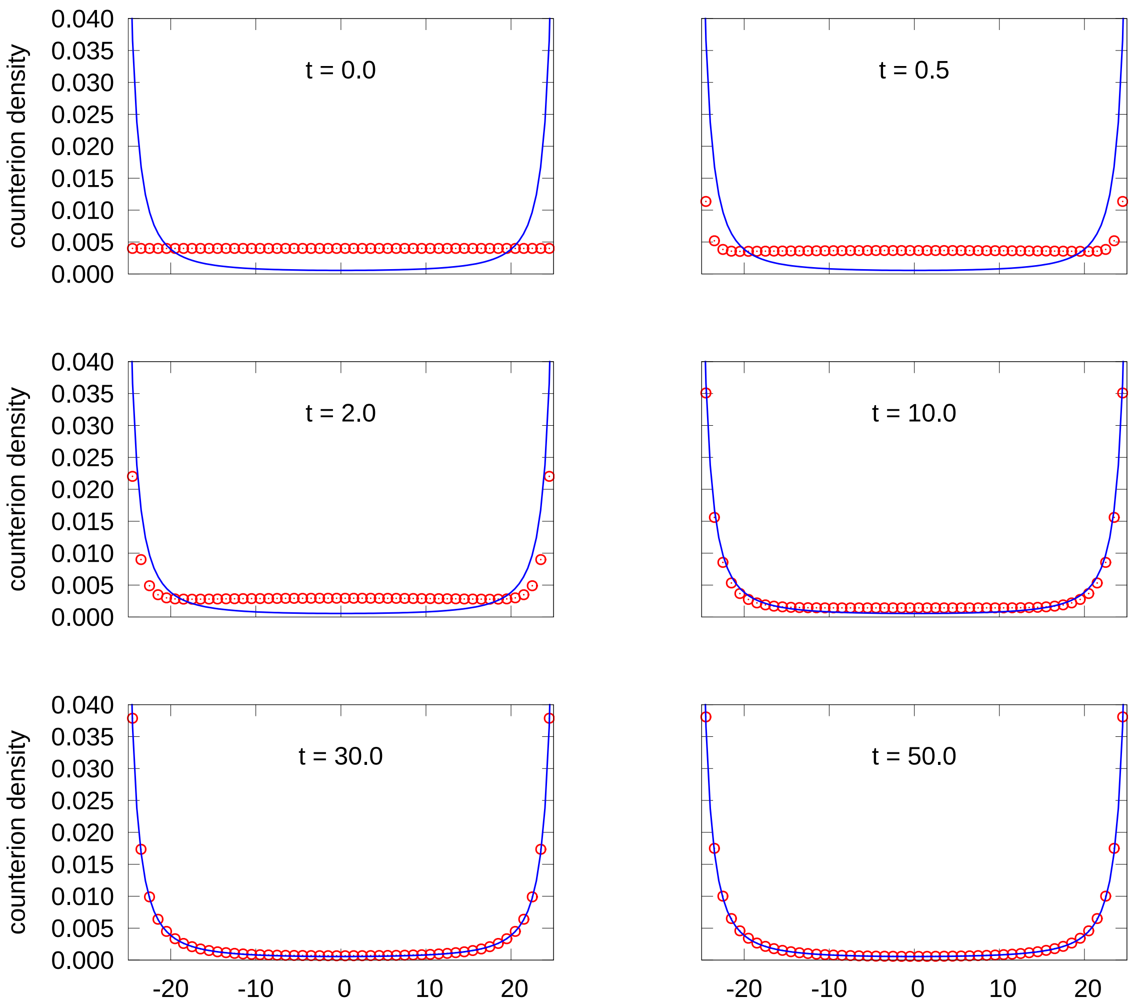

Figure 5.1: The density of a single species of monovalent ions contained in a channel consisting of two parallel flat walls and periodic boundary conditions in the directions parallel to the walls (*red circles*). The successive graphs show the initial homogeneous distribution of counterions, the transient distributions at different times, and finally the stationary state distribution agreeing with the corresponding result from Poisson-Boltzmann theory (*blue line*).

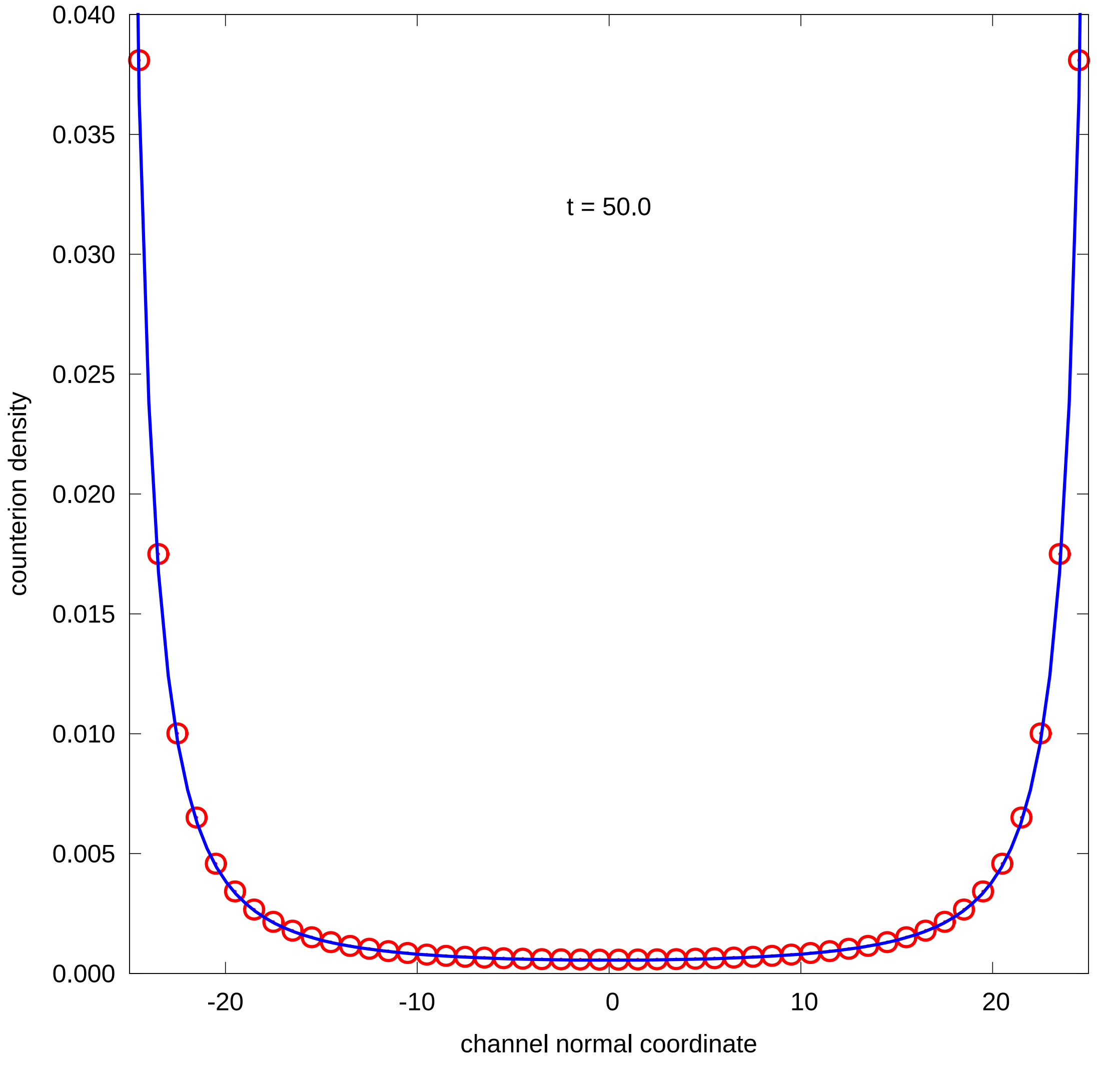

Figure 5.2: Magnification of the last frame from figure 5.1, depicting the stationary state density of the system consisting of homogeneously charged parallel flat walls, containing their counterions. Simulation results are shown as *red circles*, the prediction from analytical Poisson-Boltzmann theory as the *blue line*.

5.2 Debye-Hückel Theory

A second test case consists of the same geometry, but this time the walls are oppositely charged and the surface charge density $\sigma = \pm 0.001$ is small enough, so that the Debye-Hückel approximation is valid. Contained in between the walls are two species of monovalent oppositely charged ions at equal average densities $\bar{\rho}_{1/2} = \rho_0/2 = 0.05$. The Poisson-Boltzmann equation for this case reads as

$$\frac{\partial_x^2 \Phi(x)}{\partial x^2} = -4\pi \, k_\mathrm{B} T \, l_\mathrm{B} \, q \, \rho_0 \cdot \sinh\left(-\frac{q\Phi(x)}{k_B T}\right) \;, \tag{5.5}$$

where the potential at the wall $\Phi(0) = \zeta$ is called zeta-potential and one requires the potential $\Phi(x \to \infty) = 0$ and its derivative $E = \frac{\partial \Phi(x\to\infty)}{\partial x} = 0$, the E-field, to vanish at a distance infinitely far from the wall. A procedure analogous to the previous case then yields an implicit solution for the potential

$$\int_\zeta^{\Phi(x)} \left[\cosh\left(\frac{u}{k_\mathrm{B}T}\right)\right]^{-\frac{1}{2}} du = -\mathrm{sgn}(\zeta) \cdot x \;. \tag{5.6}$$

In the case of $\zeta \ll 1$, however, one can linearize the hyperbolic sine in eq. (5.5) and find an explicit solution to the much simpler linear equation. This approach leads to an exponential decay of the potential

$$\Phi(x) = \zeta \cdot e^{-x/\lambda_\mathsf{D}} \;, \tag{5.7}$$

with the characteristic length $\lambda_\mathsf{D} = \left(4\pi \sum_i q_i^2 \bar{\rho}_i\right)^{-\frac{1}{2}}$, the so-called Debye screening length. The parameters in the simulation result in a Debye-length of $\lambda_\mathsf{D} = 2.86$ and a zeta-potential at the walls of $\zeta = \pm 0.026$. Since the walls are separated by more than 17 Debye lengths, there is no measurable difference between a bulk electrolyte infinitely far away from the walls, and the potential in the center of the channel. The potential in the simulation will therefore resemble a superposition of the potential (5.7) for each of the two walls.

Figure 5.3 shows that in this case the simulation results coincide precisely with the Debye-Hückel approximation.

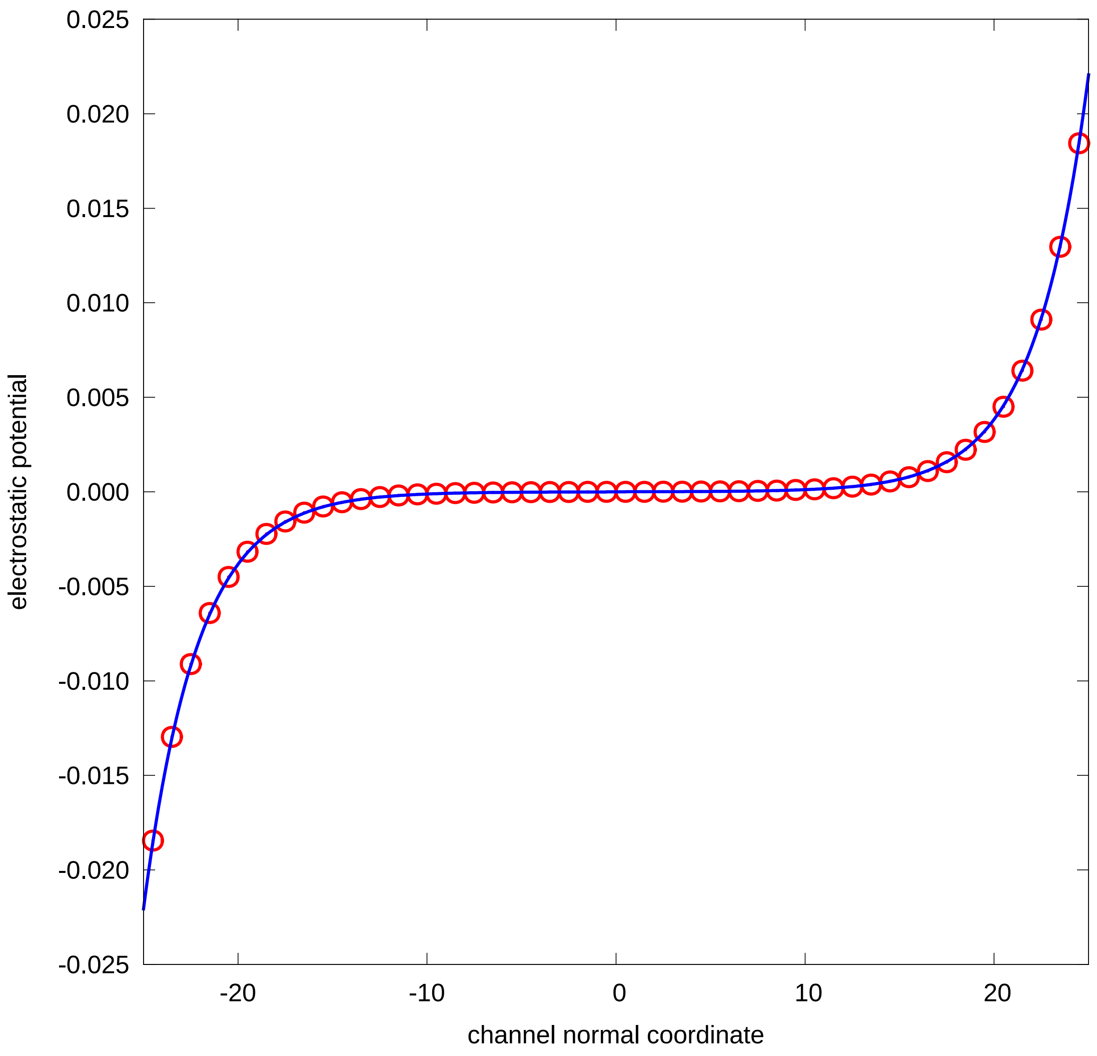

Figure 5.3: Electrostatic potential caused by the stationary state distribution of two oppositely charged monovalent particle species, contained between oppositely charged parallel plates (*red circles*). The *blue curve* represents the Debye-Hückel solution for this system, predicting an exponential decay on a known characteristic length scale. The decay of the electrostatic potential to zero in the center of the channel demonstrates that the assumption of bulk-like behaviour in this region is justified.

5.3 Electroosmotic Flow

Electroosmotic flow forms an ideal test scenario for the full scheme, which solves the electrokinetic equations. This is because it heavily relies on the coupling between the density distribution and flow field. In a typical problem involving electroosmotic flow, there is a liquid containing one or several species of charged solutes and an external electric field. The charged species create a drift velocity locally upon application of the E-field, which, through frictional coupling with the neutral fluid, causes the establishment of hydrodynamic flow. This flow in turn changes the distribution of charged species by introducting additional drift. In complex geometries, this intricate interplay can cause a multitude of interesting and unexpected phenomena [14]. To verify the correctness of this coupling in the electrokinetics solver, subject of this thesis, we need to ristrict ourselves to very simple geometries, for which reference soltuions can be obtained analytically.

Amongst the few geometries for which this is possible, is the slit-pore geometry introduced in section 5.1. As discussed in detail in an earlier study [27], the translational symmetry allows for the decoupling of the hydrodynamics from the diffusion, in all but one dimension. In addition to that, the symmetries simplify the occuring equations to a degree that allows one to find explicit solutions with purely analytical methods.

More concretely, applying an electrocal field along one of the directions parallel to the plates does not influence the charge distribution, which can only vary along the normal direction. The solution for the charge distribution given in (5.3) remains valid. Invoking local momentum conservation in the stationary state tells us that forces stemming from the E-field and acting on the charged species must be completely dissipated into the fluid. The force density acting on the fluid is therefore given by

$$\vec{f}(x) = q\rho(x) \ . \tag{5.8}$$

In addition, the symmetries cause the nonlinear term in the Navier-Stokes equations to vanish, leaving us with the simple equation

$$\frac{\partial_x^2 v_y(x)}{\partial x^2} = -\frac{qEC^2}{8\,k_\mathrm{B}T\,l_\mathrm{B}\,\eta} \cdot \cos^{-2}\left(\frac{qC}{2k_\mathrm{B}T} \cdot x\right) \ , \tag{5.9}$$

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

$$v_y(x) = \frac{E}{2\pi \, l_{\mathrm{B}} \, \eta \, q} \cdot \log \left[\frac{\cos\left(\frac{qC}{2k_{\mathrm{B}}T} \cdot x\right)}{\cos\left(\frac{qC}{2k_{\mathrm{B}}T} \cdot \frac{d}{2}\right)} \right] \ , \tag{5.10}$$

where x is the direction normal to the plates and y the direction of the E-field and parallel to the plates.

The results of a simulation of this system with parameters as in the single component system in section 5.1 are shown in figure 5.4 and 5.5. The applied E-field was $E = 0.1$ and the fluid consisted of a neutral species of density $\bar{\rho}_{\text{neutr.}} = 4$ and a species of counterions to the wall with density $\bar{\rho}_{\text{pos.}} = 0.004$ with a dynamic viscosity of $\eta_{\text{dyn.}} = \eta/(\bar{\rho}_{\text{neutr.}} + \bar{\rho}_{\text{pos.}}) = 1$.

The simulation results agree very well with the analytical ones, however, not as perfectly as for the previously tested systems. The velocities in the simulation exceed the analytically determined ones consistently by a factor of 2.7%. Further simulations of an ideal gas bulk system under the influence of an external force revealed that the cause of this is a nonlinearity in the discretized formulation of the diffusive flux, and subsequently, also in the the force density applied to the lattice-Boltzmann fluid. This effect was not reported in the original publication by Capuani et. al. [4], where the authors considered the term for force coupling only in the continuum limit, for which these nonlinearities vanish. This effect, as discussed in details in section 3.5 accounts precisely for the deviation of the simulation results from analytical calculations for a wide range of parameters tested.

Of the three different simulations carried out to verify the implementation, none took longer than several seconds. This consitutes a tremendous speedup over explicit particle simulations of the same system as carried out in [27], demonstrating the power of this approach.

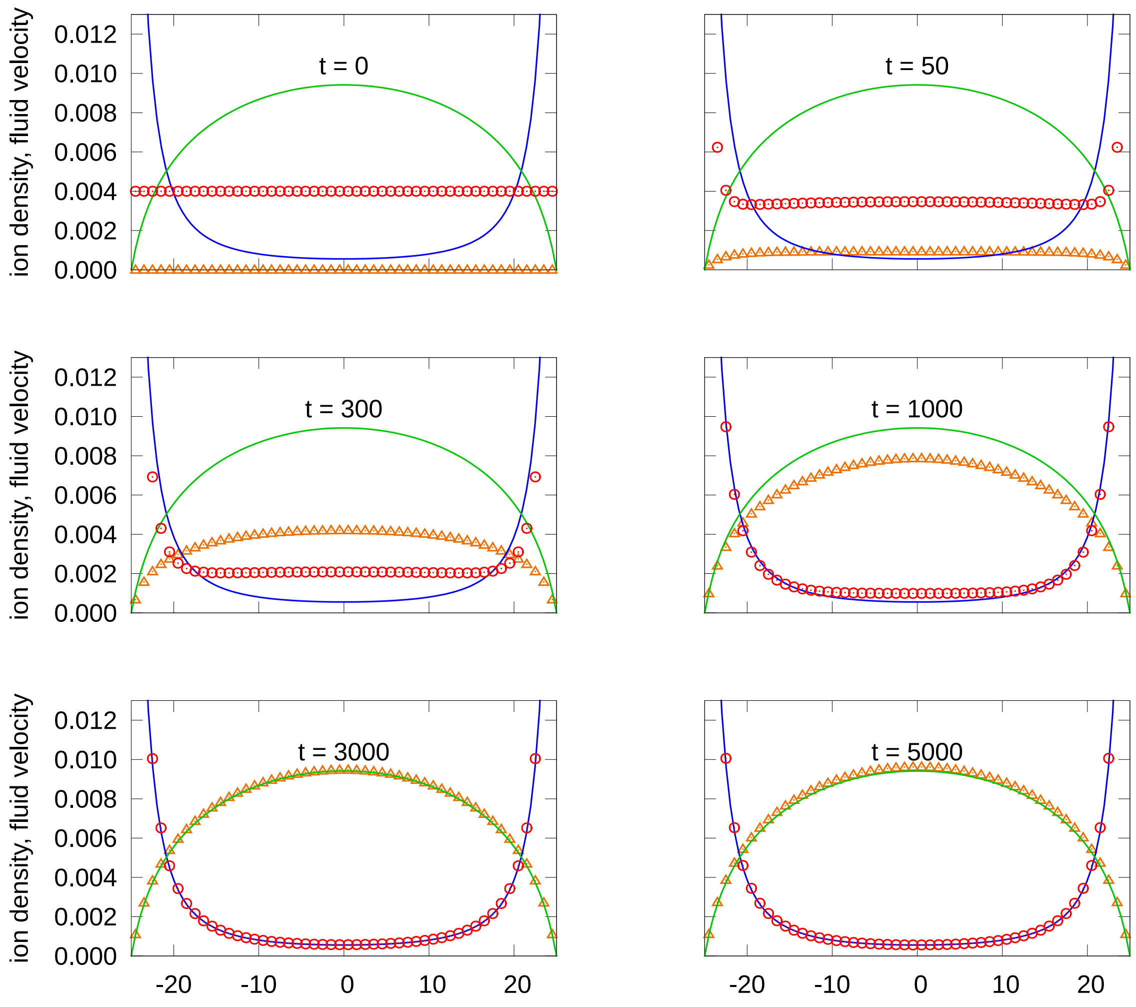

Figure 5.4: Initial, transient, and stationary states of of a single component electroosmotic flow in a channel, consisting of homogeneously charged parallel walls. Ion densities are depicted as *red circles*, fluid velocity profiles as *orange triangles*. The stationary state density (*blue line*) and velocity (*green line*) are shown as well, for comparison.

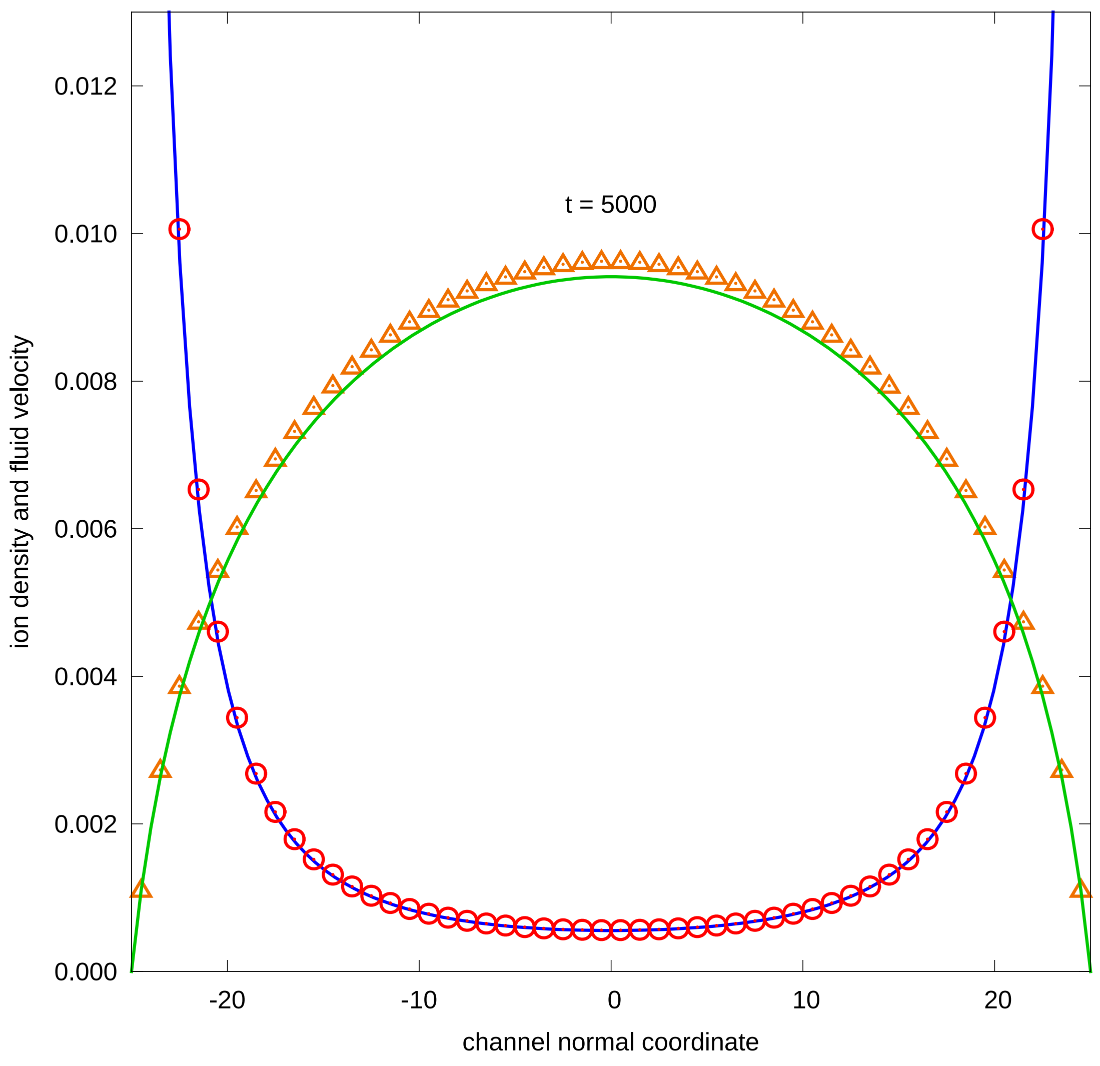

Figure 5.5: Magnification of the last frame from figure 5.4, depicting the stationary state density (*red circles*) and velocity profile (*orange triangles*) of the system consisting of homogeneously charged parallel flat walls, containing their counterions plotted over the coordinate perpendicular to the channel walls. Analytical predictions based on the electrokinetic equations for the density (*blue line*) and velocity (*green line*) are shown as well.

6 An Alternative Method for Hydrodynamics in Periodic Systems

In typical soft matter systems consisting of an aquaeous solution of macromolecules with sizes on the micrometer scale, the dynamics of the micromolecules happen on a time scale of milliseconds to seconds. In this case the flow field relaxes to a stationary state on a time scale much smaller than that of the movement of the macromolecules. It is also the case that the convective acceleration can safely be neglected due to the low Reynolds number of the flow on the order of 10^{-2} to 10^{-6}. In this case the stationary Stokes equations form a suitable model to describe the fluid flow.

In this chapter I want to discuss one possibility for an efficient algorithm to calculate such laminar flow fields of incompressible liquids in periodic boundary conditions. The method used to realize the incompressibility constraint is a so-called projection method, important not only in numerical applications but also in the mathematical theory of hydrodynamics. The idea for this part of the algorithm was taken from a publication by Starn in 2001 [31], who used it to construct an unconditionally stable Navier-Stokes solver for applications in computer graphics.

6.1 Stokes Equations

Consider the stationary Stokes equations describing laminar flow of an incompressible fluid at low Reynolds numbers

$$\vec{\Delta}\,\vec{v} = -\frac{\vec{f}}{\eta} + \nabla p\,, \tag{6.1}$$

$$\nabla \cdot \vec{v} = 0, \tag{6.2}$$

where η represents the kinematic viscosity and $\vec{\Delta}$ a laplacian acting component wise on the velocity field $\vec{v}$ in Cartesian coordinates.

They form a system of two linear, coupled, partial differential equations. While due to the linearity, this might seem trivial at first sight, the unusual coupling of the two equations makes it a numerically non trivial problem. Problems stem from the fact that one has to choose a proper pressure field as part of the source term in equation (6.1), so that equation (6.2) is fulfilled. In the following, I will explicitly construct such a pressure field by means of Helmholtz decomposition. This allows the resolution of those issues and can be used to construct a lattice algorithm, that scales like $O(n \cdot \log n)$, where n is the number of grid points.

6.2 The Algorithm

This idea, to use the fact that the flow has to be divergence free to reconstruct the proper pressure gradient, consists of the following two steps:

First, solve equation (6.1) without the pressure term to gain an initial solution for the flow field

$$\vec{\Delta}\, \vec{v}_{\text{inital}} = -\frac{\vec{f}}{\eta}\,. \tag{6.3}$$

This essentially forms a system of three uncoupled Poisson's equations, which, by means of finite differences, can easily be formulated as a system of linear equations, that then can be solved by a wide range of methods, some of which even scale as $O(n)$ (multigrid method). The solution $\vec{v}_{\text{inital}}$ to this system, however, will in general not be divergence free, as is the case in the previous example.

Secondly, by means of the Helmholtz decomposition, decompose the field into two parts; one for which the divergence vanishes at all points and one for which the curl vanishes everywhere. The divergence free part is given by

$$\vec{v}(\vec{x}) = \frac{1}{4\pi}\nabla \times \int_V \frac{\nabla \times \vec{v}_{\text{initial}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^3x' + \frac{1}{4\pi}\nabla \times \int_S \frac{\vec{v}_{\text{initial}}(\vec{x}') \times \hat{\vec{n}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^2x'\,, \tag{6.4}$$

with the second term consisting of an integral over the surface S of the simulation volume with normal vector $\hat{\vec{n}}$. This forms a solution to Stokes equations as will be shown in section 6.3.

6.3 Consistency of the Algorithm with Stokes equations

The vector field given in equation (6.4) is obviously divergence free as it is given by a vector potential. It therefore fulfills equation (6.2), however, this does not mean that it is necessarily a solution to equation (6.1). This has to be verified individually. To do so, consider the solution $\vec{v}$ to equation (6.1) with the proper pressure term.

$$\begin{aligned}\vec{\Delta}\vec{v} &= -\frac{\vec{f}}{\eta} + \nabla p \\ &= \vec{\Delta}\vec{v}_{\text{initial}} + \nabla p\end{aligned} \tag{6.5}$$

With the solution from equation (6.4), the pressure gradient can explicitly be given as

$$\begin{aligned}\nabla p &= \vec{\Delta}\vec{v} - \vec{\Delta}\vec{v}_{\text{initial}} \\ &= \vec{\Delta}(\vec{v} - \vec{v}_{\text{initial}}) \\ &= \vec{\Delta}\left(\frac{1}{4\pi}\nabla \times \int_V \frac{\nabla \times \vec{v}_{\text{initial}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^3x' + \frac{1}{4\pi}\nabla \times \int_S \frac{\vec{v}_{\text{initial}}(\vec{x}') \times \hat{\vec{n}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^2x' - \vec{v}_{\text{initial}}\right),\end{aligned} \tag{6.6}$$

or in nicer terms, applying the Helmholtz decomposition as defined in section 6.5 to $\vec{v}_{\text{initial}}$

$$\begin{aligned}\nabla p = \vec{\Delta}\Bigg(&\frac{1}{4\pi}\nabla \times \int_V \frac{\nabla \times \vec{v}_{\text{initial}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^3x' + \frac{1}{4\pi}\nabla \times \int_S \frac{\vec{v}_{\text{initial}}(\vec{x}') \times \hat{\vec{n}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^2x' \\ &-\frac{1}{4\pi}\nabla \times \int_V \frac{\nabla \times \vec{v}_{\text{initial}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^3x' - \frac{1}{4\pi}\nabla \times \int_S \frac{\vec{v}_{\text{initial}}(\vec{x}') \times \hat{\vec{n}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^2x' \\ &+\frac{1}{4\pi}\nabla \int_V \frac{\nabla \cdot \vec{v}_{\text{initial}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^3x' - \frac{1}{4\pi}\nabla \int_S \frac{\langle \vec{v}_{\text{initial}}(\vec{x}'), \hat{\vec{n}}(\vec{x}')\rangle}{|\vec{x}-\vec{x}'|}\, d^2x'\Bigg) \\ =&\frac{1}{4\pi}\vec{\Delta}\nabla\left(\int_V \frac{\nabla \cdot \vec{v}_{\text{initial}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^3x' - \int_S \frac{\langle \vec{v}_{\text{initial}}(\vec{x}'), \hat{\vec{n}}(\vec{x}')\rangle}{|\vec{x}-\vec{x}'|}\, d^2x'\right).\end{aligned} \tag{6.7}$$

As shown in 6.4, $\vec{\Delta}$ and ∇ commute for at least three times continuously differentiable functions so that

$$\nabla p = \frac{1}{4\pi}\nabla\Delta\left(\int_V \frac{\nabla \cdot \vec{v}_{\text{initial}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\, d^3x' - \int_S \frac{\langle \vec{v}_{\text{initial}}(\vec{x}'), \hat{\vec{n}}(\vec{x}')\rangle}{|\vec{x}-\vec{x}'|}\, d^2x'\right), \tag{6.8}$$

from which we can recover an explicit expression for the pressure

$$p(\vec{x}) = p(\vec{x}_0) + \frac{1}{4\pi}\int_{\vec{x}_0}^{\vec{x}} \Delta \left(\int_V \frac{\nabla \cdot \vec{v}_{\text{initial}}(\vec{x}'')}{|\vec{x}' - \vec{x}''|}\, d^3x'' - \int_S \frac{\langle \vec{v}_{\text{initial}}(\vec{x}''), \hat{\vec{n}}(\vec{x}'')\rangle}{|\vec{x}' - \vec{x}''|}\, d^2x'' \right) d\vec{x}' \tag{6.9}$$

This means that the flow field from equation (6.4) consistently solves Stokes equations. Note that existence, uniqueness, and stability is given for the stationary Stokes equations [10]. Under the condition that the solution lies in $C^3(\mathbb{R}^3)$, this method is guaranteed to find it. For arbitrary boundary geometries, this is generally not true, it might however be true for the simple geometries to which this algorithm is restricted and suffieciently smooth force densities. (c.f. section 6.6 for details on the boundary conditions).

6.4 Commutative property of $\vec{\Delta}$ and ∇

For scalar fields Ψ, the commutative property can easily be verified in cartesian coordinates.

$$\begin{aligned}
\vec{\Delta}\nabla\Psi &= \begin{pmatrix} \partial_x^2 + \partial_y^2 + \partial_z^2 \\ \partial_x^2 + \partial_y^2 + \partial_z^2 \\ \partial_x^2 + \partial_y^2 + \partial_z^2 \end{pmatrix} \begin{pmatrix} \partial_x \\ \partial_y \\ \partial_z \end{pmatrix} \Psi \\
&= \begin{pmatrix} \partial_x^2 + \partial_y^2 + \partial_z^2 \\ \partial_x^2 + \partial_y^2 + \partial_z^2 \\ \partial_x^2 + \partial_y^2 + \partial_z^2 \end{pmatrix} \begin{pmatrix} \partial_x\Psi \\ \partial_y\Psi \\ \partial_z\Psi \end{pmatrix} \\
&= \begin{pmatrix} \partial_x^3\Psi + \partial_y^2\partial_x\Psi + \partial_z^2\partial_x\Psi \\ \partial_x^2\partial_y\Psi + \partial_y^3\Psi + \partial_z^2\partial_y\Psi \\ \partial_x^2\partial_z\Psi + \partial_y^2\partial_z\Psi + \partial_z^3\Psi \end{pmatrix}
\end{aligned} \tag{6.10}$$

At this point we need to require that $\Psi \in C^3$ so that we can apply Schwartz' theorem

$$\begin{aligned}
\partial_i^2\partial_k\Psi &= \partial_i(\partial_i\partial_k\Psi),\ \Psi \in C^2 \\
&= \partial_i\partial_k(\partial_i\Psi),\ \partial_i\Psi \in C^2 \Leftarrow \Psi \in C^3 \\
&= \partial_k\partial_i^2\Psi\,,
\end{aligned} \tag{6.11}$$

to express $\vec{\Delta}\nabla\Psi$ as a gradient field

$$\begin{aligned}
\vec{\Delta}\nabla\Psi &= \begin{pmatrix} \partial_x^3\Psi + \partial_x\partial_y^2\Psi + \partial_x\partial_z^2\Psi \\ \partial_y\partial_x^2\Psi + \partial_y^3\Psi + \partial_y\partial_z^2\Psi \\ \partial_z\partial_x^2\Psi + \partial_z\partial_y^2\Psi + \partial_z^3\Psi \end{pmatrix} \\
&= \begin{pmatrix} \partial_x(\partial_x^2\Psi + \partial_y^2\Psi + \partial_z^2\Psi) \\ \partial_y(\partial_x^2\Psi + \partial_y^2\Psi + \partial_z^2\Psi) \\ \partial z(\partial_x^2\Psi + \partial_y^2\Psi + \partial_z^2\Psi) \end{pmatrix} \\
&= \begin{pmatrix} \partial_x \\ \partial_y \\ \partial_z \end{pmatrix} (\partial_x^2\Psi + \partial_y^2\Psi + \partial_z^2\Psi) \\
&= \begin{pmatrix} \partial_x \\ \partial_y \\ \partial_z \end{pmatrix} (\partial_x^2 + \partial_y^2 + \partial_z^2)\Psi \\
&= \nabla\Delta\Psi
\end{aligned} \tag{6.12}$$

So for Ψ a three times continuously differentiable scalar field, it is $\vec{\Delta}\nabla = \nabla\Delta$.

6.5 Efficient implementation of the Helmholtz decomposition

The real space representations of the curl free ($\vec{v}_{\text{div}}$) and divergence free ($\vec{v}_{\text{curl}}$) parts of a vector field $\vec{v}$,

$$\vec{v}_{\text{div}}(\vec{x}) = -\frac{1}{4\pi}\nabla\int_V \frac{\nabla\cdot\vec{v}(\vec{x}')}{|\vec{x}-\vec{x}'|}\,d^3x' + \frac{1}{4\pi}\nabla\int_S \frac{\langle\vec{v}(\vec{x}'),\hat{\vec{n}}(\vec{x}')\rangle}{|\vec{x}-\vec{x}'|}\,d^2x'\,, \tag{6.13}$$

$$\vec{v}_{\text{curl}}(\vec{x}) = \frac{1}{4\pi}\nabla\times\int_V \frac{\nabla\times\vec{v}(\vec{x}')}{|\vec{x}-\vec{x}'|}\,d^3x' + \frac{1}{4\pi}\nabla\times\int_S \frac{\vec{v}(\vec{x}')\times\hat{\vec{n}}(\vec{x}')}{|\vec{x}-\vec{x}'|}\,d^2x'\,, \tag{6.14}$$

are convenient for analytical calculations but unfortunately impractical for numerical purposes due to the $O(n^2)$ computational complexity for evaluating those expressions on a grid with n points. Fortunately, one can show that

$$\vec{v}_{\text{div}}(\vec{x}) = \text{FFT}^{-1}\left[\frac{\langle\text{FFT}[\vec{v}(\vec{x})](\vec{k}),\vec{k}\rangle}{k}\right](\vec{x}) \tag{6.15}$$

and obviously

$$\vec{v}_{\mathrm{curl}}(\vec{x}) = \vec{v}(\vec{x}) - \vec{v}_{\mathrm{div}}(\vec{x}) \,, \tag{6.16}$$

which only requires operations of linear complexity in Fourier space and Fourier transformations, which can be computed with $O(n \cdot \log n)$ complexity.

6.6 Boundary conditions

The single most important drawback of using Fourier transformations in any computation is the limited applicability to a small set of simple boundary conditions. These are: periodic, Dirichlet and Neumann boundary conditions, each defined on a pair of opposite sides of a cuboid. The calculation of the initial solution $\vec{v}_{\text{init}}$ could be performed using a method capable of implementing complex boundary conditions and the Helmholtz decomposition with the additional surface terms would be able to preserve those. Unfortunately, this general type of the Helmholtz decomposition including surface terms can not be implemented efficiently using Fourier transformations.

The boundary conditions therefore have to be limited to periodic boundaries. With some effort to replace the transformation in one direction from exponential to cosine transformations, one could implement a system of parallel plates with no-slip boundary conditions

6.7 Implementation

For evaluation purposes, the method was implemented using the Python programming language [8]. Python is an open source scripting language offering features such as dynamic type setting, automatic memory management, and array slicing. In conjunction with SciPy [20, 26], a comprehensive scientific library including many optimized implementations of high level numerical algorithms, it provides a powerful framework for rapid algorithm development.

Using this framework, an implementation of the algorithm presented in this chapter for two dimensional systems can be achieved in under 100 lines of code. The inital step, consisting of solving Poisson's equation is carried out, using a finite-difference discretization similar to the one used in section 3.3. The resulting system of linear equations is solved using an implementation of the CG method readily available in the SciPy library.

Apart from initializations of the finite-difference Matrix `A` and the force density `b`, the implementation essentially reduces to

```
v = zeros([n, 2])
v[:,0] = cg(A, b[:,0], v[:,0], maxiter=10000)[0]
v[:,1] = cg(A, b[:,1], v[:,1], maxiter=10000)[0]
v_hd = helmholtz_decompose_fft(v)
```

The central part of the algorithm, the FFT based Helmholtz decomposition for periodic systems, can be implemented in the following way, making use of the FFT implementations in the SciPy library

```
def helmholtz_decompose_fft(v):
  n = array(v.shape[0:2], int)

  vk = empty([n[0], n[1]/2+1, 2], complex)
  vk_div = empty([n[0], n[1]/2+1, 2], complex)
  vk_rot = empty([n[0], n[1]/2+1, 2], complex)
  v_hd = empty([2, n[0], n[1], 2])

  vk[:,:,0] = rfft2(v[:,:,0])
  vk[:,:,1] = rfft2(v[:,:,1])

  for x in arange(0, n[0]):
    for y in arange(0, n[1]/2+1):
      if x == 0 and y == 0:
        vk_div[x,y] = [0.0, 0.0]
        vk_rot[x,y] = vk[x,y]
      else:
        if y > n[1]/2:
          ky = y - n[1]
        else:
          ky = y

        if x > n[0]/2:
          kx = x - n[0]
        else:
          kx = x

        vk_div[x,y,0] = (kx * vk[x,y,0] + ky * vk[x,y,1]) * kx / (kx**2 + ky**2)
        vk_div[x,y,1] = (kx * vk[x,y,0] + ky * vk[x,y,1]) * ky / (kx**2 + ky**2)
        vk_rot[x,y] = vk[x,y] - vk_div[x,y]

  v_hd[0,:,:,0] = irfft2(vk_div[:,:,0])
  v_hd[0,:,:,1] = irfft2(vk_div[:,:,1])
  v_hd[1,:,:,0] = irfft2(vk_rot[:,:,0])
  v_hd[1,:,:,1] = irfft2(vk_rot[:,:,1])

  return v_hd
```

6.8 Results

For a force density localized in the center of the periodic domain, the program yields a flow field corresponding to the so-called Stokeslet, the Greens function of the Stokes equations. It does so by first computing the solution to the related pressure-gradient free system, resembling Poisson's equation. As shown in figure 6.1, this solution is not free of sources and does therefore not fulfill the incompressibility constraint.

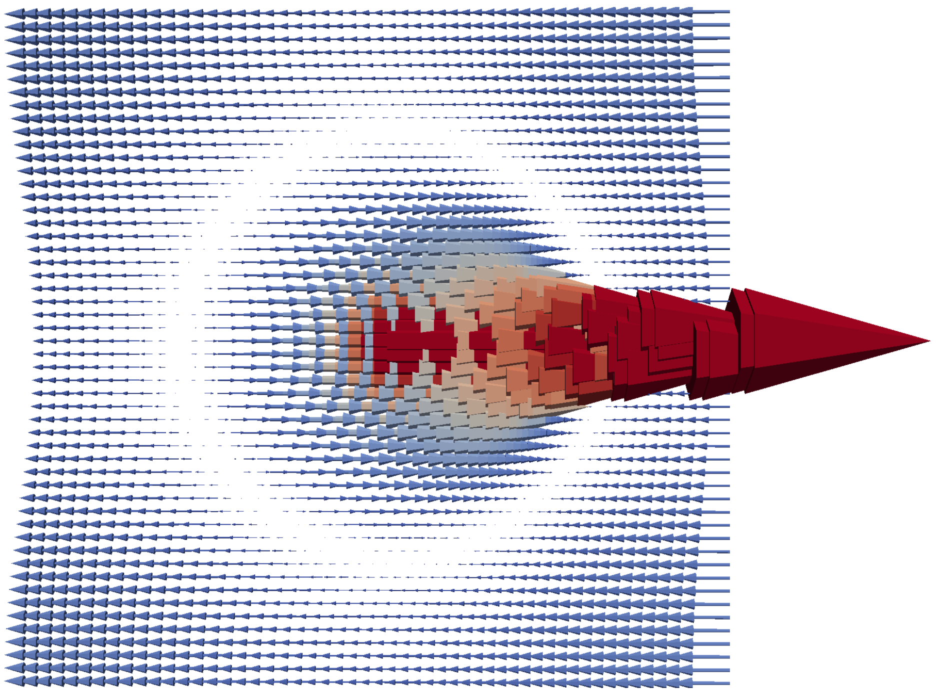

Figure 6.1: The initial solution to Stokes equation without the pressure term. This flow field is not free of sources. The calculation was carried out on a two dimensional grid of size 256^2.

In the second step, this inital flow field is decomposed into a rotation free and a divergence free component by means of the Helmholtz decomposition. Figure 6.2 and 6.3 show these rotation and divergence free fields, respectively.

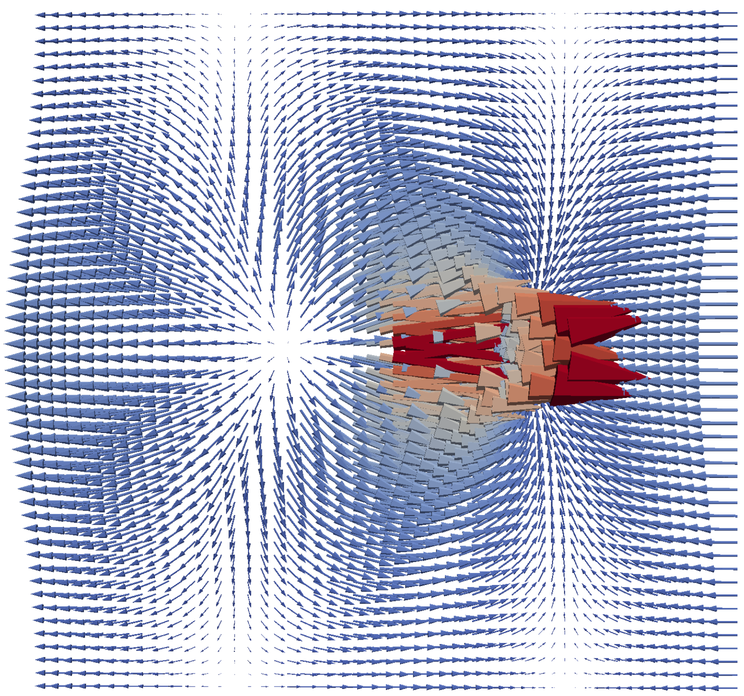

Figure 6.2: The rotation free component of the initally computed flow field for the pressure free problem shown in figure 6.1. According to the derivation in section 6.3, the pressure is completely determined by this field.

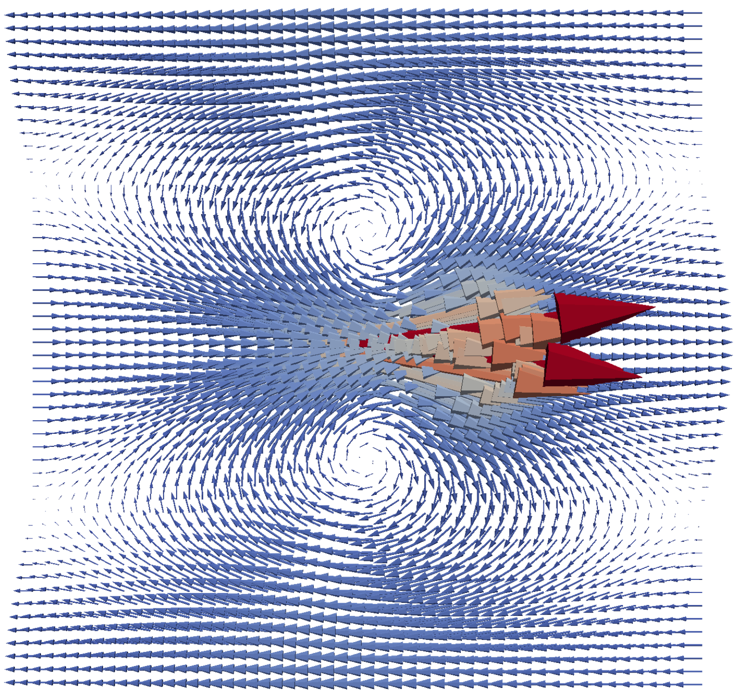

Figure 6.3: The divergence free component of the initally computed flow field for the pressure free problem shown in figure 6.1. As demonstrated in 6.3, this forms the unique solution to Stokes equations with the pressure determined such that the incompressibility constraint is fulfilled. In this case the force density is highly localized in the center of the periodic domain while a homogeneous force in the opposite direction counteracts it, such that the net force is zero. Without this there could not be any stationary solution since there is no mechanism to dissipate momentum.

The divergence free part finally gives the desired solution to the full stationary Stokes equations. The 0th Fourier mode of the flow field represents the constant part of the field, it has neither divergence, nor rotation and can therefore be arbitrarily attributed to either component. In this application it is taken as part of the divergence free component and final solution, so that this flow field has the same center of mass velocity as the inital solution where the center of mass is at rest.

In a CUDA accelerated implementation for production, the inital solution can be obtained using the FFT Poisson solver from section 3.3. This saves considerable effort, since the FFTs for solving Poisson's equation and the Helmholtz decomposition only have to be carried out once. The structure of the implementation then very much resembles the one from section 3.3, which means that the benchmarking results from section 4.5 are valid for this method as well. The only difference being that this method deals with vector fields instead of scalar fields, which means the computation time has to be multiplied by the dimensionality of the problem. While the scaling of the resulting method is $O(n \cdot \log n)$ and worse than the $O(n)$ scaling of the lattice-Boltzmann method, is would speed computations up considerably for smaller system sizes of up to 400^3.

Another advantage of this method over LB is the strict Gallilean invariance, which would allow for transformations into arbitrary frames of reference during a simulation run.

7 Conclusions

A numerical scheme to solve the time-dependent electrokinetic equations, previously proposed by Capuani et. al. [4] has been implemented in the ESPResSo molecular dynamics simulation package. The implementation makes use of CUDA capable graphics processing units in order to take advantage of the tremendous computational power that these devices provide.

The ability to include charged species of particles in an molecular dynamics simulation through a continuum description like this makes it no longer neccessary to resolve all the microscopic detail of the dynamics of electrolytic solutions. This represents a significant improvement, particularily for simulations of systems containing macromolecules, such as DNA, proteins, colloids, and polymers, whose dynamics happen on length and time scales several orders of magnitude bigger than those of the ionic solutes.

Thorough derivations of the underlying theory from statistical mechanics justify the mechanics of the algorithm and provide limits on its applicability. Notably, the transient dynamics reproduced by the algorithm are not valid in situations where there is a strong potential gradient present on the length scale of the grid spacing. While this can be overcome by increasing the grid resolution, it is important to be aware of this fact, which was not reported in the original publications introducing this method. Despite these limitations, the algorithm appears to be quite promising as a highly efficient Poisson-Boltzmann solver for complex geometries, in addition to being a fast and effective way of solving the standard electrokinetic model.

In order to verify the accuracy of the model, three test systems were investigated, for which analytical solutions were obtained. The first system consisted of a single charged species confined in between charged parallel plates. The wall charge was chosen, such that the linearized Debye-Hückel theory does not apply, and yet the results were still in perfect accordance with the analytic solution. This demonstrates that the algorithm is indeed capable of capturing the nonlinear terms of the Poisson-Boltzmann equation. Further tests were done on the two plate system with the application of an external electric field, inducing electroosmotic flow along the channel. For reasonably sized electric fields, this produced analytically correct results, showing the validity of the full scheme, including the hydrodynamic coupling.

While this module was originally developed with an eye towards elecrohydrodynamics, it provides a solid framework upon which other multicomponent problems can be tackled. One such problem, which is of increasing interest, is reaction-diffusion systems, such as the nonequilibrium swimmers discussed in more detail in the next chapter.

In addition to that, several other hydrodynamic and electrostatic solvers were implemented, which each had their own unique advantages. One possible application for the CUDA accelerated electrostatic method based on finite-differences and conjugate gradients are nanopore systems, for which dielectric contrast is known to have significant consequences.

8 Outlook

8.1 Particle Coupling

In order to not just be able to simulate electro-osmotic flow in complex geometries but also electrophoresis of macro molecules, there needs to be a coupling of the electrokinetics solver as presented in this thesis, to the particle trajectory integration routines in ESPResSo. The lattice-Boltzmann implementation, which was used in the implementation of the electrokinetics solver and is already present in ESPResSo, achieves this by introducing a frictional coupling between the fluid and particles. Particles and fluid exert a force on each other, which is proportional to their relative velocity. Since the particles are modeled as point-like and their positions do not necessarily coincide with the LB mesh nodes, a simple linear interpolation scheme is used to interpolate the fluid velocity onto the particles position and to extrapolate the force exerted by the particle on the fluid onto the LB mesh. This method has been shown to reproduce the correct hydrodynamic far field and allows for the construction of extended objects from multiple point-like beads [7].

While this approach could be applied without modifications to the electrokinetics solver in simulations only containing neutral particles, it would fail to reproduce the expected behaviour in a simulation containing charged particles. This is because there is no mechanism that would stop species of counterions from collapsing “into" a charged particle, completely neutralizing its charge and therey preventing the build up of the correct co- and counterion distribution around said charged particle.

The first idea that comes to mind to fix this issue, is to declare a number of nodes around a charges particle to be wall nodes. The force exerted by the fluid on this wall can easily be calculated by requiring momentum conservation in the bounce-back step. Issues arise when one allows such particles to move and nodes that had previously been fluid become wall and vice versa. Ladd explored ways to resolve those issues, which led to the now widespread use of these so-called Ladd boundaries [7]. Since this scheme depends on the possibility of deleting populations at the forward surface of a moving particle and initializing populations at the back surface by interpolating modes from surrounding nodes, it can not be applied directly to charged species, since it would violate charge neutrality and / or introduce non-physical charge transfer processes.

The most promising candidate to solve this problem is a scheme based on the application af an additional potential that reprells both like and oppositely charged species from the volume occupied by a particle. Going back to the initial derivation of the electrokinetics model based on statistical machanics, one can introduce another penalty term into the local free energy, accounting for the excluded volume of either only MD beads or MD beads and diffusive species.

8.2 Beyond Poisson-Boltzmann

As presented in detail in section 2, the foundation of the electrokinetics model presented in this thesis is the definition of a free energy containing only an ideal gas contribution and a contribution to the internal energy from electrostatic interactions. This yields a model, which is valid in the same limits as Poisson-Boltzmann theory, which are explicitly stated in section 2.2.

As mentioned in the discussion of possible particle couplings (sec. 8.1), an inclusion of additional contributions into the free energy is straightforward and in principle allows for the inclusion of excluded volume and charge correlation effects into the model. The change in free energy due to excluded volume can be calculated analytically and the effect of charge correllations on the internal energy can be obtained from explicit particle simulations.

If successful, this approach could extend the range of application of this model significantly, making very efficient simulations of polymer and colloidal elecrophoresis with finite Debye-lengths in the presence of multivalent salt possible.

8.3 Reaction-Diffusion Systems

One system, whose dynamics are partially governed by chemical reactions and which is currently being investigated by the soft matter community are self-propelled swimmers. One realization of such a swimmer consists of a spherical colloid made from silica and coated partially with platinum. In a solution of hydrogen peroxide (H_2O_2), the platinum acts as a catalyzer for the decomposition of H_2O_2 into H_2O and O_2. In a process called self-diffusiophoresis, the complicated interplay of diffusion and hydrodynamics causes a situation in which the particle is propelled in the direction away from the coated side even at very low fuel concentrations.

Computer simulations have been used to investigate this and many other forms of chemically swimming colloids [23, 3, 24]. Spacially resolved dynamics of systems which contain reactive species are often modeled by diffusion equations with additional source terms or complicated boundary conditions accounting for the chemical reactions depleting certain species and producing others. These equations pose a numerical challange and are typically treated with FEM methods. In the case of the chemical swimmer, it is sometimes appropriate to assume a constant fuel concentration, which allows one to model the chemical reactions using simple constant flux (Neumann) boundary conditions [3]. In this case Poisson's equation describes the stationary state of the diffusive processes, for which a multitude of efficient methods of solution exists. In other cases this assumption does not hold, such as in swimmers with highly confined chemically active regions [24].

Molecular dynamics simulations using ESPResSo, modeling the solvent, fuel, and product species using DPD have been carried out, investigating the properties of swimmers of different geometries [23]. In those simulations, the fact that a fuel particle decomposes into several product particles, which leads to an increase in the pressure, has been modeled with different interaction potentials for DPD particles representing fuel and product particles. The issue with simulations like this is the mapping of desired properties, such as diffusion constants, equations of state, viscosities, and so on, to the DPD parameters.

Leaving out the electrostatics from the electrokinetics solver presented in this thesis yields a mesoscopic lattice based method for diffusion-advection systems. By adding one more step in the integration loop, that converts densities between different species, according to the rules prescribed by the reactions, this scheme should be used to model the dynamics of diffusion-advection-reaction systems. The well known relations of this method with the statistical mechanics on the mesoscopic scale on one hand, and the macroscopic continuum description on the other, allow for much more direct control of the desired parameters.

8.4 Varying Permittivity

One detail that is often neglected in simulations of soft matter systems is spacial variation of the electrical permittivity [18, 19, 27]. Due to the relatively high permittivity of water, there is plenty of opportunity to form interfaces of high dielectric mismatch in soft matter systems and the impact on the dynamics of the system can be profound [21, 22].

In its current implementation, the electrokinetics solver does not allow for spacially varying permittivity due to the use of an FFT based finite-difference solver as discussed in sections 3.3 and 3.4. A more versatile iterative solver like the conjugate-gradient based finite-difference solver presented in section 4.4 can easily be adapted to work with spacially varying permittivity. In addition to that, it is also possible to model surfaces with dielectric mismatch using explicit surface charges, chosen such that the boundary conditions for the electrical field, stemming from the dielectric mismatch, are fulfilled. One such scheme, the ICC$\star$ algorithm, is implemented in ESPResSo [21] and could in principle be used to turn the current direct solver into an iterative solver, that is capable of realizing the proper boundary conditions for systems with inhomogeneous permittivity.

However, the current implementation of the ICC$\star$ algorithm is completely CPU based and some work on the code infrastructure would be needed to interface it with GPU based electrostatics solvers.

8.5 Fluctuations

In flows on the the macroscopic scale, fluctuations in density, momentum, and stress play no role in hydrodynamics due to their microscopic magnitudes at room temperature, compared to the magnitude of the effects usually observed on this scale. By contrast, in systems on the micro- and even more so, the nanoscale, local fluctuations in velocities and stresses usually dominate over the flow velocities and stresses present in a deterministic continuum description of the same system and the typically smooth flow profiles from the latter description can only be recovered as averages over comparatively long time scales.

Since these fluctuations are the cause of Brownian motion, the microscopic mechanism of diffusion of particles, they can not be neglected in a simulation of a system containing macromolecules. Doing so would result in purely advective propagation and the complete abscence of diffusion for said macromolecules.

Fluctuations have successfully been implemented in the lattice-Boltzmann algorithm with a multi-relaxation time collison operator, by adding a stochastic term to all non-conserved modes [1]. This method leads to fluctuations in density and momentum, exhibiting the correct distribution and fulfilling the fluctuation-dissipation theorem. As further investigations by the same authors show, this result should also apply to non-ideal LB methods, as long as the additional interactions are reversible [12, 13]. In general this does not hold, since dissipative interactions cause deviations from the Maxwell-Boltzmann distribution of momenta, which in turn requires complicated correlations in the applied noise pattern. A more thorough investigation as to whether this result applies to the electrokinetics solver as presented in this thesis is neccessary, though, before using in in conjunction with the the unmodified LB fluctuation scheme [1, 7] already implemented in ESPResSo for simulations on very small length scales.

Bibliography

[1] R. Adhikari, K. Stratford, M. E. Cates, and A. J. Wagner. Fluctuating lattice boltzmann. *EPL (Europhysics Letters)*, 71(3):473, 2005.

[2] A. Arnold, O. Lenz, S. Kesselheim, et al. Espresso 3.1: Molecular dynamics software for coarse-grained models. In *Meshfree Methods for Partial Differential Equations VI*, pages 1–23. Springer Berlin Heidelberg, 2013.

[3] L. Baraban, M. Tasinkevych, M. N. Popescu, et al. Transport of cargo by catalytic janus micro-motors. *Soft Matter*, 8(1):48–52, 2012.

[4] F. Capuani, I. Pagonabarraga, and D. Frenkel. Discrete solution of the electrokinetic equations. *Journal of Chemical Physics*, 121(2):973–986, 2004.

[5] Joan J Cerdà, Christian Holm, and Kurt Kremer. Novel simulation approaches for polymeric and soft matter systems. *Macromolecular Theory and Simulations*, 20(7):444–445, 2011.

[6] S. Dalton and N. Bell. Cusp : A c++ templated sparse matrix library. http://cusplibrary.github.com//.

[7] B. Dünweg and A. Ladd. Lattice boltzmann simulations of soft matter systems. *Advanced Computer Simulation Approaches for Soft Matter Sciences III*, pages 89–166, 2009.

[8] The Python Software Foundation. *Python v2.7.3 documentation*. http://docs.python.org/.

[9] R. J. Gonsalves. Computational Physics I, 2004. Lecture notes. http://www.physics.buffalo.edu/phy410-505-2004/Chapter6/ch6-lec2.pdf.

[10] D. Götz, P. Wittbold, and E. Emmrich. Eine Einführung in das Navier-Stokes-Problem, 2006/07. Seminararbeit.

[11] K. Grass. *Towards realistic modelling of free-solution electrophoresis: a case study on charged macromolecules*. PhD thesis, Goethe-Universität Frankfurt, 2009.

[12] M. Gross, R. Adhikari, M. E. Cates, et al. Thermal fluctuations in the lattice boltzmann method for nonideal fluids. *Physical Review E*, 82(5):056714, 2010.

[13] M. Gross, R. Adhikari, M. E. Cates, et al. Modelling thermal fluctuations in non-ideal fluids with the lattice boltzmann method. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1944):2274–2282, 2011.

[14] O. A. Hickey, C. Holm, J. L. Harden, et al. Implicit method for simulating electrohydrodynamics of polyelectrolytes. *Physical review letters*, 105(14):148301, 2010.

[15] C. Holm. Simulation Methods I. Lecture Notes, 2009/10.

[16] P. J. Hoogerbrugge and V. A. Koelman. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *Europhys. Lett*, 19(3):155–160, 1992.

[17] J. Horbach and D. Frenkel. Lattice-boltzmann method for the simulation of transport phenomena in charged colloids. *Physical Review E*, 64(6):061507, 2001.

[18] J. P. Hsu, K. L. Liu, W. L. Hsu, et al. Diffusiophoresis of a charge-regulated spherical particle normal to two parallel disks. *The Journal of Physical Chemistry B*, 114(8):2766–2778, 2010.

[19] J. P. Hsu, K. L. Liu, W. L. Hsu, et al. Diffusiophoresis of a soft sphere normal to two parallel disks. *Langmuir*, 26(20):16037–16047, 2010.

[20] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. http://www.scipy.org/.

[21] S. Kesselheim, M. Sega, and C. Holm. Applying ICC$\star$ to dna translocation: Effect of dielectric boundaries. *Computer Physics Communications*, 182(1):33–35, 2011.

[22] S. Kesselheim, M. Sega, and C. Holm. Effects of dielectric mismatch and chain flexibility on the translocation barriers of charged macromolecules through solid state nanopores. *Soft Matter*, 2012.

[23] Francesca Lugli, Emiliano Brini, and Francesco Zerbetto. Shape governs the motion of chemically propelled janus swimmers. *The Journal of Physical Chemistry C*, 116(1):592–598, 2011.

[24] B. Nijs, A. Blaaderen, J. M. Roeland, et al. Fuel concentration dependent movement of supramolecular catalytic nanomotors. *Nanoscale*, 5(4):1315–1318, 2013.

[25] Nvidia. *CUDA Toolkit Documentation*, 2012. http://docs.nvidia.com/cuda/.

[26] P. Peterson. F2py: a tool for connecting fortran and python programs. *International Journal of Computational Science and Engineering*, 4(4):296–305, 2009.

[27] G. Rempfer. Lattice-boltzmann simulations in complex geometries. Master's thesis, Universtity of Stuttgart, ICP, 2010.

[28] D. Röhm. Lattice boltzmann simulations on gpus. Master's thesis, Universtity of Stuttgart, ICP, 2011.

[29] D. Röhm and A. Arnold. Lattice boltzmann simulations on gpus with espresso. *The European Physical Journal-Special Topics*, 210(1):89–100, 2012.

[30] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 1994.

[31] J. Starn. A simple fluid solver based on the fft. *Journal of graphics tools*, 6(2):43–52, 2001.

[32] S. Succi. *The lattice Boltzmann equation*. Oxford university press New York, 2001.

[33] S. Succi, M. Sbragaglia, and S. Ubertini. Lattice boltzmann method. *Scholarpedia*, 5(5):9507, 2010.

[34] A. J. Wagner. The origin of spurious velocities in lattice boltzmann. *International Journal of Modern Physics B*, 17:193–196, 2003.

Acknowledgement

I want to thank

Christian Holm,
my supervisor and head of the Institute for Computational Physics, for his trust and investment in me over the course of the last three years.

Owen Hickey,
my colleague and friend, for sharing his extensive knowledge on electrohydrodynamics, and countless hours of joint development and debugging of simulation code, as well as the proof reading of this thesis.

Axel Arnold,
for his valuable input on electrostatics and GPU programming.

Dominic Röhm,
for his help in understanding his GPU LB implementation.

Olaf Lenz,
from who I learned much about professional software development in teams.

Florian Dommert,
for help with taming LaTeX,

Frank Huber,
the Administrator at the Institute for Computational Physics and the only Administrator I ever met, with an average reaction time of 30 s to any IT problem.

as well as my wife **Allison Rempfer**, my parents **Jörg Rempfer** and **Petra Zaiser-Rempfer**, and Allison's parents **George** and **Jean Fuller**, for their understanding and continuous support.

I hope that I am as valuable to those persons, as they are to me.