Rotne–Prager based hydrodynamics on GPUs

Master's Thesis

David Schwörer
January 28, 2015

Supervisor: Prof. Dr. Christian Holm
Institute for Computational Physics
University of Stuttgart

Co-Examiner: Prof. Dr. Siegfried Dietrich
Theory of Inhomogeneous Condensed Matter
Max Planck Institute for Intelligent Systems

Abstract

Materials with energy scales on the same order of magnitude as the thermal energy are considered “soft matter” as opposed to e.g. crystals or metals. Aside from polymers, gels, foams, liquids, as well as most biological matter, colloidal systems fall within this category. Colloids are particles whose size range from microns to millimeters, typically immersed in a solvent. Colloidal systems can be studied efficiently using coarse-grained computer simulations. In such simulations, the solvent is often modeled as a continuum using the Navier–Stokes equations.

Due to its non-linearity solving the Navier–Stokes equations is numerically challenging. An efficient method for solving the Navier–Stokes equations is the lattice-Boltzmann method. By coupling point-particles to this discrete fluid solver, coarse-grained soft matter simulations can be accelerated. In realistic suspension the long- and short-time dynamics of the colloids are well pronounced and differentiable. Usually the correct long-time behavior is sufficient for studying soft matter, such as polymer mobilities in salty environments, and other material transport properties. But the short-time dynamics can be important, for example in the crystallization of soft spheres. It is known that lattice-Boltzmann produces the correct flow field on long-time and length scales, but can produce unphysical results in the case of dense colloidal suspensions, where lattice-Boltzmann produces unphysical short-time dynamics.

For soft matter systems the solvent is assumed to be in the creeping flow limit. This means that the fluid relaxes many orders of magnitude faster than the structure of the embedded particles changes. With these limitations, the Navier–Stokes equations can be simplified and the Stokes equation is obtained. A method to simulate the hydrodynamic interactions in Stokes flow is Stokesian Dynamics. The solution for the Stokes equation of two interacting spheres is used to construct the mobility matrix containing the effective hydrodynamic interactions. The method has been first proposed by Brady in 1988 [1].

As part of this thesis Stokesian Dynamics has been implemented in ESPResSo for the use of graphic processing units (GPUs) by taking advantage of the Compute Unified Device Architecture (CUDA) framework. CUDA allows to use the massively parallel computing power of Nvidia GPUs. Using this power allows the simulation of a few thousand particles efficiently.

In order to compare the effects of the hydrodynamic interaction, the hydrodynamic

function was measured for both lattice-Boltzmann and Stokesian Dynamics. Lattice-Boltzmann with parameters typically used in the literature shows a discrepancy at short times, compared to the Stokesian Dynamics results. The Stokesian Dynamics results are assumed to be more reliable, as Stokes Dynamics is in the creeping flow limit, which is not the case for lattice-Boltzmann. The fact that lattice-Boltzmann produces different results, calls into question whether the hydrodynamic interactions produced with the typically used parameters are physical.

Choosing drastically different parameters, good agreement between lattice-Boltzmann and Stokesian Dynamics was found for either the mean squared displacement or the hydrodynamic function. In the case of a low viscosity, the hydrodynamic function shows good agreement to the Stokesian Dynamics simulation. In this case, the mean squared displacement at times close to the long-time diffusion still shows ballistic effects from the fluid. For high viscosities, the mean squared displacement matches better, but the hydrodynamic function shows nearly no hydrodynamic interactions. It is therefore unclear what parameter sets reproduce right physics, or if indeed there exists a set of parameters which produce both a reasonable mean squared displacement and hydrodynamic function.

Zusammenfassung

Stoffe mit Energieskalen in der gleichen Größenordnung wie die thermische Energie, werden als weiche Materie (soft matter) bezeichnet im Gegensatz zu Kristallen oder Metallen. Neben Polymeren, Gelen, Schäumen und den meisten biologischen Stoffen gehören Kolloide in diese Kategorie. Kolloide sind Teilchen deren Größe von Millimetern zu Mikrometern liegen und befinden sich typischerweise in einem Lösungsmittel. Kolloidale Systeme können mit Hilfe von vergröberten (coarse-grained) Computersimulation effektiv studiert werden. In diesen Simulationen wir das Lösungsmittel oft auf der Kontinuumsebene mit den Navier–Stokes Gleichungen beschreiben.

Aufgrund der nicht-linearität der Navier–Stokes Gleichungen ist es numerisch anspruchsvoll sie zu lösen. Eine effektive Methode zur Lösung der Navier–Stokes Gleichungen ist die Gitter Boltzmann Methode (lattice-Boltzmann). Durch Kopplung von Punktteilchen mit dem diskretem Fluid Löser können weiche Materie Simulationen beschleunigt werden. In realistischen Lösungen kann man zwischen Kurz- und Langzeitverhalten der Kolloide unterscheiden. Zumeist reicht es aus, das korrekte Langzeitverhalten der weichen Materie zu simulieren, zum Beispiel wenn die Beweglichkeit von Polymeren in salziger Flüssigkeit bestimmt wird. Aber auch das Kurzzeitverhalten kann wichtig sein, zum Beispiel wenn man die Kristallisation von weichen Kugeln betrachtet. Es ist bekannt, dass die Gitter Boltzmann Methode das richtige Fließverhalten berechnet für große Zeit- und Längenskalen. In dichten kolloidalen System kann es dabei aber zu deutlichen Abweichungen kommen.

Bei weicher Materie wird meist davon ausgegangen, dass das Fluid als schleichende Strömung beschrieben werden kann. Das ist dann der Fall, wenn die Flüssigkeit um viele Größenordnungen schneller relaxiert als sich die Struktur der darin eingebetteten Kolloide ändert. Mit dieser Annahme können die Navier–Stokes Gleichungen zu den Stokes Gleichungen vereinfacht werden. Eine Möglichkeit die hydrodynamischen Wechselwirkungen zu simulieren ist die Stokes'sche Dynamik (Stokesian Dynamics). Dabei wird die Lösung der Stokes Gleichung für zwei Kugeln verwendet, und damit eine Mobilitätsmatrix konstruiert, die die effektiven hydrodynamischen Wechselwirkungen beinhaltet. Diese Methode wurde zuerst von Brady 1988 vorgeschlagen [1].

Als Teil dieser Masterarbeit wurde diese Methode in ESPResSo unter der Programmiererweiterung CUDA für Graphikkarten implementiert. Die extrem parallele Rechenkapazität erlaubt es dabei Systeme mit einigen tausend Teilchen effizient auf Nvidia Graphikkarten zu simulieren.

Um den Einfluss der Hydrodynamik zu bestimmen, wurde die hydrodynamische Funktion sowohl für die Gitter Boltzmann Methode als auch für die Stokes'sche Dynamik bestimmt. Die Gitter Boltzmann Methode zeigt mit üblichen Parametern aus der Literatur eine Abweichung auf kurzen Zeiten, verglichen mit den Ergebnissen der Stokes'schen Dynamik. Es wird davon ausgegangen das die Stokes'schen Dynamik Ergebnisse vertrauenswürdiger sind, da die Stokes'sche Dynamik im Stokes Regime arbeitet, im Gegensatz zur Gitter Boltzmann Methode. Das die Gitter Boltzmann Methode andere Ergebnisse produziert, lässt hinterfragen ob die mit typisch Parameter simulierten hydrodynamischen Wechselwirkung physikalisch sind.

Durch die Wahl von drastisch anderen Parametern konnten, entweder bei der mittleren quadratische Verschiebung (mean squared displacement) oder bei der hydrodynamischen Funktion, eine gute Übereinstimmung erziehlt werden. Im Falle von kleinen Viskositäten zeigt die hydrodynamische Funktion gute Übereinstimmung, die mittlere quadratische Verschiebung zeigt dagegen bis kurz vor die Langzeitdiffusion noch ballistisches Verhalten. Für große Viskositäten stimmt die mittlere quadratische Verschiebung gut überein, die hydrodynamische Funktion zeigt aber kaum hydrodynamische Wechselwirkungen. Es ist daher nicht klar, welche Parameter die korrekte Physik reproduzieren, beziehungsweise, ob es überhaupt Paramter gibt die sowohl die mittlere quadratische Verschiebung als auch die hydrodynamische Funktion angemessen reproduzieren.

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, 28. Januar, 2015

David Schwörer

Contents

1 Introduction

The study of soft matter is a subfield of condensed matter. Materials, where the energies of the dominant processes are of the same order as the thermal energy at room temperature, are in general considered as soft matter. This is because they can be deformed by thermal fluctuation or stresses. Examples for soft matter systems are polymers, liquids, gels, foams, many biological and colloidal systems [2]. Colloidal systems consist of particles (colloids) in the range 10 nm to 10 µm, which are suspended in a fluid [3]. There are many colloidal system in our everyday live. Milk, coffee, paint and blood are examples. In fact, there are many examples of food that are colloidal systems. The search to improve some aspects of food, for example the stability of foams like whipped cream, can be done with the help of computer simulation [4, 5, 6].

The static properties of colloidal systems are often dominated by electrostatic interactions. Additional to electrostatics, the repulsion of two colloids due to excluded volume can be important. If the dynamic properties of a colloidal system are studied, hydrodynamic interactions are important. All of these forces are easily simulated, making colloids a popular topic in computer simulations [7, 8, 9, 10, 11].

Extensible Simulation Package for Research on Soft matter (ESPResSo) is an open source software package with an emphasis on long range interactions [12]. Therefore this is a great tool to simulate soft matter, where electrostatic interactions and hydrodynamic interactions, both long ranged, are important. Additional to the mentioned long range interactions, it is intended to simulate coarse-grained soft matter systems. Coarse-grained means that instead of simulating single atoms or molecules, a coarser view is used, where whole groups of atoms or molecules are treated as a single particle, sometimes called a superatom. For example colloids can be treated as single particles, despite being made up of a large number of individual atoms. ESPResSo also supports thermal fluctuations, which are in soft matter of special importance, as the energies of the interactions are of similar order as the thermal energy. These particles are naturally surrounded by a solvent, whose dynamics can be of great importance. Representing the solvent by atoms or molecules is only practically possible on the nano scale. That is why the dynamics of a fluid is described at a continuum level. Depending on the assumptions about the fluid, it can be described with the Navier–Stokes equation or the Stokes equation. An even faster way to simulate the fluid is the use of a Langevin thermostat. This ignores advection and only the thermal fluctuations are simulated.

A method for simulating hydrodynamic interaction is the lattice-Boltzmann approach,

which solves the Navier–Stokes equation [13]. The lattice-Boltzmann approach solves the discretized Boltzmann equation on a grid. The colloids can be coupled with lattice-Boltzmann using a point coupling scheme [14]. Although lattice-Boltzmann is much faster than actually simulating the solvent atoms or molecules, this is computationally still quite costly. Thanks to the massively parallel computing power of modern graphic processing units (GPUs), this can be significantly sped up if these parts of the simulation are done on the GPU. The GPU implementation of lattice-Boltzmann in ESPResSo has been realized using the Compute Unified Device Architecture (CUDA). CUDA is framework to write software for Nvidia GPUs.

In this master's thesis a Stokesian Dynamics [1] implementation in ESPResSo will be described. Stokesian Dynamics describes the effective interaction on the embedded colloids mediated by a surrounding fluid. Stokesian Dynamics uses the analytical solution of two spheres in a fluid, and constructs, using these pair mobilities, a mobility matrix containing all the pair contributions. For the far field of the hydrodynamic interactions, the Rotne–Prager tensor is used.

In this thesis we focus on the short-time dynamics of the colloids. In order to check whether or how the difference in the short-time behavior affects e.g. crystallization of colloids, longer simulation runs are necessary. It was observed that negative eigenvalues in the Rotne–Prager tensor are possible if a cutoff is introduced. The negative eigenvalues can be avoided if the Ewald summation of the Rotne–Prager tensor is used. Using the lattice-Boltzmann method, for certain parameters the hydrodynamic function could be matched with the one obtained by Stokesian Dynamics. With these parameters, the short-time diffusion was not clearly visible, as the fluid's ballistic regime was too long. With other parameters, the short-time diffusion was as expected for a colloidal system, with these parameters the hydrodynamic functions showed only small contribution from the hydrodynamic interactions.

After the short introduction in this chapter, the next chapter will give an overview of the physical models used in this thesis. Chapter 3 will introduce the Stokesian Dynamics method and will also give a short introduction to lattice-Boltzmann in ESPResSo. The implementation of Stokesian Dynamics in ESPResSo will be shortly described in chapter 4, where ESPResSo and CUDA will also be introduced. The results of the comparison of lattice-Boltzmann and Stokesian Dynamics will be presented in chapter 5.

2 Models

In colloidal science, long-range interactions of the colloidal particles are of great importance. In sec. 2.1 standard models for short- and long-range interactions will be introduced. But colloidal particles are always surrounded by a solvent. The hydrodynamic interactions of particles mediated by the solvent is long-ranged and the evolution of the fluid can be described at the continuum level by the Navier-Stokes equation. The Navier-Stokes equation will be introduced in sec. 2.2. In general, colloidal particles undergo Brownian motion. In colloidal suspension the Reynolds number is very small. Additionally, the fluid can be assumed to be incompressible. This leads to the Stokes equation, which will be introduced in sec. 2.2.1.

2.1 Colloidal systems

Colloidal systems consist of mesoscopic particles ranging in size between 10 nm and 10 µm suspended in a fluid [15, 16]. Examples for such systems are blood, smoke or paint. In this thesis, only spherical colloids in a fluid suspension will be discussed. Several interactions of different physical origin are usually present in colloidal suspensions.

Excluded volume is modeled by a potential which prevents the different colloids from overlapping. This is often done with a Weeks-Chandler-Andersen (WCA) potential [17]

$$V_{\mathrm{WCA}}(r) = \begin{cases} 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) + \epsilon & r < 2^{\frac{1}{6}}\sigma \\ 0 & \text{else} \end{cases} \tag{2.1}$$

Where σ is the length scale of the potential and ϵ the energy scale. The soft WCA potential is an approximation to the hard sphere potential. There it is strictly forbidden for two spheres to overlap:

$$V_{\mathrm{HS}}(r) = \begin{cases} \infty & r < a \\ 0 & \text{else} \end{cases} \tag{2.2}$$

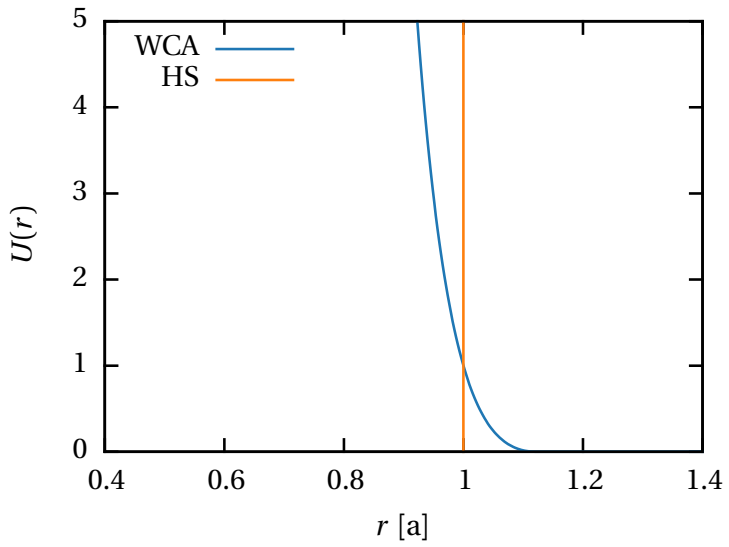

Figure 2.1: Plot of the WCA- and hard sphere (HS) potential.

Fig. 2.1 shows a sketch of both the WCA and the hard sphere potential. In MD simulations the WCA-potential is computational much more efficient than the hard sphere potential. In the case of molecular dynamics elastic collisions have to be implemented in order to handle hard spheres. This is computationally, however, not as efficient as a softer WCA potential. Therefore in MD simulations typically the WCA potential is used.

Electrostatic interactions often play an important role. The colloids studied in this thesis are charged. Additionally, there are always salt ions and/or polar solvent molecules present that screen the interaction between the colloids.

The Bjerrum length l_B is a length scale which is related to the interaction strength. It depends on the relative permittivity ϵ_r and the thermal energy $k_B T$. Additional the unity charge e and the absolute permittivity ϵ_0 are used to define the Bjerrum length:

$$l_B = \frac{e^2}{4\pi\epsilon_0\epsilon_r k_B T} \tag{2.3}$$

Another length scale is the Debye-Hückel screening length λ_D. This length scale measures how far the influence of the electrostatic interactions reaches. This length depends on the number densities n_i and the valencies z_i of the salt ions:

$$\lambda_D^{-1} = \kappa_D = \sqrt{4\pi l_B \sum_i n_i z_i^2} \tag{2.4}$$

Starting with the description via the Poisson-Boltzmann equation, it can be shown that the screened electrostatic interaction can be described sufficiently with the Yukawa potential [18, 19]

$$V(r) = l_B k_B T q_i q_j \frac{e^{-\kappa_D r}}{r} \tag{2.5}$$

where q_i is the effective charge of particle i in units of the elementary charge e. q_i is a dimensionless number.

The temperature is given as T and k_B is the Boltzmann constant. The thermodynamic properties of a Yukawa system can be characterized by two dimensionless parameters [20]

$$\kappa = \frac{r_{ws}}{\lambda_d} \tag{2.6}$$

$$\Gamma = \frac{q^2 e^2}{4\pi\epsilon_0 r_{ws} k_B T} = \frac{q^2 l_B}{r_{ws}} \tag{2.7}$$

with the Wigner–Seitz radius $r_{ws} = (\frac{3}{4}\pi n)^{1/3}$ as the unit of length. The number density n is the number of Yukawa particles per volume fraction.

Van der Waals forces are due to the induced dipole moments of the colloids. The dipole induced in a neutral atom by an external dipole decays as r^{-3}, the instantaneous dipole–induced dipole forces decay as r^{-6} and are therefore short ranged. This is an attractive force and therefore the main cause of aggregation in colloidal systems. However, aggregation is often not wanted, as it causes the colloids to sediment to the bottom of the solution. . One possibility to prevent aggregation, is to use charged colloidal particles. As they all have a charge of the same sign, they repel each other. This leads to a stabilization of the suspension. An alternative approach is to cover the particles with polymers, which leads to a loss in entropy if two colloids touch each other [21].

Colloidal systems are often used as a model system to study atomistic processes, yielding the advantage that the colloids are larger and can be observed using confocal microscopy. Interactions between these particles can be tuned by, for example, changing the salt concentration of the solvent. The hydrodynamic interactions between the colloids might change the dynamics of the system. This is a disadvantage, as an atomistic system has no solvent and this can change the dynamics. The timescales of crystallization are assumed to be much larger than the timescale on which hydrodynamic interactions are important. The influence of the hydrodynamics is therefore often neglected in simulations, as it is computationally cheaper. In such simulations, it is assumed that the dynamics of the solvent enter via an effective friction and can

Table 2.1: Various parameters for different radii a of the colloidal particles. τ_c is the sound propagation time, τ_B the relaxation time due to Brownian motion. τ_η the viscous relaxation time and τ_I the structure relaxation time. Values are given for typical parameters of an aqueous suspension of polystyrene spheres, with colloidal particle density $\rho_c \approx \rho_f$ and a fluid density of $\rho_f = 0.9982\ \mathrm{g/cm^3}$, a shear viscosity of $\eta = 0.01002\ \mathrm{g/cm\,s}$, fluid sound velocity $c_s = 1.48 \cdot 10^3\ \mathrm{m/s}$ and temperature $T = 293\ \mathrm{K}$. Description and table taken from [23].

a[nm]	1	10	100
τ_c[s]	$6.76 \cdot 10^{-13}$	$6.76 \cdot 10^{-12}$	$6.76 \cdot 10^{-11}$
τ_B[s]	$2.2 \cdot 10^{-11}$	$2.2 \cdot 10^{-9}$	$2.2 \cdot 10^{-7}$
$\overline{\Delta R^2(\tau_B)}^{1/2}/a$	$1.9 \cdot 10^{-3}$	$5.9 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$
τ_η[s]	$9.9 \cdot 10^{-11}$	$9.9 \cdot 10^{-9}$	$9.9 \cdot 10^{-7}$
τ_I [s]	$4.7 \cdot 10^{-6}$	$4.7 \cdot 10^{-3}$	4.7
τ_I/τ_B	$2.1 \cdot 10^5$	$2.1 \cdot 10^6$	$2.1 \cdot 10^7$

be scaled out. But recent results show that crystallization of colloids is affected by hydrodynamic interactions [18, 22]. Hence, investigating the short-time dynamics might explain the influence of hydrodynamics interactions on the ordering process. Table 2.1 gives an overview of the timescales of colloidal systems.

2.2 Hydrodynamics

Hydrodynamics is understood as a continuum description of fluids [24]. Therefore the molecular degrees of freedom are not taken into account. The interactions of the solvent atoms or molecules can be converted to an effective bulk and shear viscosity. Also, instead of using the velocities of the molecules themselves, a macroscopic fluid velocity $u(\vec{r})$ is used to describe the average of the molecular velocities at a location $\vec{r}$. This description lacks the thermal motion of the molecules, which can be included as thermal fluctuation. The Navier–Stokes equation [24, 15]

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u}\right) = -\nabla p + \eta\triangle\vec{u} + (\xi + \frac{1}{3}\eta)\nabla(\nabla\cdot\vec{u}) + \vec{f} \tag{2.8}$$

can be used to describe the motion of a fluid. The left side is the total time derivative of an arbitrary test volume of the moving fluid. The right hand side consist of the forces acting on the fluid. Thereby the first term is due to the pressure p of the fluid. The body

force $\vec{f}$ can be any external force acting on the fluid. η is the dynamic viscosity and ξ is often called bulk viscosity. This equation can be used for most kinds of uncharged fluids. Relativistic effects are not taken into account. It is only valid for sufficently large length scales, meaning of many particles, when the structure due to layering has decayed.

2.2.1 At low Reynolds numbers

For the systems discussed in this work, the fluid can be treated as incompressible, since aqueous solutions can be treated as incompressible in most cases. Exceptions exist, such as the study of sound waves or resonance effects, but such effects should not play a large role in colloidal systems. This means that if a specific amount of fluid flows into a test volume, the same amount has to flow out. The fluid is divergence free:

$$\nabla \cdot \vec{u} = 0 \tag{2.9}$$

Additionally, it can be assumed that the fluid relaxes faster than any other observed quantity. This results in a pseudo-static flow field. In other words, the time derivative on the left side of eq. (2.8) is negligible. The result is the time independent Stokes equation:

$$0 = -\nabla p + \eta \triangle \vec{u} + \vec{f} \tag{2.10}$$

This limit is known as Stokes flow or creeping-flow limit. Although eq. (2.10) looks much easier than eq. (2.8), it has to be solved with eq. (2.9) as a constraint. This coupling makes the Stokes equation challenging to solve. The analytical solution of the flow field $\vec{u}(\vec{r})$ around a single sphere moving with velocity $\vec{v}$ is [15]

$$\vec{u}(\vec{r}) = \left(\frac{3}{4}\frac{a}{r}(\mathbb{I} + \hat{r}\hat{r}) + \frac{1}{4}\frac{a^3}{r^3}(\mathbb{I} - 3\hat{r}\hat{r}) \right) \cdot \vec{v} \tag{2.11}$$

where a is the radius of the sphere and r is the distance from the center of the sphere. $\hat{r} = \vec{r}/r$ denotes the unit vector in the radial direction. The flow field is symmetric in the sense that the magnitude of the flow field does not depend on whether a point is in front of the sphere or behind the sphere. For higher Reynolds number this is not the case, as behind obstacles vortices can appear. The first term decays as $\frac{1}{r}$ and is dominant at larger distances. The second one is a quadrupole term and decays as $\frac{1}{r^3}$ and is short ranged. The drag $\vec{F}$ on a sphere with radius a is given by [15]

$$\vec{F} = 6\pi\eta a \vec{v} \tag{2.12}$$

2.2.2 Lubrication

Lubrication describes the fact that two solids can slide past each other if there is an extremly thin film of liquid between them. Extremly thin means here that the surface is much larger than thickness of the film. This is common, for example, in the kitchen when a wet chopping board slides over the countertop. One can also easily notice, that it is quite hard to pull the chopping board up, as it seems to stick. This is due to the thin distance between those two plates. It is for the fluid difficult to flow out, as the area through which it can flow out is very small [25].

In the context of this work, lubrication is if two colloids are very close. Then as in the case of the plates, the mobility in the direction of the vector connecting there centers is strongly reduced, while the motion orthogonal to that direction is still possible. This lubrication is not included in the Rotne–Prager tensor, and has to be added as a correction, if the colloids can get very close.

3 Methods

In this chapter the methods to describe hydrodynamic interactions used in this thesis will be discussed. We begin with Stokesian Dynamics in section 3.1. In the here discussed version, the thermalization is not straight forward. A scheme to realize the correct fluctuations is described in section 3.2. Although lattice-Boltzmann is much like Stokesian Dynamics, a method to calculate the hydrodynamic interactions, there are some important differences. They will be discussed in section 3.3 where the lattice-Boltzmann method will be briefly introduced.

In order to study the colloidal system, and in specific to compare and discuss the lattice-Boltzmann and Stokesian Dynamics method, we need observables to describe our system. The measured observables are introduced in section 3.4.

3.1 Stokesian Dynamics

The Stokesian Dynamics is an approach similar to molecular dynamics which was proposed by Brady and Bossis in ref. [1]. Ermak proposed earlier to use the Rotne–Prager tensor to include hydrodynamic interactions, but the lubrication correction was later proposed by Brady [26]. To take care of the hydrodynamic interactions, the analytical solution of two spheres in Stokes flow is taken. As the Stokes equation is linear, the analytical solution for the two sphere mobilities can be added to get the many body solution, consisting of pair interactions.

The hydrodynamic interaction is taken into account by computing a mobility matrix $\mathscr{M}$ which implicitly incorporates the hydrodynamic interaction. In the presence of external forces F^e the displacement Δx is obtained with [26]

$$\Delta x = \mathscr{M} \cdot F^e \Delta t + k_B T \nabla \cdot \mathscr{M} \Delta t + \sqrt{2 k_B T \mathscr{M} \Delta t} \cdot \Psi \tag{3.1}$$

or in explicit form

$$\Delta x_i = \sum_j \mathscr{M}_{ij} F_j^e \Delta t + \sum_j k_B T \frac{\partial \mathscr{M}_{ij}}{\partial x_j} \Delta t + \sum_j \sqrt{2 k_B T \mathscr{M} \Delta t}_{ij} \Psi_j \tag{3.2}$$

where $\vec{\Psi}$ is a Gaussian random variable with unit variance and zero mean. $k_B T$ is the thermal energy and Δt is the time step. The term containing $\vec{\Psi}$ must fulfill the fluctuation dissipation theorem. Therefore the square root of the mobility matrix is needed to

fulfill this relation. This fluctuation dissipation theorem is discussed in more detail in sec. 3.1.3. In colloidal systems, in addition to inter-particle forces and hydrodynamic interactions, thermal fluctuations have to be taken into account. This is incorporate as the random contribution in equation (3.1).

3.1.1 Far field

For the far field approximation the Rotne Prager–tensor or the Oseen tensor is commonly used. The Oseen tensor is given by [27]

$$\mathbf{M}_{ij} = \begin{cases} \frac{1}{8\pi\eta r}\left(\mathbb{I} + \hat{r}_{ij}\hat{r}_{ij}\right) & \text{if } i \neq j \\ \frac{1}{6\pi\eta a}\mathbb{I} & \text{if } i = j \end{cases} \tag{3.3}$$

where $\vec{r}_{ij}$ is the distance vector between particle i and j. Again $r_{ij} = |\vec{r}_{ij}|$ is the length and $\hat{r}_{ij} = \vec{r}_{ij}/r_{ij}$ is the direction of the vector. η is the viscosity and a the hydrodynamic radius. The self term is the solution of a sphere in Stokes flow [28]. The interaction between spheres is taken in the limit of two interacting point like spheres. Thereby only the $\frac{1}{r}$ term in equation (2.11) is taken. The force that the flow produces on another sphere gives the interparticle contribution of the Oseen tensor.

The Rotne–Prager tensor is given by [29]:

$$\mathbf{M}_{ij} = \begin{cases} \frac{1}{6\pi\eta a}\left(\frac{3a}{4r}\left(\mathbb{I} + \hat{r}_{ij}\hat{r}_{ij}\right) + \frac{1}{2}\left(\frac{a}{r}\right)^3\left(\mathbb{I} - 3\hat{r}_{ij}\hat{r}_{ij}\right)\right) & \text{if } i \neq j \\ \frac{1}{6\pi\eta a}\mathbb{I} & \text{if } i = j \end{cases} \tag{3.4}$$

The self interaction is again the solution of a single sphere in Stokes flow. The interparticle contribution includes higher moments resulting from the finite size of the spheres. The solution is obtained by computing the drag a sphere with radius a would feel in the flow field of a single sphere, see eq. (2.11). This neglects that the flow field is changed by the presence of the other sphere. The Rotne–Prager tensor includes the finite radius of the colloids, whereas the Oseen tensor is only valid for point like colloids.

An advantage of both the Rotne–Prager and the Oseen tensor is the vanishing divergence [30]:

$$\nabla \cdot \mathbf{M} = \frac{\partial \mathbf{M}_{ij}}{\partial x_j} = 0 \tag{3.5}$$

This reduces equation (3.1) to

$$\Delta x = \mathscr{M} \cdot F^e \Delta t + \sqrt{2 k_B T \mathscr{M} \Delta t} \cdot \Psi \tag{3.6}$$

3.1.2 Lubrication Correction

As two spherical particles get close to each other and finally touch, the mobility approaches 0 for some modes. The Oseen tensor is correct up to $O\left(\frac{1}{r}\right)$, while the Rotne–Prager tensor is up to $O\left(\frac{1}{r^3}\right)$. It is possible to take higher order corrections into account. In order to capture the vanishing eigenvalues of the mobility, an infinite number of terms is needed.

This interaction can be dealt with more efficiently in the resistance formulation. The resistance is the inverse of the mobility, and describes how much force is needed to move a colloid, while the mobility describes what the velocities are given a specific force. The analytical solution for two spheres in stokes flow which are close to contact is known [31]. One must take care to subtract the far field part, which was already added in the mobility formulation, to not double count the far field part in the case of the lubrication correction. The two sphere far field mobility is inverted.

$$\mathbf{M}^{2b,ff} = \frac{1}{6\pi\eta a}\begin{bmatrix} \mathbb{I} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbb{I} \end{bmatrix} \tag{3.7}$$

with

$$\mathbf{M}_{12} = \mathbf{M}_{21} = \left(\frac{3a}{4r}\left(\mathbb{I} + \hat{r}_{ij}\hat{r}_{ij}\right) + \frac{1}{2}\left(\frac{a}{r}\right)^3\left(\mathbb{I} - 3\hat{r}_{ij}\hat{r}_{ij}\right)\right) \tag{3.8}$$

Without loss of generality, $\hat{r}$ can be chosen as $\hat{e}_1$:

$$\mathbf{M}_{12} = \frac{3a}{4r}\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{1}{2}\left(\frac{a}{x}\right)^3\begin{pmatrix} -2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

$$= \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \beta \end{pmatrix} \tag{3.10}$$

with

$$\alpha = \frac{3a}{2r} - \left(\frac{a}{r}\right)^3 \tag{3.11}$$

$$\beta = \frac{3a}{4r} + \frac{1}{2}\left(\frac{a}{r}\right)^3 \tag{3.12}$$

using this abbreviation the two body far field mobility can be written as:

$$\mathbf{M}^{2b,ff} = \frac{1}{6\pi\eta a}\begin{bmatrix} & \mathbb{I} & \\ & & \end{bmatrix} \begin{bmatrix} \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \beta \end{pmatrix} \\ \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \beta \end{pmatrix} \end{bmatrix} \begin{bmatrix} & \\ \mathbb{I} & \end{bmatrix} \tag{3.13}$$

$$\left(\mathbf{M}^{2b,ff}\right)^{-1} = 6\pi\eta a \begin{bmatrix} -\frac{1}{\alpha^2-1} & 0 & 0 & \frac{\alpha}{\alpha^2-1} & 0 & 0 \\ 0 & -\frac{1}{\beta^2-1} & 0 & 0 & \frac{\beta}{\beta^2-1} & 0 \\ 0 & 0 & -\frac{1}{\beta^2-1} & 0 & 0 & \frac{\beta}{\beta^2-1} \\ \frac{\alpha}{\alpha^2-1} & 0 & 0 & -\frac{1}{\alpha^2-1} & 0 & 0 \\ 0 & \frac{\beta}{\beta^2-1} & 0 & 0 & -\frac{1}{\beta^2-1} & 0 \\ 0 & 0 & \frac{\beta}{\beta^2-1} & 0 & 0 & -\frac{1}{\beta^2-1} \end{bmatrix} \tag{3.14}$$

$$= 6\pi\eta a \begin{bmatrix} \alpha_s \hat{r}\hat{r} + \beta_s \mathbb{I} & \alpha_m \hat{r}\hat{r} + \beta_m \mathbb{I} \\ \alpha_m \hat{r}\hat{r} + \beta_m \mathbb{I} & \alpha_s \hat{r}\hat{r} + \beta_s \mathbb{I} \end{bmatrix} \tag{3.15}$$

with

$$\alpha_s = \frac{1}{1-\alpha^2} - \frac{1}{1-\beta^2} \tag{3.16}$$

$$= \frac{4\,r^6}{4\,r^6 - 9\,a^2\,r^4 + 12\,a^4\,r^2 - 4\,a^6} - \beta_s \tag{3.17}$$

$$\beta_s = \frac{1}{1-\beta^2} \tag{3.18}$$

$$= \frac{16\,r^6}{16\,r^6 - 9\,a^2\,r^4 - 12\,a^4\,r^2 - 4\,a^6} \tag{3.19}$$

$$\alpha_m = \frac{\alpha}{\alpha^2-1} - \frac{\beta}{\beta^2-1} \tag{3.20}$$

$$= -\frac{6\,a\,r^5 - 4\,a^3\,r^3}{4\,r^6 - 9\,a^2\,r^4 + 12\,a^4\,r^2 - 4\,a^6} - \beta_m \tag{3.21}$$

$$\beta_m = \frac{\beta}{\beta^2-1} \tag{3.22}$$

$$= -\frac{12\,a\,r^5 + 8\,a^3\,r^3}{16\,r^6 - 9\,a^2\,r^4 - 12\,a^4\,r^2 - 4\,a^6} \tag{3.23}$$

Equation (3.15) describes the resistance which is already incorporated in the far field. The lubrication correction $\mathbf{R}^{lc}$ is the analytical two body interaction in the near field

$\mathbf{R}^{lub}$ minus the part which was incorporated in the far field $\mathbf{R}^{2b,ff} = \left(\mathbf{M}^{2b,ff}\right)^{-1}$

$$\mathbf{R}^{lc} = \mathbf{R}^{lub} - \mathbf{R}^{2b,ff} \tag{3.24}$$

3.1.3 Thermalization

To simulate a system at a given temperature, thermal fluctuations have to be added. They have to be related to the dissipation. This can be derived by the fluctuation-dissipation theorem [32]:

$$\sigma^2 = 2\gamma k_B T \tag{3.25}$$

where σ is the strength of the fluctuating force, γ is the friction of the dissipation force and $k_B T$ the thermal energy. Or in words, the square of the strength of the fluctuation forces have to be proportional to the friction. For a given friction, it is therefore required that the strength of the fluctuations are proportional to the square root of the friction. To get from the strength of the forces to the actual forces, a Gaussian random variable Ψ is needed. Instead of the Brownian forces F^B, the Brownian displacements Δx^B are calculated, they are

$$\Delta x^B = \mathscr{M} F^B = \mathscr{M} \sqrt{\mathscr{R}} k_B T \Psi \tag{3.26}$$

$$= \mathscr{M} \sqrt{\mathscr{M}^{-1}} k_B T \Psi = \mathscr{M} \mathscr{M}^{-\frac{1}{2}} k_B T \Psi \tag{3.27}$$

$$= \sqrt{\mathscr{M}} k_B T \Psi \tag{3.28}$$

Therefore in the Stokesian Dynamics, a square root of the mobility matrix is needed. This can be done with the Cholesky decomposition of the matrix. The Cholesky decomposition is an $O(N^3)$ algorithm to compute a square root of a positive definite, symmetric matrix. In the case that the lubrication correction is added by inverting the matrices, this is obtained automatically as a byproduct of the matrix inversion. If the matrix inversion is avoided by using an iterative solver, it is not straight forward to get the correct thermal fluctuations without the $O(N^3)$ scaling [33].

3.1.4 Ewald summation

A cutoff of the slow decaying hydrodynamic interactions can lead to negative eigenvalues of the mobility matrix, as shown in section 5.1. Negative eigenvalues can cause serious problems. They violate e.g. thermodynamic laws, as they gain both potential and kinetic energy by moving to unfavorable positions in the energy landscape. Most simulations have a repulsive force to prevent particles from overlapping. In the case of negative eigenvalues, this force can result in the opposite of what it was intended for. The particles move closer together instead of moving further apart.

In addition to the problems regarding the physics of the system, this can also lead to the simulation crashing. In the case of a symmetric, positive definite matrix, special algorithms can be used for the matrix inversion. If the matrix has negative eigenvalues, this algorithm fails. Further more, the thermalization breaks down, as the Brownian displacements are proportional to the square root of the mobility matrix. If this matrix contains negative eigenvalues, there is no real matrix that fulfills the requirements of a root, and expanding the displacements to the complex space is certainly not physical.

For non-periodic systems it is straight forward to assert that no cutoff is taken if the sum is taken over all particles. In the case that we want to study a bulk system, we need periodic boundary conditions (PBCs) to simulate this efficiently. Without PBC most particles are on the wall. For example, for a box of $10 \times 10 \times 10$ particles, 48.8 % of the particles are on the outer shell. So huge particle numbers are needed, or else the boundary effects dominate. PBCs are used in basically all simulations for bulk properties. With PBCs it is impossible to directly sum over all hydrodynamic interacting particles without a cutoff, as there is an infinite number of periodic images. Even if a large cutoff is taken, negative eigenvalues can appear.

Using the Ewald summation of the Rotne–Prager tensor, as derived by Beenakker in ref. [34], the negatives eigenvalues vanish. The Ewald summation also uses a cutoff, but the terms in the Ewald sum decay fast, and therefore the error is negligible. The Ewald summation is constructed such that all terms contain an exponential function or a complementary error function as factor, which results in the fast decay.

$$\begin{aligned} 6\pi\eta\Delta\vec{x}_j = &\left(1 - 6\pi^{-\frac{1}{2}}\xi a + \frac{40}{4}\pi^{-\frac{1}{2}}\xi^3 a^3\right)\cdot\vec{F}_j \\ &+ \sum_{\substack{l \\ \vec{R}_{il_0}\neq\vec{R}_{jl}}}\sum_{i=0}^{N}\mathbf{M}^{\text{real}}(\vec{R}_{il_0} - \vec{R}_{jl})\cdot\vec{F}_i \\ &+ V^{-1}\sum_{\substack{\lambda \\ \vec{k}_\lambda\neq 0}}\sum_{i=1}^{N}\mathbf{M}^{\text{k}}(\vec{k}_\lambda)\cdot\vec{F}_i\cos\left(\vec{k}_\lambda\cdot(\vec{R}_i - \vec{R}_j)\right) \end{aligned} \tag{3.29}$$

with

$$\begin{aligned}\mathbf{M}^{\text{real}}(\vec{r}) = \mathbb{I}&\left(\left(\frac{3}{4}\frac{a}{r}+\frac{1}{2}\frac{a}{r}^{3}\right)\operatorname{erfc}(\xi r)\right.\\&+\left(4\xi^7a^3r^4+3\xi^3ar^2-20\xi^5a^3r^2-\frac{9}{2}\xi a+14\xi^3a^3+\xi\frac{a^3}{r^2}\right)\pi^{-\frac{1}{2}}\exp(-\xi^2r^2)\Big)\\&+\hat{r}\hat{r}\left(\left(\frac{3}{4}\frac{a}{r}-\frac{3}{2}\frac{a}{r}^{3}\right)\operatorname{erfc}(\xi r)\right.\\&\left.+\left(-4\xi^7a^3r^4-3\xi^3ar^2+16\xi^5a^3r^2+\frac{3}{2}\xi a-2\xi^3a^3-3\xi\frac{a^3}{r^2}\right)\pi^{-\frac{1}{2}}\exp(-\xi^2r^2)\right)\end{aligned} \tag{3.30}$$

$$\mathbf{M}^{\text{k}}(\vec{k}) = \left(1-\hat{k}\hat{k}\right)\left(a-\frac{1}{3}a^3k^2\right)\left(1+\frac{1}{4}\frac{k}{\xi}^{2}+\frac{1}{8}\frac{k}{\xi}^{4}\right)6\pi k^{-2}\exp\left(-\frac{1}{4}\frac{k}{\xi}^{2}\right) \tag{3.31}$$

The given real-space contribution was implemented as a dense matrix:

$$\begin{aligned}6\pi\eta\mathbf{M}^{\text{real}}_{ij} = &\left(1-6\pi^{-\frac{1}{2}}\xi a+\frac{40}{4}\pi^{-\frac{1}{2}}\xi^3a^3\right)\delta_{ij}\\&+\sum_{\substack{l\\R_{il_0}\neq R_{jl}}}\sum_{i=0}^{N}M^{\text{real}}(R_{il_0}-R_{jl})\end{aligned} \tag{3.32}$$

It would be straight forward to change the real space contribution to a sparse matrix. However, as the Fourier-space contribution is the time consuming part, this would not significantly speed up the simulation.

A direct summation of the Fourier-space yields an $O(N^2n_\lambda)$ scaling where n_λ is the number of Fourier-space contributions. If the optimal Ewald scaling of $O\left(N^{\frac{3}{2}}\right)$ [35]

wants to be achieved, the Fourier-part of equation (3.29) has to be rewritten:

$$6\pi\eta V\Delta\vec{x}_j^{\mathrm{k}} = \sum_{\substack{\lambda \\ \vec{k}_\lambda \neq 0}} \sum_{i=1}^{N} \mathbf{M}^{\mathrm{k}}\left(\vec{k}_\lambda\right) \cdot \vec{F}_i \cos\left(\vec{k}_\lambda \cdot (\vec{R}_i - \vec{R}_j)\right) \tag{3.33}$$

$$= \sum_{\substack{\lambda \\ \vec{k}_\lambda \neq 0}} \sum_{i=1}^{N} \mathbf{M}^{\mathrm{k}}\left(\vec{k}_\lambda\right) \cdot \vec{F}_i \left(\cos\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\cos\left(\vec{k}_\lambda \cdot \vec{R}_j\right) + \sin\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\sin\left(\vec{k}_\lambda \cdot \vec{R}_j\right)\right) \tag{3.34}$$

$$= \sum_{\substack{\lambda \\ \vec{k}_\lambda \neq 0}} \mathbf{M}^{\mathrm{k}}\left(\vec{k}_\lambda\right) \cdot \left[\left(\sum_{i=1}^{N} \vec{F}_i \cos\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\right)\cos\left(\vec{k}_\lambda \cdot \vec{R}_j\right) + \left(\sum_{i=1}^{N} \vec{F}_i \sin\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\right)\sin\left(\vec{k}_\lambda \cdot \vec{R}_j\right)\right] \tag{3.35}$$

$$= \sum_{\substack{\lambda \\ \vec{k}_\lambda \neq 0}} \left[\mathbf{M}^{\mathrm{k}}\left(\vec{k}_\lambda\right) \cdot \left(\sum_{i=1}^{N} \vec{F}_i \cos\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\right)\cos\left(\vec{k}_\lambda \cdot \vec{R}_j\right) + \mathbf{M}^{\mathrm{k}}\left(\vec{k}_\lambda\right) \cdot \left(\sum_{i=1}^{N} \vec{F}_i \sin\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\right)\sin\left(\vec{k}_\lambda \cdot \vec{R}_j\right)\right] \tag{3.36}$$

Now we can pull out the sum over the particles and put this in a separate computing kernel:

$$S_\lambda^{\cos} = \mathbf{M}^{\mathrm{k}}\left(\vec{k}_\lambda\right) \cdot \left(\sum_{i=1}^{N} \vec{F}_i \cos\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\right) \tag{3.37}$$

$$S_\lambda^{\sin} = \mathbf{M}^{\mathrm{k}}\left(\vec{k}_\lambda\right) \cdot \left(\sum_{i=1}^{N} \vec{F}_i \sin\left(\vec{k}_\lambda \cdot \vec{R}_i\right)\right) \tag{3.38}$$

Using this precomputed values, we can write the Fourier-space contribution in a computationally more efficient way:

$$6\pi\eta V\Delta\vec{x}_j^{\mathrm{k}} = \sum_{\substack{\lambda \\ \vec{k}_\lambda \neq 0}} S_\lambda^{\sin}\sin\left(\vec{k}_\lambda \cdot \vec{R}_i\right) + S_\lambda^{\cos}\cos\left(\vec{k}_\lambda \cdot \vec{R}_i\right) \tag{3.39}$$

With this the scaling is $O(N\lambda_{\max})$. This allows us to compute the matrix vector product without actually computing the matrix.

3.1.5 Avoiding the matrix inversion

In the Stokesian Dynamics the displacements are calculated using the far field contribution M^{ff} of the mobility, and a short ranged lubrication correction R^{nf} in the

resistance notation [1]. These two contribution to the mobility have to be added by summing both contribution in the resistance notation. This is comparable to electrical conductivity, where resistances of a circuit are additive.

The inverse of the total mobility is the total resistance $\mathscr{R}$ [1]

$$\mathscr{M} = \mathscr{R}^{-1} \tag{3.40}$$

$$\mathscr{R} = R^{nf} + R^{ff} = R^{nf} + \left(M^{ff}\right)^{-1} \tag{3.41}$$

$$\mathscr{M} = \mathscr{R}^{-1} = \left(R^{nf} + \left(M^{ff}\right)^{-1}\right)^{-1} \tag{3.42}$$

$$= \left(M^{ff}R^{nf} + \mathbb{I}\right)^{-1} M^{ff} \tag{3.43}$$

The equation (3.43) has the advantage that only one inversion is needed. As we want to avoid an analytic inversion, this has to be replaced by an iterative solver.

To solve the inversion iteratively, I implemented and tested different solvers, namely GMRes and BiCGStab. The Matrix $M^{ff}R^{nf} + \mathbb{I}$ is not positive definite and symmetric. Therefore the CG Algorithm cannot be used. I noticed that BiCGStab sometimes failed to solve the iterative systems. It is known that BiCGStab can fail to solve linear systems, which is known as breakdown [36]. GMRes performed well and succeeded in solving the iterative system more reliably.

Single floating point precision was often not enough to succeed, instead double floating point precision had to be used. Using the compile time switch `SD_USE_FLOAT` it is however possible to switch between double and single precision.

3.2 Matrix free thermalization of Stokesian Dynamics

The thermalization of the Stokesian Dynamics without matrix inversion has been a challenge. Reference [33] describes a possible way to do this. The authors note that this is quite slow. But as the scaling of this algorithm is better than $O(N^3)$, for large particle numbers, this approach outperforms the traditional approach with the full matrix inversion.

In this approach, instead of computing directly the Brownian displacements, the Brownian random forces are computed. The Brownian displacements have to be proportional to the square root of the total mobility matrix. As in the here described approach, the costly matrix inversion is avoided. This has the side effect that the mobility matrix is not known, and neither is the square root of the mobility matrix. The expectation value of the square of the Brownian forces F^B is proportional to the resistance matrix, and not like the displacements to the mobility matrix [33]:

$$\overline{\vec{F}^B(t)} = 0 \tag{3.44}$$

$$\overline{\vec{F}^B(t)\vec{F}^B(t')} = 2T\delta_{t,t'}\mathscr{R} \tag{3.45}$$

The over-line denotes the statistical average. Using eq. (3.41) this can be rewritten as

$$\overline{\vec{F}^B(t)\vec{F}^B(t')} = 2T\delta_{t,t'}\left(\left(\mathbf{M}^{ff}\right)^{-1} + \mathbf{R}^{nf}\right) \tag{3.46}$$

It is possible to split the random forces in a contribution from the near and one from the far field. This is the reason why the forces are used and not the displacements. To write the forces explicitly down, a Gaussian random variable Ψ with unit variance is used. The Brownian forces are given by [33]

$$\vec{F}^B(t) = \sqrt{2T\Delta t\mathscr{R}}\cdot\vec{\Psi} \tag{3.47}$$

$$\vec{F}^B(t) = \sqrt{2T\Delta t\mathbf{R}^{nf}}\cdot\vec{\Psi}^{nf} + \sqrt{2T\Delta t\mathbf{R}^{ff}}\cdot\vec{\Psi}^{ff} \tag{3.48}$$

If $\vec{\Psi}^{nf}$ and $\vec{\Psi}^{ff}$ are uncorrelated, then the cross terms vanish in the averages and eq. (3.45) is fulfilled. This allows us to find for both contributions separately a way to compute a square root of the resistance contribution. It would not be possible to do this in the mobility notation, as the resistances, but not the mobilities are additive.

In order to obtain the Brownian displacements, the computed random forces are taken and equation (3.6) is solved again iteratively for the Brownian displacements.

It is favorable to solve for the random and deterministic displacements separately, as for the deterministic part a good approximation for the inverse is given with the last computed displacements. This leads to a fast convergence of the iterative solver. The Brownian displacements have to be uncorrelated from time step to time step. It is not feasible to use the last displacement as the starting vector for the iterative solver, as this is only advantageous if the old solution is close to the new solution.

3.2.1 Far field

As mentioned, the Brownian force can be split into a contribution from the far field and one from the lubrication correction. Here a method is discussed to find the Brownian forces from the far field $F^{B,ff}$ without the need to actually compute the square root of the matrix. The forces have to fulfill:

$$\vec{F}^{B,ff}(t) = \sqrt{2T\Delta t\mathbf{R}^{ff}}\cdot\vec{\Psi}^{ff} \tag{3.49}$$

In order to avoid the explicit calculation of $\sqrt{\mathbf{R}^{ff}}$ a polynomial approximation is used. This has the advantage that the matrix $\mathbf{R}^{ff}$ or its inverse $\mathbf{M}^{ff}$ doesn't need to be known. It is sufficient to be able to compute its action on a given vector.

It would be an advantage to compute directly the approximation of $1/\sqrt{x} = x^{-\frac{1}{2}}$ to get directly the forces. In order to find a Taylor series of the function $x^{-\frac{1}{2}}$, the range of the eigenvalues needs to be known. Due to the singularity at $x = 0$, the convergence

radius r of the Taylor series at x_0 is limited to $r = x_0$. Why the eigenvalues of the matrix M^{ff} are needed can be shown if the matrix is decomposed into the sum of the normalized eigenvectors $\hat{v}_i$ and eigenvalues λ_i:

$$M^{ff} = \sum_i \lambda_i \hat{v}_i \hat{v}_i \tag{3.50}$$

where the tensor product is implied. The Taylor series of a function $f(x)$ at x_0 is

$$f(x) \approx \sum_j c_j (x - x_0)^j \tag{3.51}$$

with the coefficients c_j:

$$c_j = \frac{\partial^j}{\partial x^j} f(x) \bigg|_{x=x_0} \tag{3.52}$$

Plugging M^{ff} in (3.51) as x and then using the decomposition (3.50), this results in

$$f\left(M^{ff}\right) \approx \sum_j c_j \left(\sum_i (\lambda_i \hat{v}_i \hat{v}_i) - x_0 \mathbb{I} \right)^j \tag{3.53}$$

as different eigenvectors are orthogonal $\hat{v}_i \hat{v}_j = \delta_{ij}$, this is equal to

$$f\left(M^{ff}\right) \approx \sum_i \hat{v}_i \hat{v}_i \sum_j c_j (\lambda_i - x_0)^j \tag{3.54}$$

Having the Taylor series of the matrix rewritten like this it is obvious that the series has to converge for all eigenvalues, in order for the series of the matrix to converge. The Taylor series is not well suited due to the singular character of the inverse square root. Especially for eigenvalue close to 0, the series converges slowly.

Instead the Chebyshev polynomial is used. With Chebyshev polynomials every piecewise smooth and continuous function can be approximated in the range of $[-1 : 1]$ [37]. Again, as in the case of the Taylor series, the eigenvalues have to be in the specific range that the approximation converges. As not all eigenvalues necessarily are in this range, the function $x^{-\frac{1}{2}}$ has to be transformed so that the eigenvalues of the transformed function are in this range. To do so, the eigenvalues have to be known. To be specific, only the smallest and largest eigenvalue have to be computed. This is done with arpack-ng, a Fortran library, capable of computing the eigenvalues. It does not require a matrix or a function which it calls, instead it gives return codes. These codes tell the calling program what to do, e.g. to compute a matrix vector product, where the

vector is provided by arpack-ng. This makes it possible to have the matrix on the GPU and compute the Fourier-part of the matrix without having it stored as a matrix.

Arpack-ng can compute the smallest or largest eigenvalue, not only in magnitude, but also the largest or smallest real eigenvalue and the same for imaginary ones. For the use in Stokesian Dynamics, it is sufficient to compute only the smallest and largest in magnitude. At this point it should also be checked whether we have negative eigenvalues. The eigenvalues with the smallest and largest real value have been computed.

The eigenvalues of the far field mobility are not computed in every step, as it is expected that they do not change much between subsequent time steps. Also the exact values are not necessary, the range is a little larger then the true range, so that the approximation also converges if the range of the eigenvalues changes a bit. This has the disadvantage that for the approximation at the same accuracy a higher order is needed, but this is faster than to recompute the eigenvalues range often.

The Chebyshev polynomials T_i are defined recursively [38]:

$$T_0(x) = 1 \tag{3.55a}$$
$$T_1(x) = x \tag{3.55b}$$
$$T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x) \tag{3.55c}$$

To get the coefficients a_i for the function $f(z) = z^{-\frac{1}{2}}$ in the range $[\lambda_{\min} : \lambda_{\max}]$, the argument $z \to x$ of the function has to be linear transformed from $[\lambda_{\min} : \lambda_{\max}] \to [-1 : 1]$. This results in:

$$a_i = \frac{1}{n_i} \int_{-1}^{+1} \frac{T_i(x) f\left(\frac{2x}{\lambda_{\max}-\lambda_{\min}} - \frac{\lambda_{\max}+\lambda_{\min}}{2}\right)}{\sqrt{1-x^2}} \mathrm{d}x \tag{3.56}$$

$$n_i = \int_{-1}^{+1} \frac{T_i(x) T_i(x)}{\sqrt{1-x^2}} \mathrm{d}x = \begin{cases} \pi & i = 0 \\ \frac{\pi}{2} & \text{else} \end{cases} \tag{3.57}$$

This integral can be efficiently computed with the Chebyshev–Gauss quadrature [39]:

$$\int_{-1}^{+1} \frac{f(x)}{\sqrt{1-x^2}} \mathrm{d}x \approx \sum_{i=1}^{n} \frac{\pi}{n} f\left(\cos\left(\frac{2i-1}{2n}\pi\right)\right) \tag{3.58}$$

The approximation of the square root of the matrix is then given by:

$$\frac{1}{\sqrt{\mathbf{M}^{ff}}} = \sum_{i=0}^{\infty} a_i T_i\left(\mathbf{M}^{ff}\right) \tag{3.59}$$

With the Chebyshev polynomials:

$$T_0\left(\mathbf{M}^{ff}\right) = \mathbb{I} \tag{3.60}$$

$$T_1\left(\mathbf{M}^{ff}\right) = \frac{2}{\lambda_{\max}-\lambda_{\min}}\mathbf{M}^{ff} - \frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\mathbb{I} \tag{3.61}$$

$$T_{n+1}\left(\mathbf{M}^{ff}\right) = \left(\frac{4}{\lambda_{\max}-\lambda_{\min}}\mathbf{M}^{ff} - 2\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\mathbb{I}\right) T_n\left(\mathbf{M}^{ff}\right) - T_{n-1}\left(\mathbf{M}^{ff}\right) \tag{3.62}$$

Instead of summing to ∞ a cutoff N_c is introduced. The cutoff is determined when we compute the coefficients. Much more coefficients are computed as are used in the approximation. Then the sum of the absolute values of all computed coefficients is computed. It is then checked how many coefficients are needed to achieve the given accuracy. We can than express the random forces as a Chebyshev polynomial approximation:

$$F^{B,ff} \approx \sum_{i=0}^{N_c} a_i\, T_i(\mathbf{M}^{ff}) \cdot \Psi^{ff} \tag{3.63}$$

Instead of calculating the matrix-matrix product, only matrix-vector multiplications are used. To check whether the current approximation still fulfills the desired accuracy, the error can be approximated [33]:

$$E_{\text{cheby}} = \frac{F^{B,ff}\cdot\mathbf{M}^{ff}\cdot F^{B,ff} - \Psi^{ff}\cdot\Psi^{ff}}{\Psi^{ff}\cdot\Psi^{ff}} \tag{3.64}$$

E_{cheby} is a relative error, so it can be compared with the desired accuracy. If necessary, the eigenvalues and the Brownian forces are recomputed.

3.2.2 Near field

The approach used for the far field is not suitable for the near field, as some of the eigenvalues are zero if some particles are several colloid radii apart. In contrast, touching particles may have very large entries.

In the near field, the resistance matrix is directly composed of single contributions of the pair-resistances [1]:

$$\mathbf{R}^{nf} = \sum_{i<j}{}' \mathbf{R}^{nf}_{i,j} \tag{3.65}$$

The $'$ in the sum indicates that only close particles have to be summed over. Instead of computing the square root of the total near field resistance $\mathbf{R}^{nf}$, the square roots of the

pair resistances $\mathbf{R}_{i,j}^{nf}$ are computed. As this matrix is small and contains only 36 entries the Cholesky decomposition can be used to calculate the square root. As noted earlier, the Cholesky decomposition is an algorithm to find a square root of a positive definite and symmetric matrix, its scaling is $O(N^3)$ [40], but as we only have a small matrix, this is not a problem. This yields for the Brownian near field forces [33]

$$F^{B,nf} = \sum_{i<j}{}' \sqrt{\mathbf{R}_{i,j}^{nf}} \Psi_{ij}^{nf} \tag{3.66}$$

Ψ_{ij}^{nf} have to be uncorrelated for different pairs i, j.

3.2.3 Far field only

In the case that the lubrication is neglected, instead of the forces, directly the displacements can be calculated. This avoids the use of an iterative solver.

The implementation is straight forward, if in (3.56) instead of $f(z) = 1/\sqrt{z}$ the inverse $f(z) = \sqrt{z}$ is used, directly the displacements are obtained using (3.63).

To check whether the eigenvalues are still accurate enough for the approximation, the error can be estimated with

$$E_{\text{cheby}} = \left| \frac{\Psi \cdot \mathbf{M}^{ff} \cdot \Psi - \Delta x^B \cdot \Delta x^B}{\Psi \cdot \mathbf{M}^{ff} \cdot \Psi} \right| \tag{3.67}$$

So the advantage of a far field only implementation is not only due to the the fact that the near field does not have to be computed and that the matrix inversion can be avoided, but in addition we can directly compute the displacements instead of computing the Brownian forces first.

3.3 Lattice-Boltzmann

This is only a very short introduction to lattice-Boltzmann. There are many good review papers about lattice-Boltzmann, for example [41, 42, 43] and also many books [44, 45]. The lattice-Boltzmann method is a solver for the Navier–Stokes equations. Instead of modeling the continuous properties and directly trying to solve the Navier–Stokes equations, the Boltzmann equation is modeled. Not only space and time are discretized, but also velocities are discretized [46]. The different lattice-Boltzmann implementation are typically characterized by the number of spatial dimensions d and number of discrete velocities q per spatial node. This is typically abbreviated as DdQq. The model used in ESPResSo is D3Q19. The velocities are sketched in fig. 3.1. Lattice-Boltzmann simulates the density functions $f(\vec{r}, \vec{v}, t)$ for each velocity $\vec{v}$ at every node $\vec{r}$ at time t.

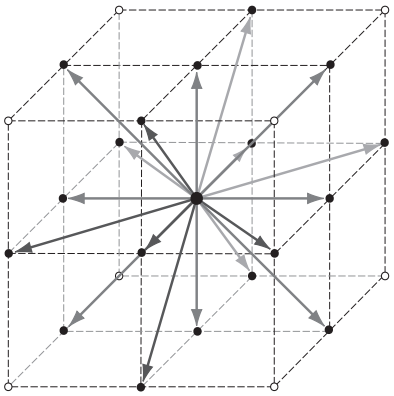

Figure 3.1: The 19 velocity vectors of the D3Q19 model. The discrete velocities are those pointing to the nearest and next nearest neighbors. Graphic taken from ref. [47].

The lattice-Boltzmann method consists of two steps. A streaming step and a relaxation step. In the streaming step the velocities are propagated according to there direction. Thereby the velocity modes are propagated to the corresponding nearest or next nearest neighbor. In the collision step, the velocities on a node relax a bit towards equilibrium

$$f(\vec{r},\vec{u},t+\Delta t)=(1-\lambda)f(\vec{r},\vec{u},t)+\lambda f^{eq}(\vec{r},\vec{u},t) \tag{3.68}$$

where λ describes how fast it relaxes as f^{eq} is the Maxwell–Boltzmann equilibrium distribution. This is a rather simple version of lattice-Boltzmann, and the implementation in ESPResSo uses a multi relaxation time scheme (MRT) where different modes relax with different relaxation times, which leads to a more realistic dynamics [48]. With help of the Chapman-Enskog expansion it can be shown that hydrodynamic interactions are correctly modeled for long-time and length scales [13].

3.3.1 Viscosity

The viscosity in lattice-Boltzmann is not a freely selectable parameter. The lattice-Boltzmann parameter which is closely linked to the dynamic viscosity is the relaxation time of the shear viscosity modes [13]:

$$\eta=\frac{\rho c_s^2\tau}{2}\frac{2+\lambda}{\lambda}=\frac{\rho\tau}{6a^2}\left(\frac{2}{\lambda}+1\right) \tag{3.69}$$

with

$$\lambda = \frac{\tau}{\tau_\lambda} \tag{3.70}$$

Thereby τ is the time step and τ_λ is the relaxation time of the viscous modes. This limits the viscosity for a given grid spacing and time step to

$$\eta > \frac{\rho\tau}{6a^2} \tag{3.71}$$

For the typical values discussed in this thesis, this is not a limitation. λ is a measure how much the viscous modes relax within a time step. This means that at least $1/\lambda$ time steps are needed such that the fluid is relaxed. Note that lattice-Boltzmann reproduces the Navier–Stokes equation for long-time and length scales [13].

A naive requirement for the timescales are therefore $t > \tau_\lambda$. Although this is certainly not sufficient for all cases, in most cases this should be a valid requirement. As momentum is propagated with the speed of sound, this means for the length scales l_t, that they should be larger than $L > c_s\tau_\lambda$ or

$$l_t > \frac{a}{\lambda} \tag{3.72}$$

Due to the point coupling scheme, if a dense suspension of particles is simulated, the distance between two particles might be on average about one grid length. It is not possible that $a > \frac{a}{\lambda}$, because $\frac{1}{\lambda}$ has to be smaller than one. One should take care that the box length L satisfies the inequality $L > \frac{a}{\lambda}$. If this is not the case, then the hydrodynamic interactions only barely depend on the distance between two particles, as the momentum mode is not relaxed when it interacts again with the particle that caused the momentum. If the characteristic feature of the decay as $1/r$ is not given but instead a constant interaction, it cannot be expected that the right results are obtained.

3.3.2 Speed of Sound

Another difference is that lattice-Boltzmann has a finite speed of sound $c_s = \sqrt{\frac{1}{3}}\frac{a}{\tau}$, where a is the grid spacing and τ the time step [13]. This means that the physical quantity c_s is defined by two non-physical properties. We are interested in the regime of a very high speed of sound, as this is the Stokes regime. In Stokesian Dynamics this is approximated by an infinite speed of sound. In lattice-Boltzmann this can be reproduced as the grid spacing goes to infinity or the time step goes to zero. The maximal grid spacing is limited by the distance of the particles. If several particles are in the same lattice-Boltzmann cell, it cannot be expected that the hydrodynamic interaction between those is well approximated. Therefore the maximal grid spacing a is limited by the particle density. The time step can be made smaller, this leads to an increase of the computation time, as more steps are needed to evolve the system by the same total time.

3.3.3 Particle coupling

There are two well studied ways to couple a MD simulation with lattice-Boltzmann [32]. The one proposed by Ladd, is to add a boundary which can move[49]. In the other approach, proposed by Ahlrichs and Dünweg, the particles exchange momentum with the fluid at the point where the particles are located [14]. This is also called the point coupling scheme. The friction γ controls the strength of the momentum transfer between the fluid and the particles. This produces an effective hydrodynamic radius

$$r_h = \frac{1}{6\pi\left(\frac{\eta}{\gamma}+\frac{g}{a}\right)} \tag{3.73}$$

where η is the viscosity of the fluid [32]. The numerical constant g depends on the box size and increases for larger boxes. For a cubic box of size $L = 16a$ the constant is $g \approx 0.036$ [50]. Together with the grid constant a, this is a numerical artefact arising from the grid of the fluid solver. This means that the maximal hydrodynamic radius is limited to

$$r_h \leq \frac{a}{6\pi g} \approx 1.5a \tag{3.74}$$

3.3.4 Over-damped motion

Stokesian Dynamics is typically simulated as an over-damped system. This means that particle velocities are neglected, and the forces result directly in displacements. This is a justified approximation, as in such systems the time τ_B, in which the velocity auto correlation function decays is much shorter than the time τ_I over which the configuration changes, see table 2.1. The timestep of the simulation is larger than the time over which the velocities change. As we are not interested in this short-time behavior of the system, this motion can be neglected. In lattice-Boltzmann the velocities are needed, as they are important for fluid coupling. In order to still simulate the right physical system, it has to be assured that the system reaches short-time diffusion, before the inter-particle interactions become dominant. Therefore it is not sufficient to choose a small time step, also the friction γ has to be large enough.

3.3.5 Lubrication Correction

It is known that the near field of the lattice-Boltzmann methods do not correctly mimic the short ranged hydrodynamic interaction. There are methods to add a near field correction [51]. This is similar to the lubrication correction in the Stokesian Dynamics. But in lattice-Boltzmann there is no analytical expression for the hydrodynamic interaction between two close particles. It is not possible to subtract from the near field the

already computed interaction. Although it is possible to add a lubrication correction in lattice-Boltzmann, there are still many implementations without this correction.

3.4 Oberservables

In order to compare the different methods, some observables are needed to measure the behavior of the system. I calculated the mean square displacement, its derivative and the time dependent self diffusion function. Further I calculated the structure factor, a static quantity describing the spatial distribution of the particles. Finally, I measured the hydrodynamic function. This quantity describes the influence of hydrodynamic interactions.

3.4.1 Mean Square Displacement

The mean square displacement (MSD) describes how far a particle has traveled on average within a certain time. The MSD $W(t)$

$$W(t) = \frac{1}{2d}\frac{1}{N}\overline{\sum_{i=1}^{N}\left(r_i(0) - r_i(t)\right)^2} \tag{3.75}$$

depends on the time t, the number of dimensions d and N is the number of particles [52, 23]. The MSD contains in a typical colloidal systems three parts:

- for short-times we have the ballistic regime. In this regime the MSD grows like t^2. This is only valid for times smaller than the decay time of the velocity autocorrelation function. In the case of an over-damped integrator, here Stokesian or Brownian Dynamics, this regime is not simulated. In the case of the Langevin thermostat or the lattice-Boltzmann, this is the timescale $\tau < \frac{1}{\gamma}$ if the friction γ is sufficiently high.

- for intermediate times we find a short-time diffusion. This is where the particles diffuse, as their momentum is negligible, and they wander due to Brownian motion. On these time and length scales, the interaction between the particles do not matter. This is because this is the timescale before they feel a force due to a neighboring particle, e.g. by colliding with it. In this regime the MSD is linear and it's expectation value is given by

  $$W(t) = D_s t$$

  D_s is the short-time diffusion constant. If the solution is dense and the particles have a long ballistic regime, the short-time diffusion might not exist. Then between the ballistic regime and the longtime diffusion, no further diffusive part

exists. This would be the case if the viscosity is small. The short-time diffusion is also called self-diffusion.

- for times larger than the timescale of the short-time diffusion, the long-time diffusion is reached. Again the MSD grows linearly:

$$W(t) = D_l\, t$$

On this timescale the interactions matter, because these are the timescales in which the particles see their neighbors and feel the forces constraining them, preventing them from diffusing freely. Due to this interaction the long-time diffusion is smaller than the short-time diffusion [23]

$$D_l < D_s$$

This inequality is also valid if an attractive interaction acts on the particles. In the case of no interaction, the short and long-time diffusion would be the same.

3.4.2 Time dependent self diffusion function

In order to examine the regimes of the mean square displacement, the time dependent self diffusion function is introduced. To be specific, there are two versions, one being the time derivative of $W(t)$ [23]

$$\mathscr{D}(t) = \frac{\mathrm{d}}{\mathrm{d}t} W(t) \tag{3.76}$$

and another where the mean square displacement is divided by the time [23]

$$D(t) = \frac{1}{t} W(t) \tag{3.77}$$

Both definitions give the same result in the long-time limit $D^l = \mathscr{D}(t \to \infty) = D(t \to \infty)$. As $\mathscr{D}(t)$ is a local quantity because it doesn't average over previous changes, this quantity will be used within this thesis.

3.4.3 Wave-vector dependent diffusion

The Wave-vector dependent diffusion (or k-dependent diffusion) is a measure for the time on which density fluctuations relax on a specific length scale.

The k-vector dependent diffusion is calculated using the structure factor $S(k)$. This is the Fourier transformation of the radial distribution function $g(r)$ [23]

$$S(k) = 1 + n \int \big(g(r) - 1\big) e^{i\vec{k}\cdot\vec{r}} \mathrm{d}r^3 \tag{3.78}$$

The radial distribution function measures the probability to find in a certain distance r other particles compared to the average density. In the case of an ideal gas $g(r)=1$ is constant, as the particles don't have any interaction. For perfect crystals the structure factor is a periodic function and for liquids this function converges to one, in the limit of large k values.

To calculate the k-vector dependent diffusion, the autocorrelation function (ACF) $S(k,t)$ of the structure factor is computed. $S(k,t)$ is the spatial Fourier transform of the van Hove function $G(r,t)$ [53]:

$$S(k,t)=\int G(r,t)\exp\left(-i\vec{k}\cdot\vec{r}\right)\mathrm{d}\vec{r} \tag{3.79}$$

The ACF $S(k,t)$ decays exponentially with an on k dependent time $\tau_{\mathrm{ACF}}(k)$. $\tau_{\mathrm{ACF}}(k)$ time gives finally the time dependent diffusion [23]

$$D(k)=-\frac{1}{q^2}\frac{\partial}{\partial t}\log(S(k,t))=-\frac{1}{q^2}\frac{1}{\tau_{\mathrm{ACF}}} \tag{3.80}$$

$D(k)$ depends also on the time, whether the derivative is taken in the short time or in the long-time diffusion limit. This is because the ACF of the structure factor is in general a multi-exponential decaying function. This is also shown in sec. 5.2.2. In order to measure the derivative, a fit with an exponential decaying function of the form $a\exp(-b\tau)$ with $b=1/\tau_{\mathrm{ACF}}$ the inverse of the decay time.

3.4.4 Hydrodynamic function

The hydrodynamic function $H(k)$ contains the effects of the hydrodynamic interaction on short-time scales [23, 54]. The Hydrodynamic function is given by [23]:

$$H(k)=\overline{\frac{1}{N}\sum_{i,j=1}^{N}\hat{k}\cdot\mathbf{D}_{ij}(\vec{R}^N)\cdot\hat{k}\exp(ik(R_i-R_j))} \tag{3.81}$$

This form has the disadvantage that the diffusion tensor needs to be known for a given configuration. Therefore in a lattice-Boltzmann simulation this can not be used to calculate the hydrodynamic function.

In the case of Brownian dynamics, $D_{ij}(\vec{R}^N)$ is a diagonal matrix

$$D_{ij}(\vec{R}^N)=\delta_{ij}D\mathbb{I} \tag{3.82}$$

putting this in equation (3.81) results in a constant function $H(k)$, as the direction of $\hat{k}$ does not matter because $D_{ij}\propto\mathbb{I}$ and r_i-r_j is always zero if the diffusion tensor D_{ij}

does not vanish. Although in the case of a Langevin thermostat the diffusion tensor is not explicitly known, the hydrodynamic function is a constant.

Another way to compute $H(k)$ is using the k-vector dependent diffusion [23]

$$H(k) = D(k)S(k) \tag{3.83}$$

This has the advantage that the diffusion matrix **D** does not need to be known. Only in the case of Brownian Dynamics or Stokesian Dynamics is the short-time diffusion matrix known. In order to compare the hydrodynamic interactions of lattice-Boltzmann and Stokesian Dynamics it is necessary to use this formulation.

For low k values the hydrodynamic function describes the collective behavior. The longtime collective diffusion can be measured e.g. by macroscopic diffusion experiments [23].

On the other hand, for high k values the single particle dynamics is visible. $H(k)$ converges in this regime to the single particle diffusion:

$$\lim_{k\to\infty} H(k) = D \tag{3.84}$$

Therefore it is also common to define the hydrodynamic function $\tilde{H}(k)$ normalized by the single particle diffusion:

$$\tilde{H}(k) = \frac{H(k)}{D} = \frac{H(k)}{H(k_\infty)} \tag{3.85}$$

with k_∞ a sufficiently large Fourier-space vector.

If the normalized hydrodynamic function $\tilde{H}(k)$ is larger than 1, the dynamics on the length scale corresponding to this k-vector is accelerated by hydrodynamics. If it is lower than 1, the dynamics with hydrodynamic interactions is retarded compared to the case without hydrodynamic interactions. As this makes it easier to compare the effect of hydrodynamic interaction between different systems, this is can be advantageous. Due to the normalization, the function lacks the information of the short-time diffusion, as the function always converges to 1 for sufficiently large k-vectors.

4 Implementation

The target of the implementation was to simulate larger numbers of particles, than it would be possible with the matrix inversion based implementation described by Brady [1]. Instead of the P3M-like approach described by Sierou and Brady, an Ewald summation was used [55]. This was chosen due to the limited time of a master's thesis. The thermalization was done in the way proposed by Banchio et al. [33]. The implementation is written in CUDA, so that it can use the massive parallel computing power of modern graphic cards. First CUDA and some features of graphic cards in general will be described in section 4.1. In the section 4.2 the simulation package ESPResSo will be introduced. The implementation of Stokesian Dynamics in ESPResSo using CUDA will be described in section 4.3.

4.1 CUDA

The Compute Unified Device Architecture (CUDA) is Nvidia's set of software tools to make it easier to write general purpose code for graphic cards [46]. It is an extension to the C/C++ language with specific features for graphic cards.

The idea of using GPUs not only for computing images but for arbitrary calculations is not new. In the beginning of general–purpose computing on graphics processing units (GPGPU), the GPU had to be used as if it would actually calculate an image [56]. Thereby only the instruction for image manipulation supported by the GPU could be used for solving other numerical equations. In the mean time this has much improved. It is now possible to write C/C++ or Fortran code using CUDA.

Another way is OpenACC or OpenMP. Both standards define pragmas, enabling the compiler to compile code for an accelerator like the GPU. This has the advantage that only a single code has to be maintained which can run one different hardware. Also some of the optimizations of the code for running on a GPU can also speed up the execution on a central processing unit (CPU). OpenMP supports accelerators like GPUs starting with version 4.0. At the time of the writing of this thesis, the GNU compiler GCC is still working on the support of accelerators [57]. It supports OpenMP 4.0 — except the support for GPUs. The clang compiler of the LLVM suite is still further behind in the support of OpenMP [58, 59].

4.1.1 Kernels

In CUDA the functions or subroutines running on the GPU are called kernels. These kernels are called and configured from the CPU, which is also called host. Typically many kernels are executed simultaneously to use the parallel computing power of GPUs. They are grouped in blocks and the blocks in a grid. Each thread executes the same kernel. Each thread has some variables predefined. `threadIdx` is a predefined variable. It is a 3 component vector and assigns each thread within a block a unique number. Additional `blockIdx` assigns each block a unique 3-component vector within the grid. Thereby every thread can be easily assigned its part of the computational task.

A block is internally grouped into warps, consisting of 32 threads. Within a thread the threads are all executed synchronously in parallel. If a block consists of more than one warp, synchronous execution is not guaranteed. Due to the fact that all threads within a warp are executed in parallel, similar to a vector processor, it is disadvantageous to have conditional statements. This only matters if they evaluate differently for different threads in a warp. In such a case the different branches have to execute sequentially while the other threads are waiting.

In order to execute a kernel on the GPU, some parameters have to be set. For example the number of threads or blocks have to be set. It is possible to change the block size, depending on the current problem size. This is important for optimal performance. Additional to the block and grid size, the cache size can be configured or the size of the shared memory per block can be configured. CUDA supports streams. With streams, different kernels can be executed in parallel, if they are in different streams. This can also be configured at the kernel call time. If two kernels are in the same stream, they are executed serialized. This is important if one kernel depends on the data of another kernel.

4.1.2 Hardware features

In order to achieve good performance on a GPU, it is necessary to take care of the specific properties of graphic cards. GPUs are designed for maximal throughput. In contrast, CPUs are designed for minimal latency. As shown in figure 4.1 a GPU devotes less transistors to caches and control units than a CPU. Therefore different cores cannot execute independent instructions. A streaming multiprocessor (SMX) is responsible for controlling the cores. The multiprocessor is also responsible for the registers. The registers are not associated with a single core but with the multiprocessor. This is one of the reasons why it is fast to switch between threads [61]. It is favorable to switch between threads in the case that the current threads is waiting e.g. for data from global memory. With this mechanism, the computational resources can be used quite efficiently, even though there is not much cache. While on a CPU the prefetcher tries to

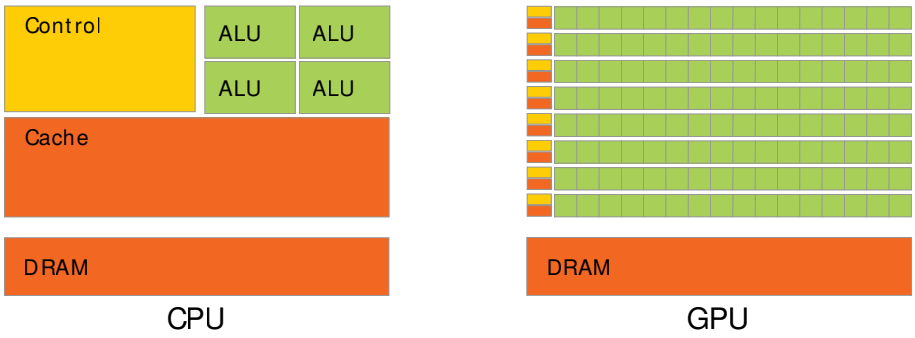

Figure 4.1: Schematic comparison of CPUs and GPUs. GPUs devote more transistors to calculations. A CPU has more cache per core. Also not every core on a GPU has its own control unit. All warps within a thread have to execute the same instruction or idle. Graphic taken from ref. [60].

estimate what data needs to be read next, on a GPU this is not needed. This only works if there are enough threads that can be computed while others are waiting for data from global memory.

Another important point is the way memory is read from global memory. If data is read from global memory, it is, by default, cached. A cache line is 128 bytes long. That means that when ever something is read from global memory, the whole cache line is read. Therefore, memory accesses should be constructed such, that the whole cache line is used. This in general achieved by letting consecutive threads read consecutive data. As cache lines are always aligned to 128 bytes, the data in memory should also be aligned. Alignment means that the the address p_1 of the first element which is read, is a multiple of 128, i.e. $p_1 \mod 128 = 0$. If this is not the case, than two cache lines have to be read. This can slow down the execution of the program tremendously, if the execution time is limited by the speed of memory transaction.

4.2 ESPResSo

On the project home-page of ESPResSo[1] the package is described as

> [...] a highly versatile software package for performing and analyzing scientific Molecular Dynamics many-particle simulations of coarse-grained atomistic or bead-spring models as they are used in soft-matter research in physics, chemistry and molecular biology. It can be used to simulate systems such as polymers, liquid crystals, colloids, ferrofluids and biological systems, for example DNA and lipid membranes.

[1] `http://espressomd.org`

> ESPResSo is free, open-source software published under the GNU General Public License (GPL). It is parallelized and can be employed on desktop machines, convenience clusters as well as on supercomputers with hundreds of CPUs. The parallel code is controlled via the scripting language Tcl, which gives the software its great flexibility.
>
> ESPResSo is used in scientific working groups all over the world both as a production platform as well as a research platform for developing new algorithms and methods for coarse-grained simulations. It is mainly developed at the Institute for Computational Physics of the University of Stuttgart, but has contributors from all over the world.

The fact that ESPResSo is licensed under the GPL has the advantage that every user can read the source code and understand how the software or the algorithm works. As earlier Stokesian Dynamics implementations were not published under an open license, for example [33, 55, 1], it has not been possible to improve the existing codes. There are open source implementation of hydrodynamic interactions, for example one by Kopp and Höfling [62]. But this implementation approximates the lubrication correction by a first order Taylor approximation. As the eigenvalues of the lubrication correction are often larger than one, this approximation would not converge even if a higher order approximation would be used. Also the Ewald summation, which is needed for periodic boundary conditions, is missing.

As the target was to implement this in ESPResSo, the implementations was written from scratch. As this code is now freely available, other can improve this code rather than solve challenges that have been already solved.

Other useful features are the large number of implemented observables, and the correlator. The correlator is especially useful, as this is needed to e.g. calculate the hydrodynamic function or the mean square displacement.

4.2.1 GPU support

Due to the vast computational power of graphic cards, some algorithms in ESPResSo have been ported to CUDA. Examples are the lattice-Boltzmann, P^3M and MMM1D [63]. In addition the electrokinetics solver has been only implemented for the GPU [46]. Especially for grid based application, the GPU can give a huge speedup. The GPU implementation run on a single Nvidia Tesla M2050 is about 50 times faster than the CPU implementation run on two Intel Xeon E5620 quad core processors.

4.2.2 Lattice-Boltzmann

The lattice-Boltzmann implementation in ESPResSo uses a point coupling scheme. The particle exchanges momentum proportional to the difference in the velocity of the fluid compared to the particle's velocity and a friction constant. Thereby the fluid is extrapolated considering the surrounding lattice nodes. Also the force is smeared onto the grid, using a two or three point coupling scheme.

The lattice-Boltzmann code in ESPResSo does not have a lubrication correction. As the particle coupling is done with point coupling, the method described in ref. [51] is not applicable.

4.3 Stokesian Dynamics implementation

This section will introduce the implementation of Stokesian Dynamics in ESPResSo. Figure 4.2 gives an overview over the most important functions, and the order they are called. Some of the function will be introduced in sec. 4.3.1. The way the matrices are stored in memory will be shown in sec. 4.3.2. The different versions of the code, which can also influence the functions called, are listed in sec. 4.3.3. Some timings and a rough approximation of the scaling are shown in sec. 4.3.4.

4.3.1 Functions

`integrate_sd()`

This is a replacement for the existing `integrate_vv()` function. The original velocity verlet integrator had four separate function calls, to do the different parts of the position and velocity updates. Therefore this function has been replaced by a separate function, as some parts are not required any more in the case of Stokesian Dynamics.

The function is mainly responsible for calling the function to calculate the forces and to call `propagate_pos_sd()`

`sd_compute_mobility_matrix_real_short()`

This function implements the real part of the Ewald summation, as it is described in eq. (3.32). For every three lines of the matrix, a thread is responsible. In the beginning, the threads read the positions for the particle of the warp. Later the other positions are read coalesced and cached in shared memory. The dense matrix, which is the output of this function, is written coalesced. In the matrix every line is 128 byte aligned. The function requires that the cutoff is not larger than $L/2$, where L is the box length. This way it is sufficient to loop over every particle only once.

- integrate_sd() *
  - propagate_pos_sd()
    - propagate_pos_sd_cuda()
      - sd_compute_displacement()
        - sd_compute_mobility_matrix_real_short() *
        - sd_compute_mobility_matrix_wave_cpu() *
        - sd_compute_mobility_sines() *
        - sd_compute_resistance_matrix_sparse() *
        - sd_iterative_solver()
          - sd_gmres_solver() *
        - curandGenerateNormalReal() *
        - sd_compute_brownian_force_nearfield()
        - sd_compute_brownian_force_farfield() *
          - calculate_maxmin_eigenvalues()
          - calculate_chebyshev_coefficents()
        - sd_iterative_solver()
          - sd_gmres_solver() *
      - sd_real_integrate

Figure 4.2: Function tree of the Stokesian Dynamics implementation in ESPResSo. The indention means that the indented function is called by the one in the level above. The function marked with * are later described in more detail.

If the feature SD_NOT_PERIODIC is enabled, the Ewald summation is not used. Instead of sd_compute_mobility_matrix_real_short() the function sd_compute_mobility_matrix() is called. In this case the normal Rotne–Prager tensor is used to construct the mobility matrix without any cutoff.

sd_compute_mobility_matrix_wave_cpu()

The $\vec{k}_\lambda$ and corresponding matrices $\mathbf{M}^k(\vec{k}_\lambda)$ as defined in eq. (3.31) are calculated. First the number of needed vectors is calculated. Additionally the required memory on the GPU is allocated, and the computed vectors and matrices are copied to the GPU.

`sd_compute_mobility_sines()`

In this function $\sin\left(\vec{k}_\lambda \cdot \vec{R}_i\right)$ and $\cos\left(\vec{k}_\lambda \cdot \vec{R}_i\right)$ are computed. They are later needed to compute the Fourier space contribution to the matrix vector product.

`sd_compute_resistance_matrix_sparse()`

This function computes the lubrication correction described in equation (3.24). A naive implementation would implement the loop to find the interacting particles in this function:

```
1   int i=threadIdx.x+blockDim.x*blockIdx.x;
2   for (int j=0;j<N;j++){
3     real dist2=0;
4     for (int l=0;l<3;l++)
5       dist2+=SQR(pos[i*3+l]-pos[i*3+l]);
6     if (dist2 < cutoff2){
7       /*
8        *  calculate lubrication
9        */
10    }
11  }
```

This implementation would be fine to run on a CPU. On a GPU the problem is that the code would branch. As the cutoff is of the length of the particle radius, most particles are not interacting. Therefore most times the branch is not taken. On a GPU, if any thread within a warp takes this branch, all others have to wait for this branch to finish. As most of the computation is done in this loop, most of the time only a single core would be computing, while the rest is waiting. In such an implementation, the massively parallel power of the GPU is not used and it is slower than a CPU. The here chosen solution, is to find the particles that are close to each other first, and store them in a list. They are stored in `col_idx` of the matrix struct. Then instead of looping over all particles, it is only necessary to loop over the particles which are in the range.

To find the interacting particles can either be done in an $O(N^2)$ loop or with the use of a bucket sort. Both cases are favorable compared to the naive implementation. The $O(N^2)$ version has still the problem that the branch to write out the positions of interacting particles is not taken by all threads, but as this is much faster than actually computing the lubrication correction, this is still worth doing.

`sd_gmres_solver()`

The GMRes solver has been implemented as described by Meister in ref. [36].

`curandGenerateNormalReal()`

This is a function of the cuRAND library. The advantage compared to the in ESPResSo existing random number generator is that the data is directly on the GPU and doesn't need to be copied to the GPU. The type of the random number generator is the default, which is an implementation of the XORWOW algorithm [64].

As it is possible to set an offset for the random number generator, the state of the random number generator is defined by the offset and the initial seed. This is important for checkpointing the simulation run.

`sd_compute_brownian_force_farfield()`

This function calculates the approximation of the far field contribution of the Brownian forces based on the old eigenvalues of the mobility far field. If the thereby achieved approximation is not sufficient, the functions is called again and calls first `calculate_maxmin_eigenvalues()` and then `calculate_chebyshev_coefficents()`. The first one uses the Fortran library arpack-ng to calculate the eigenvalues of the far field mobility matrix. The function `calculate_chebyshev_coefficients()` calculates the Chebyshev coefficients a_i for the approximation in equation (3.63). If the far field only version of the code, `SD_FF_ONLY`, is used, than directly the displacements instead of the forces are calculated.

4.3.2 Matrix Memory Layout

In order to simplify matrix operations, all properties of a matrix have been put in a struct:

```
1  struct matrix{
2  public:
3    real * data;
4    int  * col_idx;
5    int  * row_l;
6    wavepart * wavespace;
7    bool is_sparse;
8    int size;
9    int ldd;
10   int ldd_short;
```

```
11    matrix();
12    ~matrix();
13    void _free_wavespace();
14  private:
15    matrix(matrix & other); // disable copy ↩
          operator
16  };
```

`data` is the pointer to the data in the matrix. In the case it is saved as dense matrix, the elements within a row are stored one after another in memory. The beginning of a row was always 128 byte aligned. All the arrays are stored on the GPU, and should therefore not be dereferenced on the host.

`size` gives the number of rows, and in the case of a dense matrix the number of elements per row. `ldd` is the length a row in memory. This is the size rounded to the next integer divisible by 32.

If a sparse matrix is stored, a scheme similar to ELLPACK-R as described in [65] is used. The scheme of this storage format is shortly explained in fig. 4.3. As the matrix

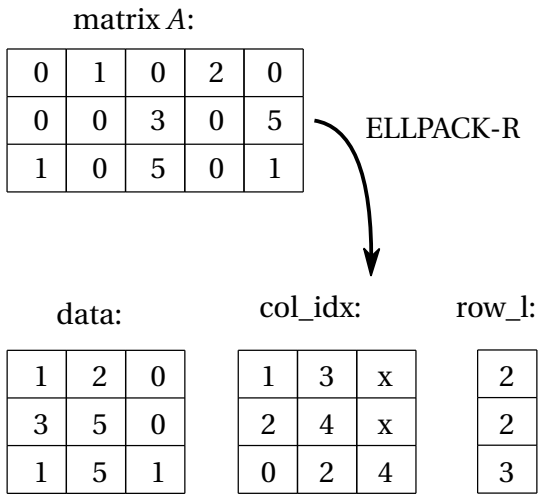

Figure 4.3: The ELLPACK-R Format. This format is optimized for matrix vector multiplications on GPUs [65].

consists of symmetric 3×3 blocks, instead of saving the actual row length for each row, only the length divided by three is saved for every three rows. The same applies for the column index, here only one index per block is saved.

As the Ewald summation of the Rotne–Prager tensor is used for periodic systems, the matrix has, in addition to the real space contribution, a Fourier space contribution. As

it saves time to not add the Fourier space to the matrix, the Fourier space contribution is also stored as an extension to the matrix struct. This extension is again a struct:

```
1  struct wavepart{
2  public:
3    real * vecs;
4    real * matrices;
5    real * cosines;
6    real * sines;
7    int num;
8    int max;
9    int & ldd_short;
10   wavepart(int _ldd_short);
11   ~wavepart();
12 private:
13   wavepart(wavepart & other);
14 };
```

All `num` $\vec{k}$ vectors are stored in the pointer `vecs`. The symmetric matrices $\mathbf{M}^k$ are stored in `matrices`. Thereby only the 6 unique components are stored. For every matrix first the diagonal entries and then the off-diagonal entries $\mathbf{M}^k_{12}$, $\mathbf{M}^k_{13}$ and $\mathbf{M}^k_{23}$. `sines` and `cosines` contains the pre-computed values $\sin(\vec{k}\cdot\vec{r}_i)$ respectively $\cos(\vec{k}\cdot\vec{r}_i)$. The fast index is the particle index and the Fourier space index the slow one. This means that the sines for the same k vector are directly one after another in memory. `ldd_short` is the rounded number of particles.

4.3.3 Versions of the code

The code is able to switch between configurations at compile time. It is for example possible to switch between single and double precision for the GPU code. Single precision means floating point numbers of 32 bit length, and double precision means floating point numbers of 64 bit length. The 32 bit numbers are also called single or floats. Nvidia's consumer GPUs, like the GeForce series, have notably fewer double floating units as the high performance computing (HPC) GPUs [60]. Therefore it is, especially if the software is run on GeForce cards, useful to use the single floating point precision. The problem is, however, that the accuracy of single floating point is sometimes not sufficient. Examples are that the eigenvalues of the mobility matrix $\mathbf{M}^{ff}$ cannot be accurately determined or the iterative solver fails. If such problems occur double precision can help to stabilize the simulation. With the feature `SD_USE_FLOAT` the code

can be told to use floats instead of double. The default is double precision.

It is also possible to simulate periodic and non-periodic systems with the current Stokesian Dynamics implementation. Depending on which version is used different kernels are used for the calculation of the mobility in the far field. The default is to use periodic boundary conditions and the Ewald summation. This can be switched with the feature `SD_NOT_PERIODIC` to use no periodicity in the mobility calculation. Note that the force calculations are not influenced by this feature.

Another compile time switch disables the lubrication correction. In this case the iterative solver has not to be used. This can significantly speed up the simulation, and is therefore useful if lubrication does not play a big role for the simulated spheres. If there are no spheres within the cutoff radius of lubrication correction, than the iterative solver is not used anyway. If there are sometimes some spheres which would have even a tiny lubrication correction, this can slow down the code significantly. The radial distribution function can be used to estimate the influence of the lubrication correction. The lubrication correction can be disabled with the feature `SD_FF_ONLY`.

4.3.4 Timings and Scaling

Figure 4.4 shows the timings of some test runs for various configurations. These timings show only a rough estimate of the speed of the code. The particular system and how fast the eigenvalues change can influence the speed strongly, especially in the case of the lubrication correction. If all particles are further away from each other, than the cutoff of the lubrication correction, than it is only slightly slower than the version without lubrication correction. The version with lubrication and with periodic boundaries is so slow, that only for small particle numbers the duration was acceptable. For this case the GPU version is certainly not useful, as most kernels use only one thread per particle. In the case of 128 particles, this means that many multi processors are idling. In this case, latency hiding is not possible, and a CPU would probably be faster.

The simulations have been run without lubrication and without Ewald summation for both single and double precision. It is noticeable that the double precision implementation is only slightly slower than the single precision one. This is probably because the timings are limited by the latency on the GPU for read or write transactions from global memory, but also by transfer times from the GPU to the host. The transfer times are for small data packages almost independent of the data size. In the case of periodic boundary conditions, double precision is more than twice as slow as single precision. Hence, the simulation time is not dominated by latencies. Here, also the memory bandwidth or the number of double precision floating point units is limiting the speed.

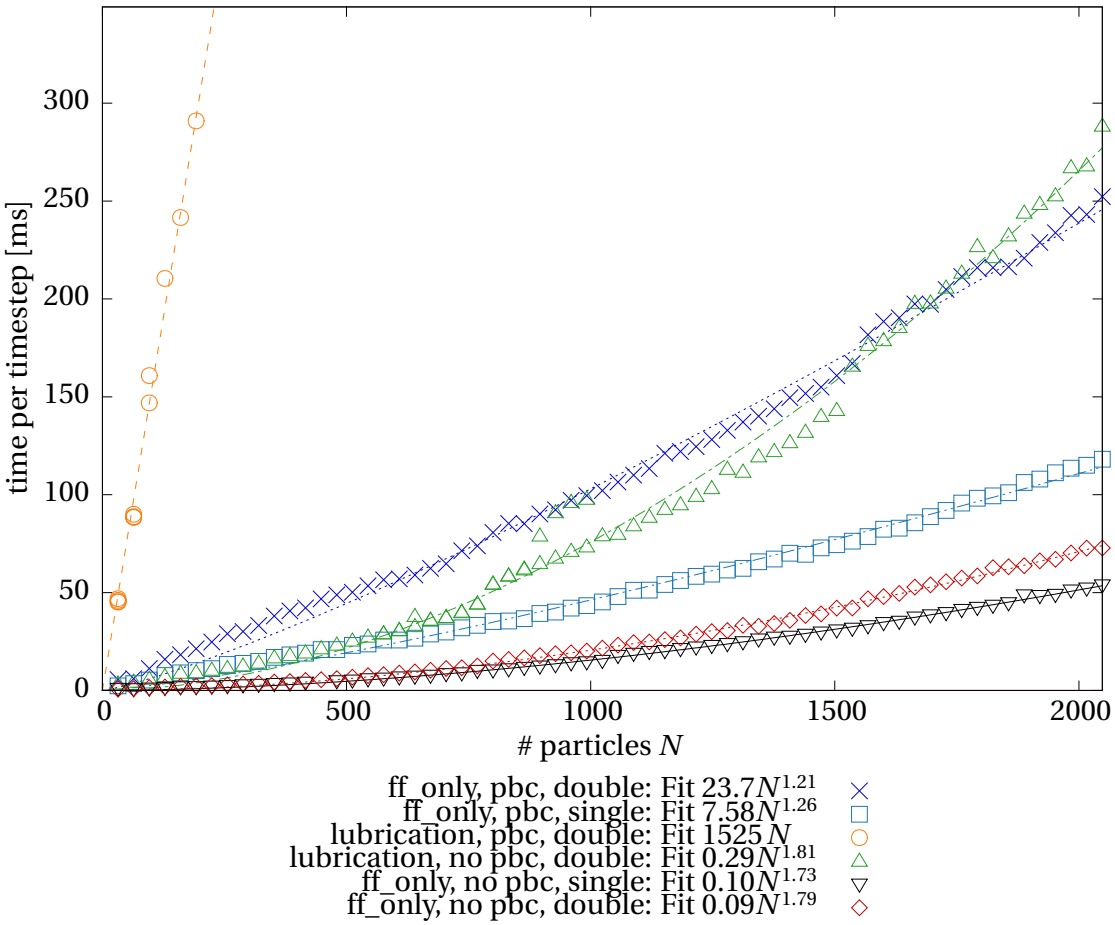

Figure 4.4: Timings of the Stokesian Dynamics implementation for various configurations (lowest is fastest). The slowest version was the one with lubrication correction and Ewald summation (orange circles). This configuration was only stable for double precision. The Ewald summation (blue crosses) is for small particle numbers faster than the lubrication correction (green triangle), however as the Ewald summation has a scaling of $O\left(N^{3/2}\right)$ and the other $O(N^2)$, for $N > 1500$ the Ewald summation is faster. The fastest version is without lubrication correction, without Ewald summation and with single precision. It is notable that the double precision is not much slower than single precision.

5 Results

In this chapter the results will be presented. Section 5.1 will discuss the issues which arise when negative eigenvalues appear in the mobility matrix due to using a cutoff for the hydrodynamic interactions. In section 5.2.6 Stokesian Dynamics will be used to study a colloidal system. The results will be compared with lattice-Boltzmann simulations.

5.1 Negative eigenvalues of $\mathscr{M}$ in Stokesian Dynamics

During the implementation, negative eigenvalues of the mobility matrix $\mathscr{M}$ were observed with PBCs. This was before the Ewald summation was implemented. The studied system had a volume fraction of $\phi \approx 0.12$ and 500 particles. Periodic boundary conditions were applied by using the nearest image convention. Every particle interacts only with the nearest image of every other particle. In order to assert that the negative eigenvalues were not due to the asymmetric cutoff, a spherical cutoff was introduced. And even after expanding the cutoff to ten times the box length, negative eigenvalues persisted.

The negative eigenvalues also appeared without PBCs when an artificial cutoff was introduced. This indicated that the problem is due to the cutoff which is introduced when PBCs are implemented with the nearest image convention.

An example for a configuration where a cutoff results in a negative eigenvalue is shown in fig. 5.1. The arrows represent a force acting on the particles. The way the cutoff is chosen results in more particles within the range of the cutoff that feel a force in the opposite direction as the particle itself. As the particles are close, they feel, due to hydrodynamic interactions, a forces in the opposite direction as they are initially pulled. This leads to a movement in the opposite direction as the force, shown with the arrows. Mathematically this is represented by the fact that this eigenvector has a negative eigenvalue. This corresponding negative eigenvalue is $\lambda_n \approx -0.137\frac{1}{6\pi\eta a}$.

The problems caused by negative eigenvalues are mentioned in sec. 3.1.4. In the case without PBCs, this can be avoided if no cutoff is introduced. With PBCs, the Ewald summation has to be used, in order to avoid a cutoff producing negative eigenvalues.

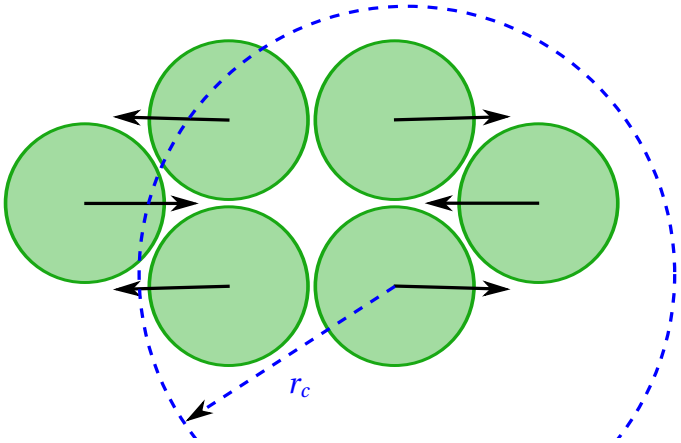

Figure 5.1: Simple example of a configuration that leads to a negative eigenvalue in the mobility matrix. The mobility matrix was constructed using the Rotne–Prager tensor. Shown in green are the spherical particles, which are all on the same plane. The cutoff radius r_c is shown for the lower right particle by the dashed blue line. The high-dimensional eigenvector of the mobility with the negative eigenvalue is shown with solid black arrows.

5.2 Comparison of lattice-Boltzmann and Stokesian Dynamics

In this section the hydrodynamic function of a Yukawa system was studied. This was done with ESPResSo and the Stokesian Dynamics implementation which is also part of ESPResSo. The GPU implementation of lattice-Boltzmann in ESPResSo was used [63]. The system parameters of the simulations, are introduced in sec. 5.2.1. In order to compare the hydrodynamic functions for lattice-Boltzmann and Stokesian Dynamics, some properties of the hydrodynamic function are discussed in sec. 5.2.2 and sec. 5.2.3. They are important as the measured hydrodynamic function is a quantity describing the hydrodynamic interactions that the two algorithms produce. Results obtained with the Langevin thermostat are discussed in sec. 5.2.4 The hydrodynamic function of the Stokesian Dynamic simulation is presented in sec. 5.2.5 before in sec. 5.2.6 the lattice-Boltzmann results are analyzed.

5.2.1 System Parameters

As described in section 2.1, a Yukawa system can be described with two dimensionless parameters, κ and Γ. For these simulations they where chosen as $\kappa = 3$ and $\Gamma = 500$. For these parameters the Yukawa system is in a fluid like state. The other parameters

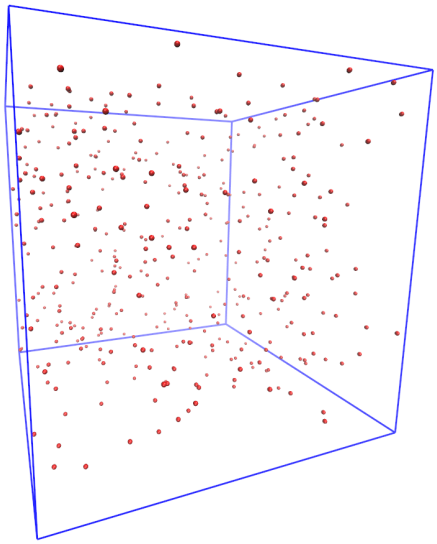

Figure 5.2: Sketch of the simulated system. As the hard core repulsion was not modeled, the hydrodynamic radius of the colloids is shown in red. The box of the periodic boundary condition is sketched in blue. The graph has been created with VMD [66].

are given in molecular dynamics units. The particle density was $n = 0.1a^{-3}$. The box size L, if not otherwise mentioned, is chosen as $L = 16a$. All simulations were done in equilibrium. For the Stokesian dynamics simulation a hydrodynamic radius of $r_h^{SD} = 0.4a$ was chosen. The viscosity η in the Stokesian Dynamics and Brownian Dynamics simulation was chosen as $\eta = 1$.

Fig. 5.2 shows a sketch of the simulated system. The colloids interact via Yukawa potentials and hydrodynamic interaction. As no hard core repulsion was modeld, the graph shows the hydrodynamic radius of the Stokesian Dynamics simulation.

The structure factor obtained with Brownian Dynamics is shown in fig 5.3. As this is a static property, it does not depend on the hydrodynamic interactions and also not on the simulation type. The first peak is around $k = 3.25/a$. The second at $k \approx 6.24/a$ and the third at $k \approx 9.5/a$ are much smaller. The peaks are related to the mean particle nearest neighbor distance. The forth at $k \approx 12.9/a$ and fifth at $k \approx 16.5/a$ are barely visible, as there is no long-ranged order, as expected for a fluid [23]. It looks similar to a radial distribution function $g(r)$. This is due to the fact, that both $g(r)$ in the limit of high r and $S(k)$ in the limit of high k decay to one, if there is no long ranged order. In the limit of $k \to 0$ the structure factor does not decay to zero, unlike $g(r)$. The value for $k \to 0$ is related to the isothermal compressibility [67].

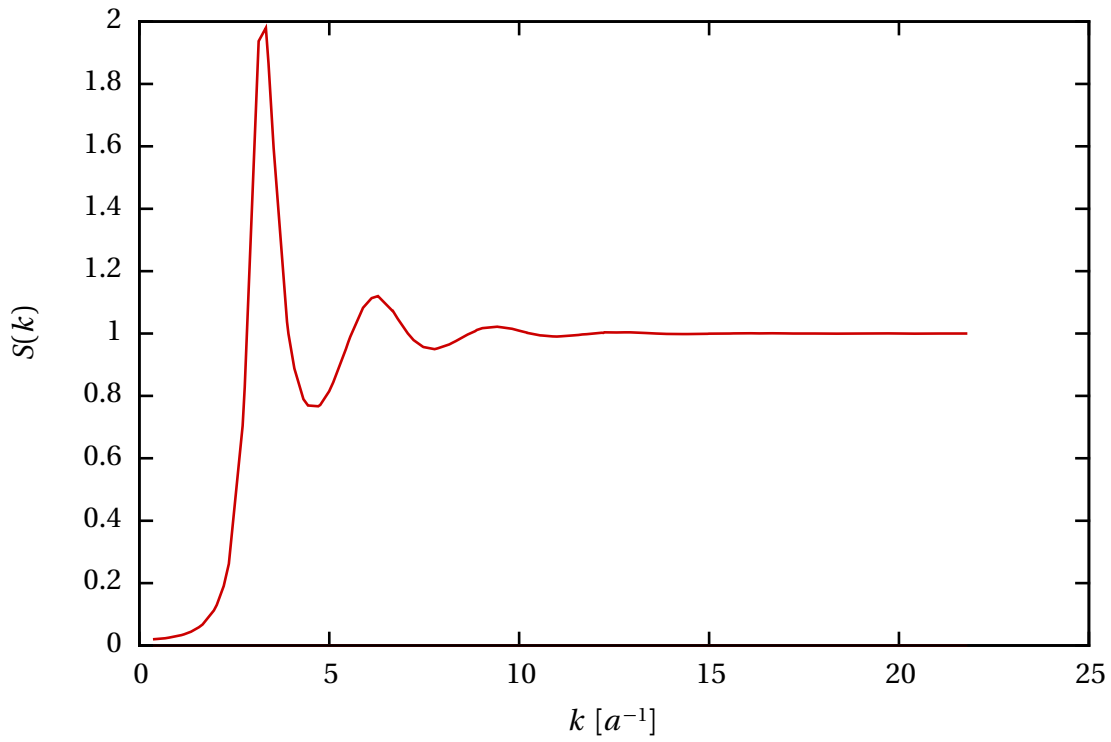

Figure 5.3: Plot of the static structure factor over the k-vector of a Brownian Dynamics simulation. The system parameters are mentioned in sec. 5.2.1.

5.2.2 Autocorrelation Function of the Structure Factor

The hydrodynamic function is calculated using the autocorrelation function (ACF) of the structure factor. As mentioned in section 3.4.3, this function should decay exponentially. Fig. 5.4 shows an example of the ACF for a single k vector of the structure factor. The simulation data fits well if several exponential decays are fitted to the simulation data. On the left, for the smallest timescale, the exponential decay in the short time diffusion is fitted, with a decay rate of $\frac{1}{\tau_1} \approx 5.21$. The decay in the long-time diffusion was not fitted, as the long-time diffusion start at $t \approx 2$, see fig. 5.6. The other relaxation rates $\frac{1}{\tau_2} \approx 2.12$ and $\frac{1}{\tau_3} \approx 0.76$ are on the timescales where the transition from short-time and long time diffusion is taking place and the physical origin of the relaxation process is unclear at this time.

This multi-exponential behavior is visible for all k vectors in the ACF of the structure factor and not only for this single k vector. The time where the transition between the different decay rates happens, however depends on the k vector. For example, the collective modes have for times larger than the short-time diffusion still the same

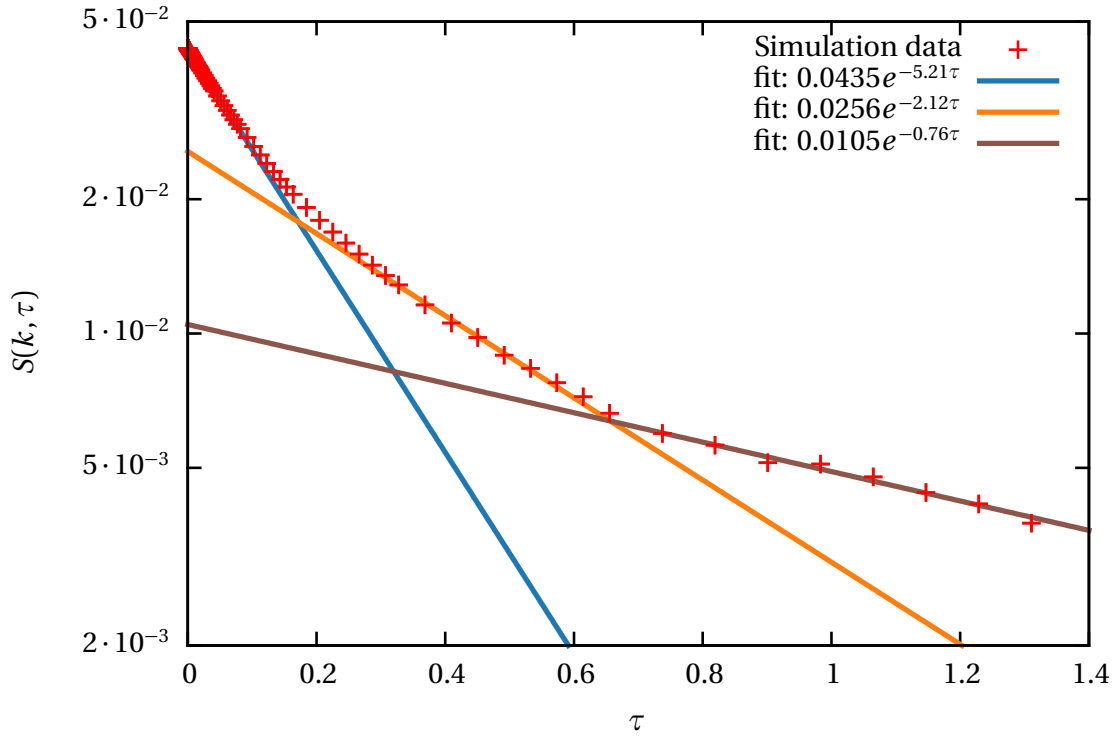

Figure 5.4: Time dependence of the ACF of the structure factor for $k = 1.57/a$ using a Brownian Dynamics simulation. Additional to the simulation data, shown with symbols, three fits of exponential functions are plotted. See sec. 5.2.1 for the system parameters.

decay rate. For intermediate regimes, the transition from the first decay to the second happens directly at the end of the short-time diffusion. The decay here is shown for a Brownian Dynamics simulation, but this multi-exponential behavior is also observed for lattice-Boltzmann, Stokesian Dynamics and the Langevin thermostat. Therefore, the exponential decay has to measured in the short time diffusion.

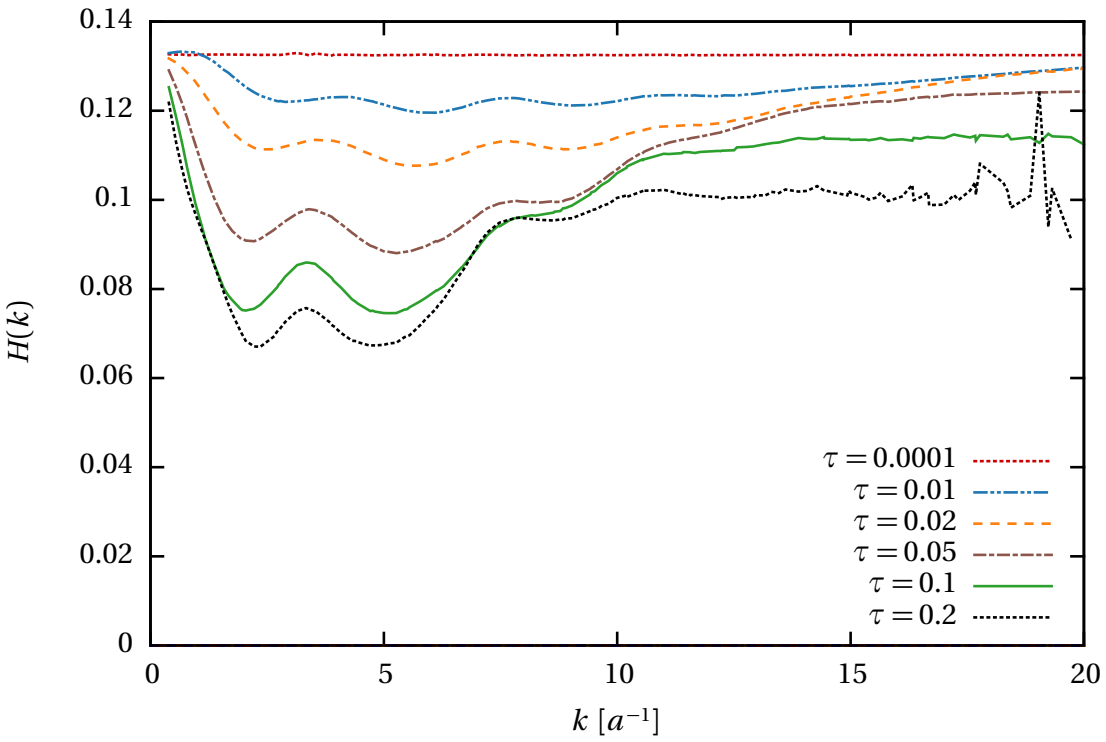

Figure 5.5: Hydrodynamic function of a Brownian Dynamics simulation of colloids. The derivative of the autocorrelation function of the structure factor is taken at different times. The uppermost curve, taken at $\tau = 0.0001$, shows the expected result. See sec. 5.2.1 for the system parameters.

5.2.3 Time dependence of the hydrodynamic function

The hydrodynamic function is calculated using equations (3.80) and (3.83). That means that $H(k)$ has in reality a time dependency $H(k,\tau)$, as it matters where the time derivative is taken. It has been shown in section 3.4.4 that the hydrodynamic function of Brownian Dynamics simulations is a constant.

Figure 5.5 shows different measured hydrodynamic functions. Although it can be shown analytically, see sec. 3.4.4, not all measured hydrodynamic functions are, as expected, a constant. Only if the derivative is taken at $\tau < 0.001$, the measured result matches this expectation. For $\tau > 0.001$ the results are not constants any more. The hydrodynamic function measured at $\tau = 0.01$ is still relatively close to a constant value. The others are further away from the expected result, and with increasing τ the deviation increases. The reason why they are not constants, is that they are not taken in the short-time diffusive regime, meaning after the ballistic regime, but before the in-

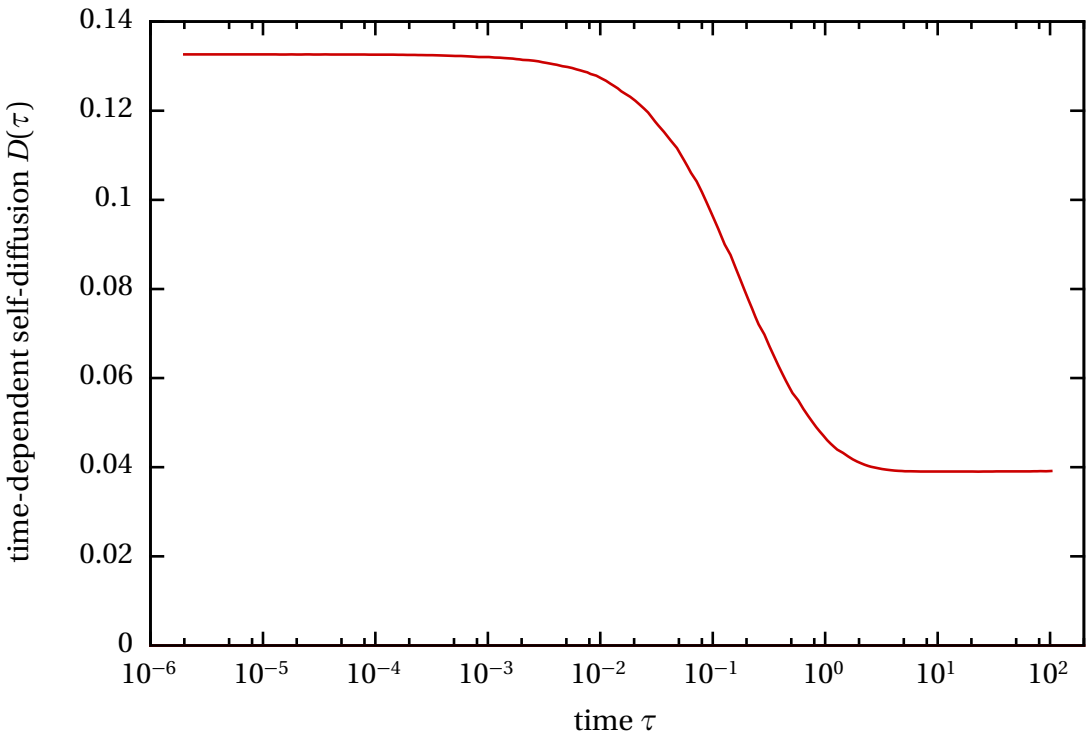

Figure 5.6: Plot of the time-dependent self-diffusion function using Brownian Dynamics. See sec. 5.2.1 for the system parameters.

fluence of neighboring particles is felt. References [23, 54] note that the derivative of the time dependent structure factor has to be taken in the short-time diffusive regime.

It is interesting to note that for $k > 15/a$ the hydrodynamic functions for $\tau > 0.001$ is increasing. This is surprising, as the structure factor is for these k vectors already decayed, and no structure is expected for these high k values.

The time-dependent self-diffusion function using Brownian dynamics of the colloidal system is shown in fig. 5.6. The regions of diffusion are where the function is constant. It consists for small timescales of a short-time diffusion, as Brownian Dynamics directly simulates this motion and no ballistic regime. Between $\tau = 10^{-3}$ and $\tau = 2$, there is a transition region. As the slope of $\mathscr{D}(t)$ is descending in this region, this is also called subdiffusive. For larger times the longtime diffusion is reached. The longtime diffusion is about one third in magnitude of the short-time diffusion. The viscosity or the hydrodynamic radius would not change this ratio, as the ballistic regime is not simulated, the viscosity and the hydrodynamic radius only lead to a rescaling of the time. If the viscosity would be increase by a factor of 10, the diffusion constant would decrease by a factor of 10 and at the same time, the timescale would be shifted

by a order of magnitude to larger times.

5.2.4 Time-dependent self-diffusion function using the Langevin thermostat

In order to see the short-time diffusion regime using a Langevin thermostat, the friction γ has to be sufficiently high. The inverse of the friction is the timescale over which the velocity autocorrelation function decays. This is only true if the timescale γ^{-1} is smaller than the timescale of inter particle collisions. In order to simulate colloidal systems, this should be fulfilled, as shown in table 2.1. Fig. 5.7 shows time-dependent self-

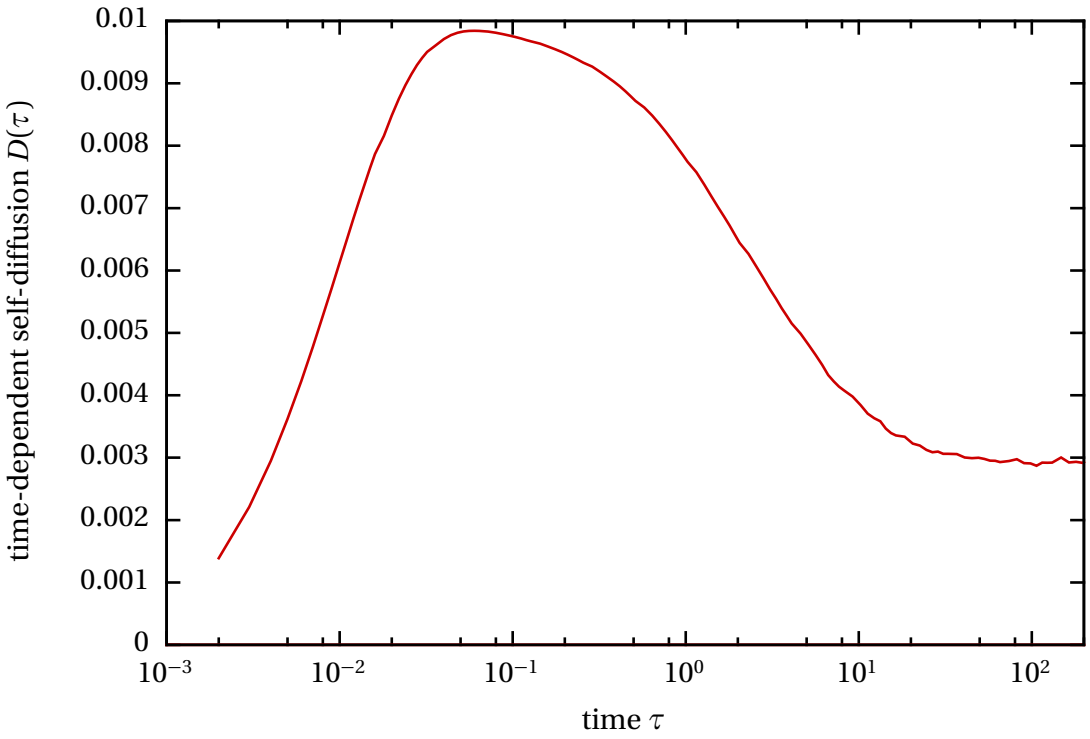

Figure 5.7: Time-dependent self-diffusion function using the Langevin thermostat with a friction $\gamma = 100$. The other parameters are described in sec. 5.2.1.

diffusion using a Langevin thermostat with a friction of $\gamma = 100$. As the Langevin thermostat does not neglect the velocities, the ballistic regime can be seen for $\tau < 0.02$. For $\tau > 0.05$ a subdiffusive regime follows. The long-time diffusion is reached for $\tau > 20$. As there is no clear plateau between the ballistic regime and the subdiffusion regime, the short-time diffusion regime, seen at small τ in fig. 5.6, is not reproduced. For the

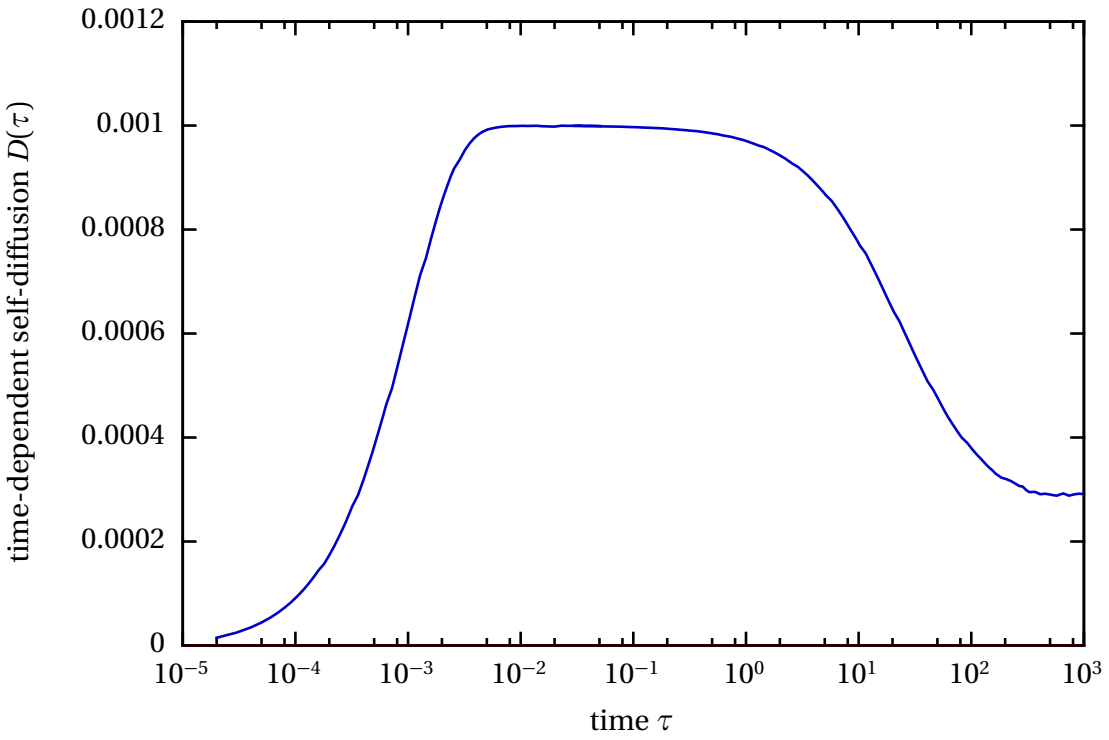

Figure 5.8: Time-dependent self-diffusion function using the Langevin thermostat with a friction $\gamma = 1000$. The other parameters are described in sec. 5.2.1.

system, this means that they are in the ballistic regime, when the influence of the next particles comes into play. A corresponding physical system would have a very dense gas in which the colloids move. To simulate however colloids in an aqueous suspension, the friction has to be increased.

Figure 5.8 shows the the time dependent diffusion for a friction coefficient of $\gamma = 1000$ using the Langevin thermostat. The short-time diffusion is visible between $t \approx 0.005$ and $t \approx 0.5$. Again, for shorter times the ballistic regime is visible. Note that the increase is linear. It looks distorted, as this is a semi-log plot. For higher times first the subdiffusive regime between $\tau \approx 0.5$ and $\tau \approx 500$ and finally long-time diffusion is reached for $\tau \approx 1000$.

The absolute values of the diffusion for $\gamma = 1000$ are smaller than in the case of $\gamma = 100$ and even smaller than in the case of Brownian Dynamics. This is no problem if those two methods should be compared, because by scaling the viscosity in Brownian Dynamics or in Stokesian Dynamics, the results can be matched in the short or long-time diffusion. The ratio between short-time diffusion and long-time diffusion, is, as in the case of Brownian Dynamics, about one third.

Unlike in the case of a friction of $\gamma = 100$, here a clear short-time diffusion regime is visible. As Ref. [68] mentioned, it is possible to telescope the timescales together, but if an important regime, like the short time diffusion is neglected, this is not physical. Additionally to simulate the correct physics, the short-time diffusion is needed to calculate the hydrodynamic function.

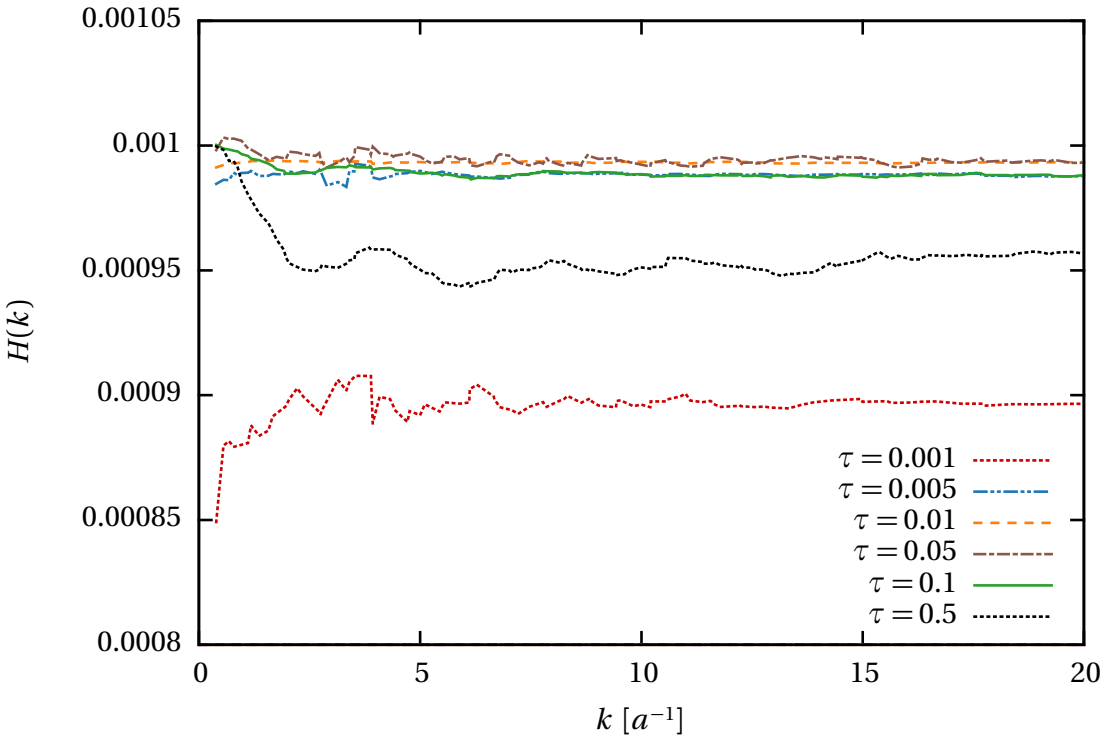

Figure 5.9: Measured hydrodynamic functions $H(k)$ of the Langevin thermostat with a friction of $\gamma = 1000$, taken at different τ in the short-time diffusion. The other system parameters are given in sec. 5.2.1.

The hydrodynamic function, shown in fig. 5.9 has been measured within the short-time diffusive regime. To be precise, for $\tau = 0.001$ the system is still in the transition between ballistic and short-time diffusion. This results in an underestimation of the hydrodynamic function. For $\tau = 0.005$ and $\tau = 0.1$ the functions are close to a constant value, but a bit smaller in magnitude than they for $\tau = 0.01$ and $\tau = 0.05$. This means that the short time diffusion is not perfectly flat, but the short time diffusion is in the center at $\tau \approx 0.02$ a bit larger than at $\tau = 0.1$ or $\tau = 0.0005$ The function measured at $\tau = 0.5$ shows for small k values the right value, however, for larger k values, the function decreases. As discussed in sec. 5.2.2 the collective modes have only for higher times a lower decay rate, therfore it is not surprising to see in the collective regime a

higher hydrodynamic function than for higher k values. If the derivates are taken in the range $0.005 \geq \tau \geq 0.1$, the measured hydrodynamic functions yield the expected result, a constant independent of k.

5.2.5 Hydrodynamic function of Stokesian Dynamics

Figure 5.10 shows the hydrodynamic function of the Yukawa system, described in sec. 5.2.1, which was obtained with the Stokesian Dynamics implementation. It was measured at $\tau = 0.0001$, therefore the result should not be influenced by the problems discussed in sec. 5.2.3. It is clearly visible that the collective diffusion is reduced due to the hydro-

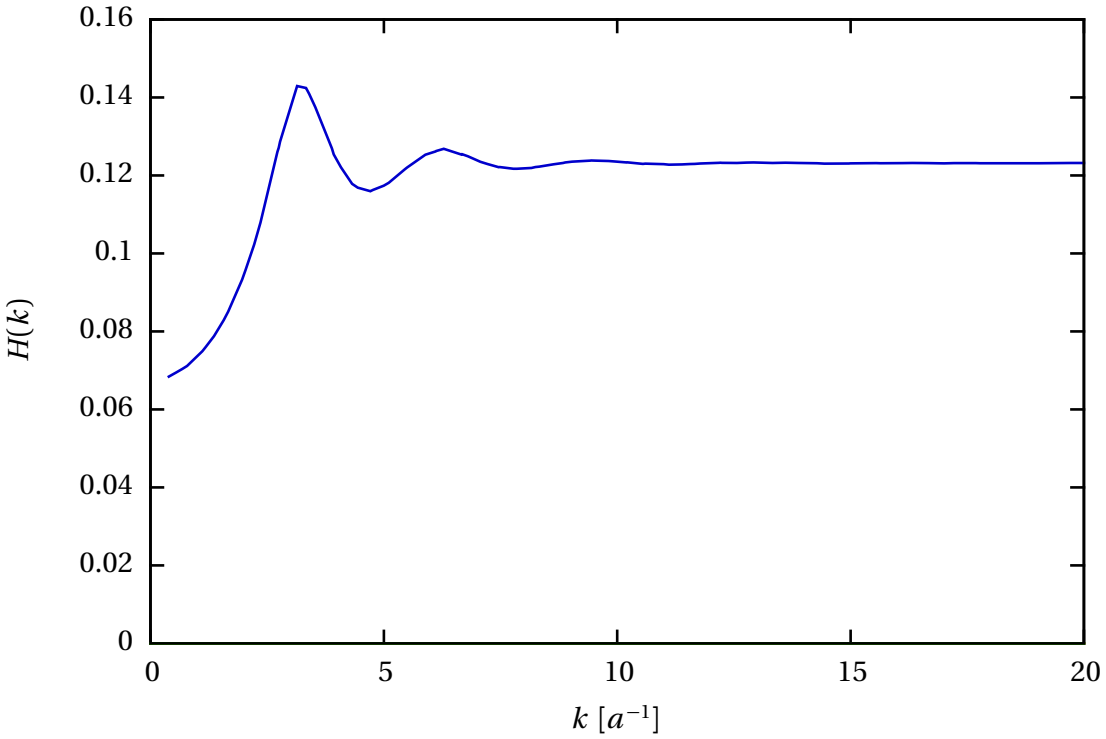

Figure 5.10: Plot of the hydrodynamic function obtained using Stokesian Dynamics. See sec. 5.2.1 for the system parameters.

dynamic interaction. At $k \approx 3/a$ the hydrodynamic interactions increase the motion. This is also where the structure factor has its first peak, which makes sense since hydrodynamic interaction should be strongest between nearest neighbors. Also at $k \approx 6/a$ the structure factor has a peak which is smaller than the main peak at $k \approx 3$, and the same holds for the hydrodynamic function. The next peak at $k \approx 9/a$ is only barely visible. It seems that at length scales where there is structure, the motion is increased.

This could be explained by the fact that the motion of together moving spheres is increased by hydrodynamic interactions.

For large k vectors, the hydrodynamic function is not changed in comparison to Brownian Dynamics. This corresponds to short length scales, where there is only a single particle, and the hydrodynamic interaction is only that of a single sphere with the fluid. This is also what is modeled in Brownian Dynamics, and therefore it can be explained that both methods match in the single particle short-time diffusion.

5.2.6 Viscosity in the Lattice-Boltzmann Method

Lattice-Boltzmann has additional parameters compared to the Langevin thermostat. The lattice-Boltzmann implementation in ESPResSo needs additionally to the friction γ also the grid spacing a, the viscosity η, the time step Δt and the fluid density ρ. Due to the point coupling scheme, the grid spacing a is limited as the hydrodynamic radius is mainly influenced by the grid spacing, see eq. (3.73). The time step has to be at least small enough to resolve the fluctuation of the particles, and has to be

$$\tau < \frac{1}{\gamma} \tag{5.1}$$

so that the integration is stable. Choosing an even smaller τ would lead to a further increase of the computational cost and is therefore unfavorable.

The viscosity has been chosen to tune the hydrodynamic function to fit the results obtained from the Stokesian Dynamics simulation. As shown before in section 5.2.3 it is necessary to measure the time-dependent self-diffusion to measure the hydrodynamic function in the region of short-time diffusion. Therefore, first the time-dependent self-diffusion has been measured for different viscosities η. The results are shown in fig. 5.11. With increasing viscosity the diffusion decreases. For viscosities $\eta < 100$ the short-time diffusion vanishes. For lower viscosities the time-dependent self-diffusion only reaches a plateau in the longtime diffusion. The short-time diffusion is not reproduced for such low η. For $\eta \leq 10$ it looks like the ballistic regime would extend up to $\tau \approx 0.5$. But this regime is not a ballistic regime, but rather a superdiffusive regime where $\mathscr{D}(t)$ obeys a power law

$$\mathscr{D}(t) \propto t^{\alpha} \tag{5.2}$$

where the exponent α is smaller than 2. For the case of $\eta = 0.2$ the exponent is $\alpha \approx 1.66$. The reason for this superdiffusion is probably that the momentum relaxation time of the fluid is too large. This introduces long-time correlation in the system, which results in the vanishing of the short-time diffusion. As shown in table 2.1 the relaxation time of the fluid is larger than the time over which the ballistic motion decays, but it still should

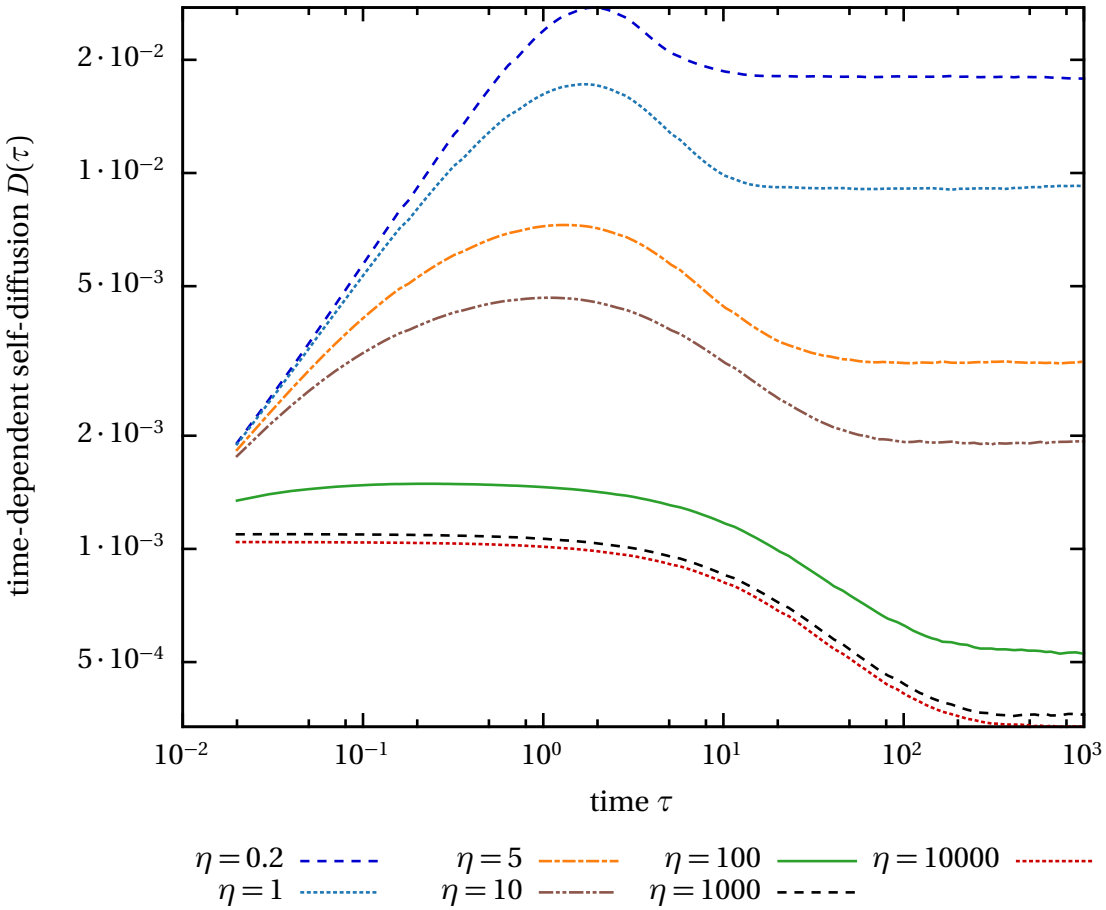

Figure 5.11: Time-dependent self-diffusion for lattice-Boltzmann. The ballistic regime is omitted. All simulations are done with a friction of $\gamma = 1000$ and a time step $\Delta t = 0.0005$. The other system parameters are given in sec. 5.2.1.

be much smaller than the time over which the configuration changes. Therefore the short-time diffusion should be clearly visible.

Figure 5.12 shows the hydrodynamic function of a lattice-Boltzmann simulation for different viscosities. With decreasing viscosity, the hydrodynamic function increases. The collective diffusion for lower k values increases far less then the single particle region. For $\eta = 10000$ the hydrodynamic function is nearly constant. For the lower viscosities, the particles are dragged with the lattice-Boltzmann fluid. The lower the viscosity, the stronger the influence of the fluid. Note that in a physical system, the viscosity should not have an influence on the normalized hydrodynamic function. The viscosity would lead to a scaling in time, which would lead to a scaling of the relaxation

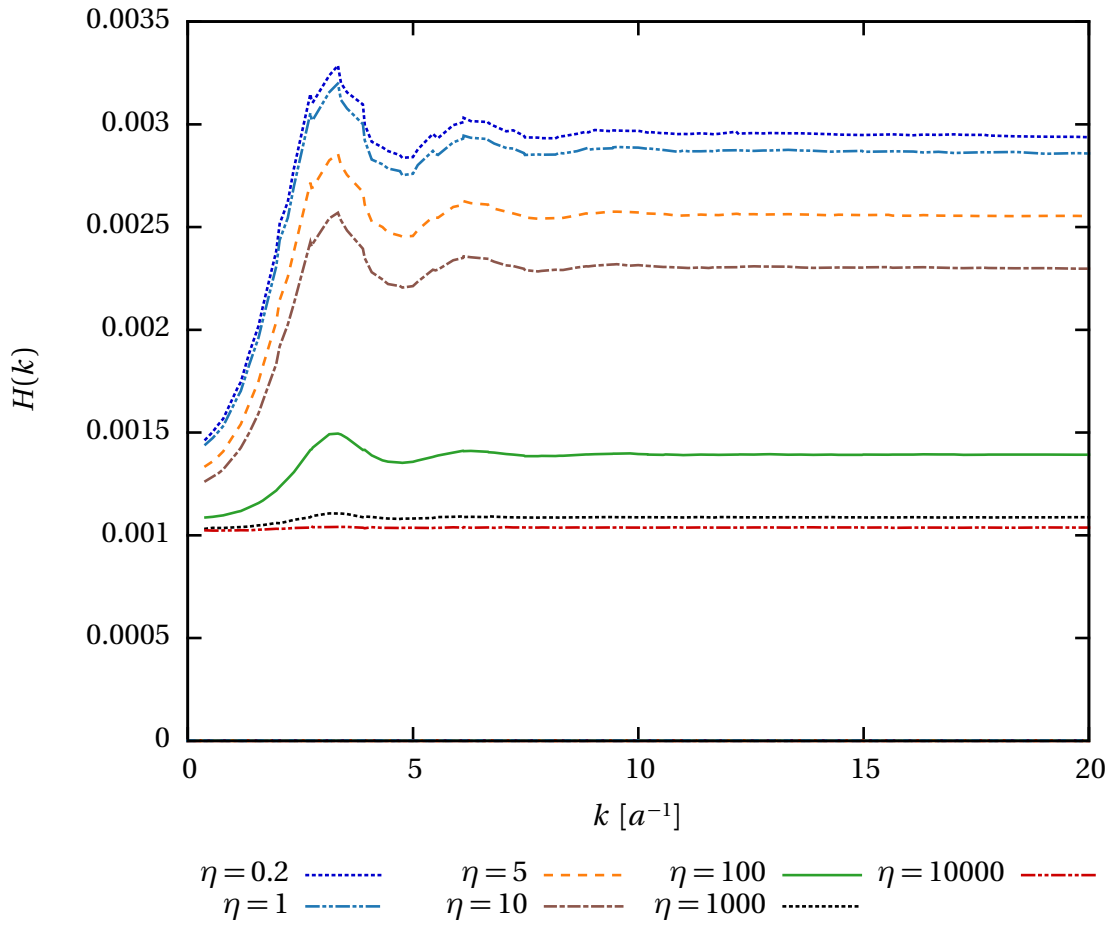

Figure 5.12: Hydrodynamic function over k for lattice-Boltzmann. All simulations are done with a friction of $\gamma = 1000$ and a time step $\Delta t = 0.0005$. The other system parameters are given in sec. 5.2.1.

times of the autocorrelation function of the structure factor. But as this is for all k vectors the same scaling, after the normalization this difference would be gone.

Figure 5.13 displays the normalized hydrodynamic function for both lattice-Boltzmann and Stokesian Dynamics. The lattice-Boltzmann results for viscosities $\eta > 10$ the influence of the hydrodynamic interaction is underestimated. For $\eta = 10$ the collective influence, at low k values, is correctly modeled. For higher k values at $k > 3$ the influence is under estimated. For $\eta < 10$ the slow down of the collective mode is over estimated. At the first peak at $k \approx 3.25/a$ the influence is under estimated, and the increase of mobility is too low. To be specific, the transition from over to under estimation happens at $k \approx 1.4/a$. Although there are differences for $\eta \leq 10$ between lattice-Boltzmann and

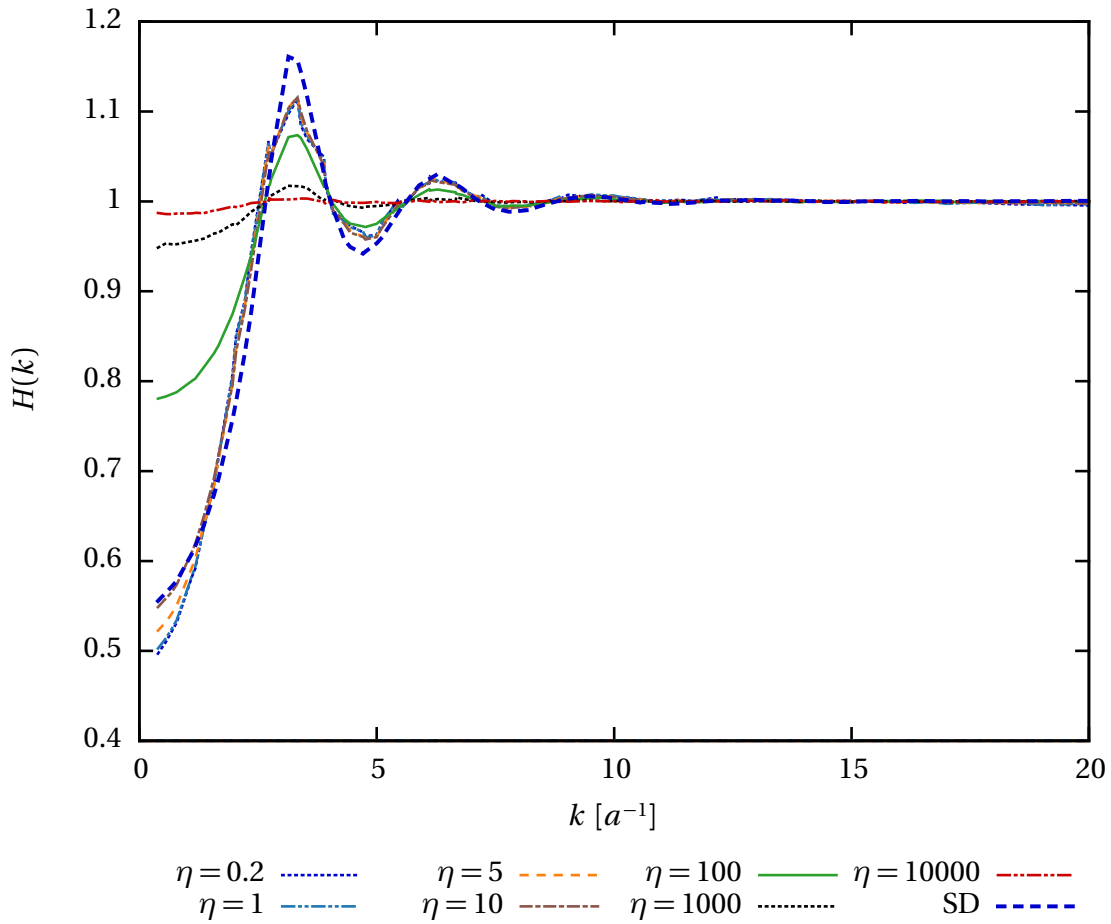

Figure 5.13: Normalized hydrodynamic function $\tilde{H}(k)$ over k for lattice-Boltzmann and Stokesian Dynamics (SD). All lattice-Boltzmann simulations are done with a friction of $\gamma = 1000$ and a time step $\Delta t = 0.0005$. The other system parameters are given in sec. 5.2.1.

Stokesian Dynamics, the agreement is still rather good. For higher viscosities, the influence of the hydrodynamic interaction vanishes with increasing viscosity. This would mean that smaller viscosities give better results, but the time dependent single particle diffusion suggested the opposite, that larger viscosities model reality better. It is therefore, based on this analysis, not clear what parameters to use.

6 Conclusion and Outlook

Stokesian Dynamics is a powerful tool to simulate hydrodynamic interactions. The fact that it is possible to directly simulate an over-damped system is an advantage, as it lowers the computational cost.

One very important thing which one should take care of, is to avoid a cutoff in the far field of the hydrodynamic interactions, as discussed in sec. 5.1. In the absence of periodic boundary conditions this is straight forward to fulfill by looping over all particle pairs. If periodic boundary conditions are used, the Ewald summation of the Rotne–Prager tensor has to be used. In this way negative eigenvalues can be avoided. This is for physical reasons important, but also for practical reasons, as negative eigenvalues can lead to a break down of the implicit or explicit solvers for the inverse matrix. It is also not possible to thermalize a system with negative eigenvalues. Additionally, the system can behave unphysically if negative eigenvalues appear.

In the comparison of the hydrodynamic function, obtained with lattice-Boltzmann and Stokesian Dynamics, it was possible to tune the lattice-Boltzmann parameters to obtain a hydrodynamic function close to the one of Stokesian Dynamics in sec. 5.2.6. It seems that, even with point coupling, proper hydrodynamics can be simulated. It was, with the chosen method, not distinguishable whether for frictions $\gamma < 1000$ hydrodynamics were incorporated correctly in the simulation as discussed in sec. 5.2.4. If one uses lattice-Boltzmann for the simulation of a colloidal system, one should therefore always check the mean square displacement. If the system does not reach short-time diffusion before the inter-particle interactions become dominant, the friction should be increased. In addition, the hydrodynamic function should be checked. Hydrodynamic functions of other simulation methods can be matched to the one produced by Stokesian Dynamics when the system cannot be simulated with Stokesian Dynamics. This is not limited to lattice-Boltzmann simulations. All methods in which reasonable results can only be expected within a certain limit, or if the parameters are not directly linked to physics, at least the mean squared displacement and the hydrodynamic function should be matched. Examples are dissipative particle dynamics (DPD) [69], multi-particle collision dynamics (MPC / SRD) [70], or as studied here the lattice-Boltzmann algorithm [71].

The current version of the code for a system consisting of 1000 particles about needs 100 ms for a time step on a GeForce GTX 680 with periodic boundary conditions switched on. This can probably be further improved, if a grid based FFT method, as described by

Sierou and Brady in ref. [55] would be implemented. As regular grid based algorithms can be especially efficient on GPUs, using CUDA for implementation would be beneficial. The improvement would be necessary, to study systems with more than a few thousand particles.

A second possible improvement of the code would be to incorporate higher moments of the force density in the mobility matrix. The current version of this implementation has the first moments, namely forces and displacements. Higher moments would be needed to simulate dense suspensions. The higher degrees of freedom, such as rotation, are short ranged. Therefore they can probably be neglected in Yukawa systems, where the electrostatic repulsion prevents colloids from getting close. To study systems of dense suspensions, or suspensions where the particles are close to contact, the higher moments might be important for the simulation of the correct dynamics of the system.

With the current implementation it is possible to study systems of up to 5000 particles. Hydrodynamic interactions in colloidal systems can be studied, as already done in this work. Here the focus was on the short-time dynamics of colloids interacting via the Yukawa potential. Also long-time dynamics can be studied. The crystallization process of a colloidal Yukawa system can be studied. This would allow one to study whether the deviations in the short-time dynamics of the hydrodynamic function of lattice-Boltzmann influence the results in the long-time dynamics of the crystallization.

It might also be of interest to further study the hydrodynamic function obtained by lattice-Boltzmann simulations. Although the hydrodynamic function of the lattice-Boltzmann fluid is close to the one obtained with Stokesian Dynamics, there is still room for improvement. A possibility would be to vary the time step, and to increase thereby the speed of sound. A better match with the hydrodynamic function of the Stokesian Dynamics could be in this way achieved. The disadvantage is that a smaller time step always means a higher computational cost, as the total number of time steps increases in order to propagate the system by the same time.

Bibliography

[1] J. F. Brady and G. Bossis, "Stokesian dynamics," *Annual Review of Fluid Mechanics*, vol. 20, pp. 111–157, 1988.

[2] J. J. Cerdaà, C. Holm, and K. Kremer, "Novel simulation approaches for polymeric and soft matter systems," *Macromolecular Theory and Simulations*, vol. 20, pp. 444–445, 2011.

[3] S. Raafatnia, O. A. Hickey, and C. Holm, "Mobility reversal of polyelectrolyte-grafted colloids in monovalent salt solutions," *Phys. Rev. Lett.*, vol. 113, p. 238301, Dec 2014.

[4] S. I. Kam, "Improved mechanistic foam simulation with foam catastrophe theory," *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, vol. 318, no. 1–3, pp. 62 – 77, 2008.

[5] S. Bardenhagen, A. Brydon, and J. Guilkey, "Insight into the physics of foam densification via numerical simulation," *Journal of the Mechanics and Physics of Solids*, vol. 53, no. 3, pp. 597 – 617, 2005.

[6] S. Magrabi, B. Dlugogorski, and G. Jameson, "Bubble size distribution and coarsening of aqueous foams," *Chemical Engineering Science*, vol. 54, no. 18, pp. 4007 – 4022, 1999.

[7] D. D. Brewer and S. Kumar, "Dynamic simulation of sediment films of yukawa-stabilized particles," *Phys. Rev. E*, vol. 91, p. 022304, Feb 2015.

[8] G. A. Méndez-Maldonado, G. A. Chapela, J. A. Martínez-González, J. A. Moreno, E. Díaz-Herrera, and J. Alejandre, "Fluid-solid coexistence from two-phase simulations: Binary colloidal mixtures and square well systems," *The Journal of Chemical Physics*, vol. 142, no. 5, pp. –, 2015.

[9] K. J. Naidoo and J. Schnitker, "Melting of two-dimensional colloidal crystals: A simulation study of the yukawa system," *The Journal of Chemical Physics*, vol. 100, no. 4, pp. 3114–3121, 1994.

[10] M. Broccio, D. Costa, Y. Liu, and S.-H. Chen, "The structural properties of a two-yukawa fluid: Simulation and analytical results," *The Journal of Chemical Physics*, vol. 124, no. 8, 2006.

[11] H. Löwen, "Structure and brownian dynamics of the two-dimensional yukawa fluid," *Journal of Physics: Condensed Matter*, vol. 4, no. 50, p. 10105, 1992.

[12] A. Arnold, B. A. Mann, H. Limbach, and C. Holm, "Espresso – an extensible simulation package for research on soft matter systems," in *Forschung und wissenschaftliches Rechnen 2003*, Gesellschaft für wissenschaftliche Datenverarbeitung mbH.

[13] U. D. Schiller, *Thermal fluctuations and boundary conditions in the lattice Boltzmann method*. PhD thesis, Johannes Gutenberg-Universitaet Mainz, Fachbereich 08: Physik, Mathematik und Informatik, 2008.

[14] P. Ahlrichs and B. Dünweg, "Simulation of a single polymer chain in solution by combining lattice boltzmann and molecular dynamics," *Journal of Chemical Physics*, vol. 111, no. 17, pp. 8225–8239, 1999.

[15] S. Kim and S. J. Karrila, *Microhydrodynamics: principles and selected applications*. Courier Corporation, 2013.

[16] S. Stanislaw, A. J. V., G. R. G., H. Michael, H. Kazuyuki, J. R. G., K. Przemyslaw, M. Ingrid, M. Werner, P. Stanisław, and S. R. F. T., "Terminology of polymers and polymerization processes in dispersed systems (iupac recommendations 2011)," *Pure and Applied Chemistry*, vol. 83, p. 2229, 2011.

[17] J. D. Weeks, D. Chandler, and H. C. Andersen, "Role of repulsive forces in determining the equilibrium structure of simple liquids," *Journal of Chemical Physics*, vol. 54, p. 5237, 1971.

[18] D. Roehm, S. Kesselheim, and A. Arnold, "Hydrodynamic interactions slow down crystallization of soft colloids," *Soft Matter*, pp. –, 2014.

[19] K. Kremer, M. O. Robbins, and G. S. Grest, "Phase-diagram of yukawa systems - model for charge-stabilized colloids," *Physical Review Letters*, vol. 57, p. 2694, 1986.

[20] S. Hamaguchi, R. Farouki, and D. Dubin, "Phase diagram of yukawa systems near the one-component-plasma limit revisited," *The Journal of chemical physics*, vol. 105, p. 7641, 1996.

[21] D. F. Evans and H. Wennerström, *The Colloidal Domain*. New York: Wiley-VCH, 1999.

[22] M. Radu and T. Schilling, "Solvent hydrodynamics speed up crystal nucleation in suspensions of hard spheres," *EPL (Europhysics Letters)*, vol. 105, no. 2, p. 26001, 2014.

[23] G. Naegele, "On the dynamics and structure of charge-stabilized suspensions," *Physics Reports*, vol. 272, no. 5–6, pp. 215–372, 1996.

[24] L. Landau and E. Lifshitz, "Statistische physik, vol. 5," *Akademie-Verlag, Berlin*, 1987.

[25] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge university press, 2000.

[26] D. L. Ermak and J. McCammon, "Brownian dynamics with hydrodynamic interactions," *The Journal of chemical physics*, vol. 69, p. 1352, 1978.

[27] H. Yamakawa, *Modern theory of polymer solutions*. 1971.

[28] W. Demtröder, *Experimentalphysik 1: Mechanik und Wärme*. Springer-Verlag, 2006.

[29] J. Rotne and S. Prager, "Variational treatment of hydrodynamic interaction in polymers," *The Journal of Chemical Physics*, vol. 50, p. 4831, 1969.

[30] K. ICHIKI, "Improvement of the stokesian dynamics method for systems with a finite number of particles," *Journal of Fluid Mechanics*, vol. 452, pp. 231–262, 2 2002.

[31] M. W. Kim and D. G. Pfeiffer, "Poly-electrolyte properties of ionomeric polymers," *Euro. Phys. Lett.*, vol. 5, p. 321, 1988.

[32] S. Kesselheim, *Simulations of DNA Translocation through nanopores*. PhD thesis, Universitaet Stuttgart, 2014.

[33] A. J. Banchio and J. F. Brady, "Accelerated stokesian dynamics: Brownian motion," *The Journal of Chemical Physics*, vol. 118, no. 22, pp. 10323–10332, 2003.

[34] C. W. J. Beenakker, "Ewald sum of the rotne–prager tensor," *The Journal of Chemical Physics*, vol. 85, no. 3, pp. 1581–1582, 1986.

[35] M. Deserno and C. Holm, "How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines," *Journal of Chemical Physics*, vol. 109, p. 7678, 1998.

[36] A. Meister, *Numerik linearer Gleichungssysteme: Eine Einführung in moderne Verfahren*. Wiesbaden: Vieweg, 3., überarbeitete auflage ed., 2008.

[37] J. P. Boyd, *Chebyshev and Fourier spectral methods*. Courier Corporation, 2000.

[38] "Chebyshev polynomial of the first kind – from wolfram mathworld." `http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html`. Accessed: 2015 Jan. 23.

[39] "Chebyshev-gauss quadrature – from wolfram mathworld." `http://mathworld.wolfram.com/Chebyshev-GaussQuadrature.html`. Accessed: 2014 Dec. 05.

[40] W. H. Press, *Numerical recipes: the art of scientific computing*. Cambridge Univ Pr, 2007.

[41] R. Benzi, S. Succi, and M. Vergassola, "The lattice boltzmann equation: theory and applications," *Physics Reports*, vol. 222, no. 3, pp. 145–197, 1992.

[42] S. Chen and G. D. Doolen, "Lattice boltzmann method for fluid flows," *Annual Review of Fluid Mechanics*, vol. 30, no. 1, pp. 329–364, 1998.

[43] D. d'Humieres, "Multiple–relaxation–time lattice boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002.

[44] S. Succi, *The lattice Boltzmann equation for fluid dynamics and beyond*. Oxford University Press, USA, 2001.

[45] D. A. Wolf-Gladrow, *Lattice-gas cellular automata and lattice Boltzmann models: an introduction*, vol. 1725. Springer, 2000.

[46] G. Rempfer, "A lattice based model for electrokinetics." Master Thesis, Institute for Computational Physics, University of Stuttgart, March, 2013.

[47] K. C. Grass, *Towards realistic modelling of free solution electrophoresis: a case study on charged macromolecules*. PhD thesis, Goethe-Universität Frankfurt am Main, 2008.

[48] L.-S. Luo, W. Liao, X. Chen, Y. Peng, and W. Zhang, "Numerics of the lattice boltzmann method: Effects of collision models on the lattice boltzmann simulations," *Phys. Rev. E*, vol. 83, p. 056710, May 2011.

[49] A. J. C. Ladd, "Numerical simulations of particulate suspensions via a discretized boltzmann equation. part 1. theoretical foundation," *Journal of Fluid Mechanics*, vol. 271, pp. 285–309, 1994.

[50] O. B. Usta, A. J. C. Ladd, and J. E. Butler, "Lattice-boltzmann simulations of the dynamics of polymer solutions in periodic and confined geometries," *Journal of Chemical Physics*, vol. 122, p. 094902, 2005.

[51] N.-Q. Nguyen and A. Ladd, "Lubrication corrections for lattice-boltzmann simulations of particle suspensions," *Physical Review E*, vol. 66, no. 4, p. 046708, 2002.

[52] X. ting Zhao, H. Yang, Y. zhen Sheng, J. yin Li, and M. Sun, "Molecular dynamics simulation on the effect of the distance between SWCNTs for short polymers diffusion among single wall carbon nanotubes," *Computational Materials Science*, vol. 95, no. 0, pp. 446 – 450, 2014.

[53] L. Van Hove, "Correlations in space and time and born approximation scattering in systems of interacting particles," *Phys. Rev.*, vol. 95, pp. 249–262, Jul 1954.

[54] J. Gapinski, A. Patkowski, and G. Nägele, "Generic behavior of the hydrodynamic function of charged colloidal suspensions," *The Journal of chemical physics*, vol. 132, no. 5, p. 054510, 2010.

[55] A. Sierou and J. F. Brady, "Accelerated stokesian dynamics simulations," *Journal of Fluid Mechanics*, vol. 448, pp. 115–146, 12 2001.

[56] C. Trendall and A. J. Stewart, "General calculations using graphics hardware, with application to interactive caustics," in *Eurographics Workshop on Rendering*, pp. 287–298, Springer, June 2000.

[57] "Status of openmp in gcc – the gnu project." `http://gcc.gnu.org/wiki/openmp`. Accessed: 2015 Jan. 09.

[58] "Openmp support for clang – the llvm project." `http://openmp.llvm.org/`. Accessed: 2015 Jan. 09.

[59] "An implementation of the openmp c/c++ language extensions in clang/llvm compiler – openmp/clang." `https://clang-omp.github.io/`. Accessed: 2015 Jan. 09.

[60] NVIDIA Corporation, *NVIDIA CUDA C Programming Guide Version 6.5*, 2014.

[61] "Cuda c best practices guide," tech. rep., NVIDIA, 2014.

[62] M. Kopp and F. Höfling, "GPU-accelerated simulation of colloidal suspensions with direct hydrodynamic interactions," *Eur. Phys. J. Special Topics (2012)*, 2012.

[63] D. Röhm and A. Arnold, "Lattice Boltzmann simulations on GPUs with ESPResSo," *The European Physical Journal Special Topics*, vol. 210, pp. 89–100, 2012.

[64] NVIDIA Corporation, *cuRAND Library - Programming Guide Version 6.5*, 2014.

[65] M. A. Wafai, *Sparse matrix vector multiplications on graphic processors.* 2009.

[66] W. Humphrey, A. Dalke, and K. Schulten, "VMD: Visual molecular dynamics," *Journal of Molecular Graphics*, vol. 14, pp. 33–38, 1996.

[67] J. Jarzynski, J. R. Smirnow, and C. M. Davis, "Isothermal compressibility and the structure factor of liquid alkali metals," *Phys. Rev.*, vol. 178, pp. 288–291, Feb 1969.

[68] J. T. Padding and A. A. Louis, "Hydrodynamic interactions and brownian forces in colloidal suspensions: Coarse-graining over time and length scales," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 74, no. 3, p. 031402, 2006.

[69] P. J. Hoogerbrugge and J. M. V. A. Koelman, "Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics," *Europhysics Letters*, vol. 19, no. 3, pp. 155–160, 1992.

[70] A. Malevanets and R. Kapral, "Mesoscopic model for solvent dynamics," *Journal of Chemical Physics*, vol. 110, pp. 8605–8613, May 1999.

[71] A. J. C. Ladd, "Dynamical simulations of sedimenting spheres," *Phys. Fluids A*, vol. 5, no. 2, pp. 299–310, 1993.