Spinodal Decomposition of Polymer-Solvent Systems: Theory and Simulation

Dissertation zur Erlangung des Grades

„Doktor der Naturwissenschaften“

am Fachbereich Physik, Mathematik und Informatik
der Johannes Gutenberg-Universität in Mainz

Dominic Spiller

Mainz, den 3. August 2021

Betreuer:

Apl. Prof. Dr. Burkhard Dünweg

Datum der mündlichen Prüfung:

18. Januar 2022

Acknowledgments

I would like to thank the TRR146 of the DFG for the financial support that made this research project possible. Being able to visit conferences and workshops broadened my horizon significantly on both a scientific as well as personal level. Let me also thank the lead of the theory group for affiliating me, and for providing excellent working conditions.

To my supervisor I am deeply grateful for his kindness, patience, and for being an excellent mentor. His expert guidance was without a doubt integral to the completion of this thesis. The lessons I learned from him during countless hours of enlightening discussion are invaluable.

Furthermore, I would like to thank our collaborators for their contributions to this project, as well as their insightful comments and suggestions.

I want to thank my colleagues for the multitude of occasions in which they have helped me overcome major and minor difficulties, and for making the MPIP a wonderful workplace.

To my family and friends, I am in great debt. Without their continuous support throughout my studies, it would not have been possible to accomplish this thesis.

Furthermore, I would like to thank the library, IT, and administrative staff at the MPIP for their assistance in numerous cases, as well as all the other people involved that cannot be mentioned explicitly here.

Abstract

The unmixing dynamics of a polymer-solvent system is investigated via computer simulation of a mesoscopic model. The polymer component is represented by a bead-spring model where the quality of the solvent can be varied by adjusting the attractive component of the pair interaction. The polymers are dissipatively coupled to a Lattice-Boltzmann solvent background in order to include hydrodynamic interactions. The project's overarching goal is to compare the present model to a continuum model that is being investigated in a collaborating group and perform a parameter matching. The fact that the available macroscopic models were found to be lacking a comprehensible connection to microscopic physics motivated the creation of a novel continuum theory from scratch. This theory is derived from microscopic principles, always keeping the numerical implementation in mind and at the same time making sure that it is compatible with the GENERIC formalism and fundamental symmetries. Thus, all parameters have a well-defined meaning with respect to the computer model.

Numerical simulations of the mesoscopic model are performed using a coupled Lattice Boltzmann/Molecular Dynamics approach which at the same time provides the basic picture used in the derivation of the continuum model. The attraction strength corresponding to the theta solvent is estimated for two-dimensional systems. Three-dimensional systems are simulated at various densities. The results are used to estimate the parameters of the analytically determined Van der Waals equation of state. The coarsening dynamics during phase separation is examined by calculating the dynamic structure factor for both two- and three-dimensional systems. Comparison to a simple fluid reveals that in the viscoelastic case dynamic scaling is violated, meaning that the phase-separation dynamics of the present computer model is indeed non-standard. First comparisons are made to a macroscopic model which was simulated by a collaborating group. Furthermore, the use of Minkowski functionals for analyzing the dynamics of the emerging structures' geometrical properties is explored.

In parallel, a procedure for calculating Lattice Boltzmann weights based on the numerical solution of the Maxwell-Boltzmann constraints has been developed and implemented in a Python script. The script determines whether or not a valid model corresponding to the user-supplied parameters exists and calculates its weights if this is the case. A procedure to obtain particular models even when there are infinitely many solutions to the problem is available. Furthermore, the script provides a function to determine the validity of a solution, which can be used to verify existing models from the literature.

Zusammenfassung

Die Entmischungsdynamik eines Polymer-Lösungsmittelsystems wird durch Computersimulation eines mesoskopischen Modells untersucht. Die Polymerkomponente wird durch ein Kugel-Feder-Modell dargestellt, bei dem die Qualität des Lösungsmittels durch Anpassung der attraktiven Komponente der Paarwechselwirkung variiert werden kann. Die Polymere sind dissipativ an einen Lattice-Boltzmann-Lösungsmittelhintergrund gekoppelt, um hydrodynamische Wechselwirkungen zu berücksichtigen. Das übergeordnete Ziel des Projekts ist es, das vorliegende Modell mit einem Kontinuumsmodell zu vergleichen, das in einer kollaborierenden Gruppe untersucht wird, und einen Parameterabgleich durchzuführen. Die Tatsache, dass den verfügbaren makroskopischen Modellen eine nachvollziehbare Verbindung zur mikroskopischen Physik fehlt, motivierte die Herleitung einer neuartigen Kontinuumstheorie. Diese Theorie wird aus mikroskopischen Prinzipien abgeleitet, wobei die numerische Umsetzung stets im Auge behalten und gleichzeitig sichergestellt wird, dass sie mit dem GENERIC-Formalismus und den fundamentalen physikalischen Symmetrien kompatibel ist. Somit haben alle Parameter eine wohldefinierte Bedeutung in Bezug auf das Computermodell.

Numerische Simulationen des mesoskopischen Modells werden mit einem gekoppelten Lattice Boltzmann/Molekulardynamik-Ansatz durchgeführt, der gleichzeitig das Grundbild liefert, das bei der Herleitung des Kontinuumsmodells verwendet wird. Für zweidimensionale Systeme wird die dem Theta-Lösungsmittel entsprechende Attraktionsstärke abgeschätzt. Dreidimensionale Systeme werden bei verschiedenen Dichten simuliert und die Ergebnisse verwendet, um die Parameter der analytisch bestimmten Van-der-Waals-Zustandsgleichung abzuschätzen. Die Vergröberungsdynamik bei der Phasentrennung wird durch Berechnung des dynamischen Strukturfaktors sowohl für zwei- als auch für dreidimensionale Systeme untersucht. Der Vergleich mit einem einfachen Fluid zeigt, dass im viskoelastischen Fall die dynamische Skalierung verletzt wird, was bedeutet, dass die Phasentrennungsdynamik des vorliegenden Computermodells tatsächlich nicht dem Standardverhalten entspricht. Erste Vergleiche werden mit einem makroskopischen Modell angestellt, das von einer kollaborierenden Arbeitsgruppe simuliert wurde. Des Weiteren wird die Verwendung von Minkowski-Funktionalen zur Analyse der Dynamik von geometrischen Eigenschaften der während der Phasenseparation entstehenden Strukturen erprobt.

Parallel dazu wurde ein Verfahren zur Berechnung von Lattice-Boltzmann-Gewichten basierend auf der numerischen Lösung der Maxwell-Boltzmann-Constraints entwickelt und in einem Python-Skript implementiert. Das Skript ermittelt, ob zu den vom Benutzer angegebenen Parametern ein gültiges Modell existiert und berechnet dessen Gewichte, falls dies der Fall ist. Es wird gezeigt, wie bestimmte Modelle auch dann generiert werden können, wenn es unendlich viele Lösungen für das Problem gibt. Darüber hinaus bietet das Skript eine Funktion zur Bestimmung der Gültigkeit einer gegebenen Lösung, welche zur Überprüfung bestehender Modelle aus der Literatur verwendet werden kann.

Contents

Glossary

E_{kin} kinetic energy. 66, 157

E_{pot} potential energy. 66, 157

FENE Acronym for Finitely Extensible Nonlinear Elastic. The FENE potential is used as bonded interaction in the Kremer-Grest model for polymer melts. 45, 46, 49, 105, 114, 127

F Helmholtz free energy $U - TS$. 14–16, 21–24, 42, 145, 165

GENERIC general equation for nonequilibrium reversible-irreversible coupling. 3, 4, 6, 38, 39, 42, 43, 63, 65, 78, 126, 127

$\mathscr{H}$ Hamiltonian. 25–27, 30, 31, 40–42, 67–72, 76–79, 81, 86, 154, 155, 157, 159–161, 164, 165, 170

LBE Lattice Boltzmann equation. 54

LB Lattice Boltzmann. 2, 4, 6, 45, 52, 54, 57, 60–63, 65, 73, 89, 95, 99, 101–104, 107, 111, 125–127

LCST lower critical solution temperature. 8

LJC Lennard-Jones Cosine potential. Variant of the Lennard-Jones potential in which the strength of interaction can be tuned. 50, 105

$\mathscr{L}$ Lagrangian. 66, 67

MBC Maxwell Boltzmann constraints for the weights of a Lattice Boltzmann model. 89, 90

MD Molecular Dynamics. 2, 4, 6, 45, 46, 52, 60–63, 65, 73, 101–103, 107, 111, 125–127

MRT Multiple relaxation time scheme for the Lattice Boltzmann collision operator. 54, 58–60, 62

N_{part} total number of particles in the system. 102

RW random walk. 8, 9

SAW self-avoiding walk. 9, 10

T_θ The temperature at which the solvent becomes the θ-solvent and polymer chains scale like ideal chains.. 8–10

UCST upper critical solution temperature. 7, 8

U thermodynamic internal energy. 15, 39

VEMH Viscoelastic Model H. 65

V_m Excluded volume of a monomer. 8

c_m^{**} threshold concentration between semidilute and concentrated solutions. 10

c_m^{*} critical monomer concentration at which polymer chains begin to overlap. 8, 9

c_m monomer concentration (number of particles per volume). 7

c_s speed of sound. 55, 57

$\mathbf{C}$ conformation tensor. 36, 38, 87, 88, 113, 114, 156–158

δt_LB Lattice Boltzmann time step. 54, 55, 58–60, 62

δt_MD Molecular Dynamics time step. 46, 47, 49, 52, 53, 62, 106

ε_LJ Energy parameter in the Lennard-Jones potential. 50, 51, 105

i Imaginary unit $\mathrm{i}^2 = -1$. 13, 28, 30, 31, 47, 110

k_B Boltzmann constant. 8, 10–13, 19, 21–24, 33–38, 48, 49, 52, 55, 60, 61, 82, 107, 162–165

$\mathbf{K}$ velocity gradient tensor $K_{\alpha\beta} = \partial_\beta v_\alpha$. 35–37, 87

ζ_L friction constant of the Langevin thermostat. 48, 105

LHS Left-hand side. 91, 92

$\mathscr{Z}_\mathrm{c}$ canonical partition function. 14, 21–23, 164, 165, 167

$\mathscr{Z}_\mathrm{gc}$ grand canonical partition function. 167–169

$\mathscr{Z}$ partition function. 167–171

r_c cutoff radius for MD simulation. 48, 51, 105

$\mathbf{R}_\mathrm{e}$ end-to-end vector of a polymer chain. 8

RHS Right-hand side. 91, 92, 94, 95

r_s skin thickness for MD simulation. 48, 106

σ_LJ Length parameter in the Lennard-Jones potential. 50, 51, 105

T_c critical temperature. 20

$\mathbb{1}$ unity matrix. 36–38, 87, 113, 159, 164

$\eta_{\alpha\beta\gamma\delta}$ fourth-order viscosity tensor. 55, 60, 62, 66, 77, 79

η_b bulk viscosity. 37, 55, 60, 81

η_s shear viscosity. 30–32, 37, 55, 60, 81, 86, 153, 154

ξ_c size of correlation blob. 9, 10

ξ_T size of thermal blob. 9, 10

Chapter 1

Introduction

While the phase separation of simple fluids is a well-known process that has been studied experimentally and theoretically for a long time [1–6], a range of interesting new phenomena can arise when one of the components has viscoelastic properties. The morphologies occurring in the spinodal decomposition of simple fluids are mainly determined by the interplay between surface tension, diffusion, convection, and inertia. Introducing viscoelastic effects can break the separation of time scales making the theoretical treatment challenging. One of the most prominent features of the so-called viscoelastic phase separation is the formation of long-range networks in the viscoelastic component; additional characteristic phenomena include a slowing-down of the dynamics in the early stages and the volume shrinking of the viscoelastic component at intermediate times. While being intriguing from the standpoint of fundamental research, the physical mechanisms taking place in viscoelastic phase separation are ubiquitous in industrial processes such as the production of plastic foams and filters, as well as food processing.

While some deviation from the standard behavior was already predicted theoretically by Binder et al. in 1986 [7], the characteristic freezing and network structures were first observed experimentally by Tanaka and Nishi [8] and later Aubert [9] as well as Song and Torkelson [10]. Additional studies by Tanaka and coworkers followed [11, 12], featuring a theoretical description in terms of a two-fluid model based on the works of Doi and Onuki [13] as well as Milner [14]. It was proposed that a mere dynamic asymmetry between components, i.e. when the configurational relaxation time of one component is much larger than the other's, is enough to produce viscoelastic-like phase separation [15]. For example, the characteristic network formation was observed in colloidal suspensions as well [16]. While the universal dynamic scaling is expected to break down in entangled polymer solutions due to the loss of self-similarity, the colloidal suspensions show universal but non-standard dynamic growth exponents [17]. A review on viscoelastic phase separation is provided by reference [18].

There have been numerous efforts to derive continuum theories for binary fluid mixtures with one viscoelastic component in terms of hydrodynamic two-fluid models which feature separate velocity fields for each component, or equivalently the relative velocity together with an average velocity. Pleiner and Harden have derived such equations purely relying on thermodynamics and basic symmetry principles [19]. However, this approach results in a model with many unknown parameters making its interpretation challenging. Other theories couple Oldroyd-B-like viscoelastic equations for the polymer conformation tensor to the classical hydrodynamic equations, such as the one by Taniguchi and Onuki [20]. The model by Tanaka et al. presented in references [12] and [21] is based on a similar philosophy and proposes a splitting of the polymeric stress tensor into a shear-stress and a diagonal bulk stress, which then enters the final dynamic equations merely as a scalar. Zhou, Zhang, and E have however pointed out that this model violates the second law of thermodynamics and propose a modified one in ref. [22] that does not suffer from this problem.

1.1 Objective of the thesis

This thesis was created in the framework of project C3 in the collaborative research center TRR146, which is funded by the German science foundation DFG. In this project, the phenomenon of viscoelastic phase separation is studied numerically using two distinct approaches.

On the one hand, a mesoscopic model where the polymer molecules are described by bead-spring chains and the solvent is represented on a discrete lattice is adopted. The particle dynamics is simulated with a Molecular Dynamics (MD) approach, while for the solvent the Lattice Boltzmann (LB) method is used. The components are coupled by Stokes friction forces resulting in a momentum transfer from polymer to solvent and vice versa. By setting the strength of the non-bonded attraction between beads, the solvent quality can be adjusted implicitly, and phase separation can be induced. Numerical simulations of this model were performed as part of the present thesis.

On the other hand, macroscopic models such as the one by Zhou, Zhang, and E, as well as related and modified models, are studied by a collaborating group at the Johannes Gutenberg Universität in Mainz. They are analyzed with respect to their mathematical properties such as solubility and well-posedness and simulated by using finite-elements methods.

The advantage of the mesoscopic model lies in the fact that it has a clear-cut connection to the underlying microscopic physics and can therefore be considered sound. Even though it is more efficient than all-atom simulations with explicit solvent particles, the simulation of large systems over long time periods is computationally still costly. With the finite-elements

methods, much larger length and time scales are accessible; however, the connection to microscopic physics is not as clear.

The project's ultimate goal is to use the numerical results from the mesoscopic model to calibrate the parameters of a suitable macroscopic model, and in this way obtain an efficient and well-interpretable simulation strategy for viscoelastic phase separation. This parameter matching will be facilitated by a second collaborating group at Technische Universität Darmstadt by methodologies from the field of inverse problems.

While there are some very general structural and dynamical properties like the structure factor and related dynamic scaling laws for which a comparison between both approaches is possible, it soon became evident that the available macroscopic models are not well-suited for a detailed quantitative matching of parameters. Key factors contributing to this are the somewhat arbitrary splitting of the stress tensor and phenomenological parameters that were introduced only to enforce thermodynamic consistency, which makes their interpretation difficult. Furthermore, the existing models based on an Oldroyd-B constitutive equation suffer some conceptual problems, which will be elaborated on in section 2.5. Because of this, the main focus of this thesis has been shifted from the production and analysis of numerical data and the comparison to the macroscopic model towards the development of a new viscoelastic two-fluid model with a clear connection to the mesoscopic simulation model and microscopic physics.

Several strategies for the derivation of the continuum dynamics were considered, one of them being the Rayleigh formalism based on the Onsager symmetry relations [23–25]. However, this is only applicable for purely dissipative processes and relating the transport coefficients to microscopic physics is not straightforward. It was then found that the Hamiltonian Poisson bracket formalism can bridge conservative microscopic dynamics and their continuum counterpart by representing the continuous fields in terms of the microscopic variables. This is in spirit similar to what has been done by Stark and Lubensky for nematic liquid crystals in ref. [26]. In conjunction with the GENERIC formalism [27, 28], this finally made possible the construction of the full dynamic equations involving both Hamiltonian and dissipative contributions.

1.2 Outline

The Outline of this thesis is as follows: In chapter 2, the theoretical foundation of the thesis is built, starting by outlining general static and dynamic scaling concepts of polymer solutions. Then, the theory of the phase separation of regular fluids is recapitulated in terms of Flory-Huggins theory and dissipative dynamical models such as the Cahn-Hilliard equation and model H. It is shown how a suitable equation of state for the system at hand can be obtained using Van der Waals theory. The Oldroyd-B model for viscoelastic

fluids is introduced, and its problems are discussed. Finally, the theoretical framework comprising the Hamiltonian Poisson bracket formalism and the GENERIC formalism, which are later used in the derivation of a viscoelastic two-fluid model, is introduced.

Chapter 3 encompasses the numerical methods used in the simulations. This includes a brief introduction of Molecular Dynamics methods and the descriptions of the particular potentials used in the production of the numerical results. Furthermore, the fluctuating Lattice Boltzmann method, which is employed to model the solvent, and its coupling to the MD particles are described.

The theoretical and computational developments achieved in the framework of this thesis are compiled in chapter 4. This includes the derivation of a two-fluid model aiming at modeling the continuum dynamics of viscoelastic phase separation. Furthermore, the workings of a Python script, which has been developed to automatize the calculation of weights needed for the construction of LB models, is described. This script simplifies a task that would otherwise, depending on the exact nature of the desired model, be tedious up to practically impossible to do manually.

Chapter 5 features numerical results obtained via simulation of the coupled LB/MD scheme. For two-dimensional systems, the critical attraction strength for the chain-collapse transition is estimated. The dynamic structure factor during phase separation and the derived coarsening dynamics are presented for both two- and three-dimensional systems; these results are compared to the original macroscopic model. Minkowski functionals are used as a way of describing the dynamics of geometrical properties obtained from three-dimensional configurations.

In chapter 6, the results of the thesis are summarized, and some prospect for possible future work on the topic is given.

Supplementary material, such as some information on functional derivatives, as well as the documentation of the developed software, is found in the appendix. Extensive calculations that would otherwise hinder the flow of the main text have likewise been outsourced there. Furthermore, some of the work that does not directly follow the mainline of the thesis but might still be of interest can be found in this place.

Chapter 2

Theoretical background

In this chapter, the fundamental physics of polymer solutions and their phase-separation behavior is introduced. Furthermore, the theoretical tools and concepts needed to derive the viscoelastic phase-separation model in section 4.1 are presented.

Section 2.1 starts by reviewing static and dynamic properties of polymer solutions in general by discussing scaling laws at different densities and temperatures. Flory-Huggins theory (section 2.2) gives some fundamental insight into the phase behavior of mixtures and is used to derive an equation of state. It also provides us with a criterion for the curvature of the free energy curve that determines whether or not phase separation occurs for a particular composition. An alternative equation of state is derived by Van der Waals theory (section 2.3), which explicitly accounts for a finite volume of the polymer beads, making the description of compressible fluids more realistic. This expression for the pressure can be used to complete the phase-separation model that is developed in section 4.1.

The remainder of the chapter is concerned with the field-theoretic description of phase separation dynamics. The simple energy criterion derived from Flory-Huggins theory is improved upon by including fluctuations, and it is shown that only fluctuations beyond some critical wave vector can render the system unstable (section 2.4.1). Model B, also known as the Cahn-Hilliard equation (section 2.4.2), is the most basic dynamical model for phase separation of fluids treated here. It does however neglect hydrodynamic advection, which is accounted for in Model H (section 2.4.3). Dynamic scaling laws for the growth of characteristic length scales are motivated; they are later compared to numerical results obtained from the dynamic structure factor in section 5.4.

If one of the fluid components is a polymer, viscoelastic degrees of freedom need to be included as well. One of the most basic rheological models for a solution is the Oldroyd-B model, which is described in section 2.5. Here the polymer molecules are approximated

by harmonic dumbbells, which are subject to thermal fluctuations and friction with respect to a solvent background. The orientation of molecules is described by the average conformation tensor. This dynamic average is evaluated using the Fokker-Planck equation for the probability density of the end-to-end vector resulting in a relaxation equation for the conformation tensor towards its thermal equilibrium value. This then serves as a basis for the derivation of the hydrodynamic stress tensor. While the underlying microscopic picture is in spirit very similar to the one used to construct the novel model in section 4.1, the treatment of fluctuations is problematic for reasons which are elaborated on in the text.

To derive the novel phase separation model, a different strategy is pursued: Starting from a microscopic picture inspired by the LB/MD simulation model described in section 3.2.4, the conservative and dissipative parts of the dynamics are treated separately and without fluctuations. Once the full equations are constructed, there is the possibility to add fluctuations transparently and consistently, treating all variables on the same footing. The derivation of these equations is carried out in such a way that basic principles of nonequilibrium thermodynamics are automatically satisfied. One crucial principle is encoded in the Onsager reciprocal relations introduced in section 2.6, which require the symmetry of the matrix of transport coefficients of dissipative processes.

On the other hand, the conservative dynamics can be described by an antisymmetric matrix of Poisson brackets (section 2.7), a representation that can be derived from Hamiltonian mechanics. In the GENERIC formalism (section 2.8), conservative and dissipative dynamics are modeled in one unified theory with two distinct operators. These operators are designed in such a way that the conservative part has the Poisson bracket structure while the dissipative part satisfies Onsager's symmetry relations and the second law of thermodynamics.

In section 4.1, the Poisson bracket formalism introduced here is used to derive the conservative dynamics from a suitably constructed Hamiltonian. A dissipation rate is derived from a microscopic dissipative coupling inspired by the LB/MD coupling from section 3.2.4. Comparing this to the GENERIC dissipation rate yields the dissipative part of the dynamics. The pressure in the complete compressible model must follow a thermodynamic equation of state for which the expression obtained from Van der Waals theory is a sound choice.

2.1 Polymer solutions

The phase behavior of polymer solutions is very diverse. Here we try to give a brief overview of the statics and dynamics of polymer solutions focusing on scaling relations. We mainly rely on the references [29–32], where the interested reader can find more information.

Depending on the value of the monomer concentration c_m and temperature T, a polymer solution can be classified as dilute, semi-dilute or concentrated as shown in the phase diagram in fig. 2.1.

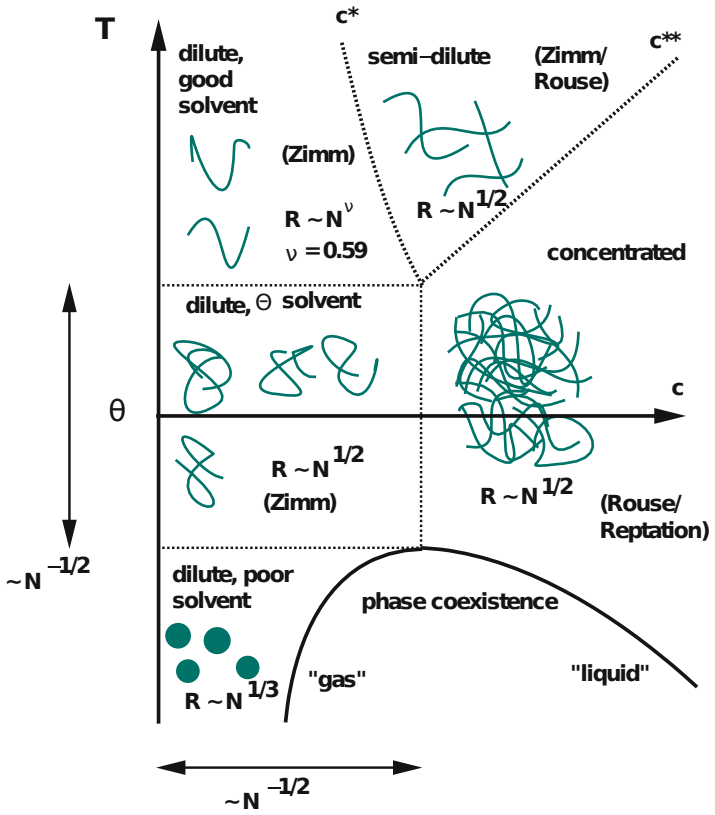


Figure 2.1: General temperature-concentration phase diagram of a polymer solution. Source: [32], with kind permission of the author.

2.1.1 Statics

Typically, polymer chains collapse at low temperatures or poor solvents and tend to demix inside the phase coexistence curve, which is also called the binodal. The temperature at the extremum of the binodal is then called upper critical solution temperature (UCST). Above the UCST all compositions of polymer and solvent are typically miscible. However, certain

polymers-solvent combinations can demix at high temperatures. The lowest temperature of this upper phase coexistence region is called lower critical solution temperature (LCST).

Apart from temperature, the solvent quality can also be related to the excluded volume of a monomer V_m. If $U(r)$ is the effective interaction energy between two monomers at distance r, including interactions with the surrounding solvent, the excluded volume is defined by

$$V_\mathrm{m} = \int \mathrm{d}^d\mathbf{r} \left(1 - \exp\left(\frac{-U(r)}{k_\mathrm{B}T}\right)\right). \tag{2.1}$$

It is positive if repulsive interactions dominate and negative if attractive interactions dominate.

The theta temperature T_θ marks the point at which the repulsive interaction of the solvent is just strong enough to cancel the hard-core repulsion in the polymer chains and $V_\mathrm{m} = 0$. In this case one uses the term *theta solvent*. Then, a polymer with N monomers behaves like a random walk (RW)* at all concentrations and the chain size scales like:

$$\left\langle R_\mathrm{g}^2 \right\rangle \propto \left\langle R_\mathrm{e}^2 \right\rangle \propto N \tag{2.2}$$

in the limit of large N. The angular brackets $\langle\cdot\rangle$ denote the ensemble average and we have introduced the end-to-end vector

$$\mathbf{R}_\mathrm{e} = \mathbf{r}_N - \mathbf{r}_1, \tag{2.3}$$

where $\mathbf{r}_i$ is the position of bead i. The radius of gyration R_g is in some sense the average distance of the beads from the chain's center of mass, which is expressed by

$$R_\mathrm{g}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{r}_i - \frac{1}{N}\sum_{j=1}^{N}\mathbf{r}_j\right)^2 = \frac{1}{2N^2}\sum_{i=1}^{N}\left(\mathbf{r}_i - \mathbf{r}_j\right)^2. \tag{2.4}$$

The ideal scaling at the theta point is exploited to determine the collapse transition for two-dimensional polymer solutions by numerical simulations in section 5.2.

Let us now define the concentration $c_\mathrm{m}^*(T)$ at which the system becomes dense enough so that individual chains begin to interact (overlap limit). We estimate c_m^* as the concentration at which there are N monomers per volume of a chain. In the theta solvent the chains always scale like random walks, independent of the concentration. Hence

$$c_\mathrm{m}^*(T_\theta) \sim \frac{N}{\langle {R_\mathrm{e}}^2\rangle^{3/2}} \propto N^{-1/2}. \tag{2.5}$$

*There are small corrections to scaling due to three-body forces.

Here we use the sign '$\sim$' to indicate proportionality with dimensionless constant prefactor and '$\propto$' to indicate proportionality with a factor that has units.

In a *good solvent* $T > T_\theta$, there is positive excluded volume, however monomers of the same chain only begin to 'see' each other at a length-scale beyond the thermal blob size ξ_T. The monomers practically do not interact inside the thermal blobs, and the subchains in the blobs are ideal with scaling exponent 1/2. The thermal blobs themselves, however, form a self-avoiding random-walk (SAW) with exponent $\nu \approx 0.59$. In the limit of an athermal solvent ($T \to \infty$), ξ_T becomes identical to the monomer size b, and the chain behaves like a SAW on all scales. The overlap concentration then scales like

$$c_{\mathrm{m}}^*(T > T_\theta) \propto N^{1-3\nu}. \tag{2.6}$$

As temperature increases, the total size of the chains increases as well, meaning that c_{m}^* must decrease as the system heats up.

At concentrations $c > c_{\mathrm{m}}^*$ individual chains start to interact and the solution becomes *semidilute*. While at small length scales, the monomers interact primarily with the solvent and with monomers of the same chain, other chains become 'visible' beyond a certain correlation length ξ_c. This motivates the definition of correlation blobs with size ξ_c. Like in a melt, excluded volume interactions are screened on length scales larger than the correlation blobs and the chains as a whole scale like random walks. Thus there is a hierarchy of blobs, where inside each correlation blob there is a chain of thermal blobs with SAW configuration, while the subchains in the thermal blobs again scale like an RW (fig. 2.2).

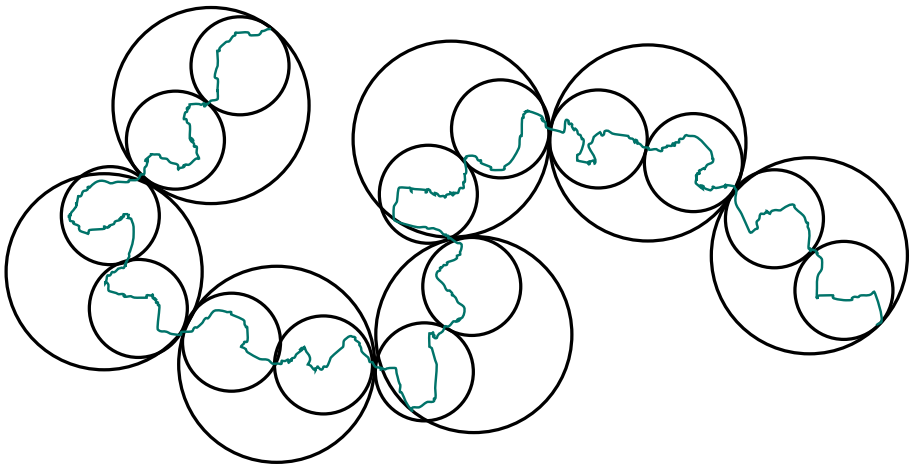


Figure 2.2: Blob hierarchy in the semidilute regime. The large circles are the correlation blobs forming a random walk. Inside are the thermal blobs with SAW structure. Within the thermal blobs, the chain performs a random walk.

While the size of the thermal blobs ξ_T is independent of concentration, the correlation blobs become smaller as the density increases. By c_{m}^{**} we denote the concentration at which $\xi_c = \xi_T$, no SAW conformations are left, and the system becomes concentrated.

In *poor solvents* $T < T_\theta$, the thermal blobs form compact globules whose size scales like $N^{1/3}$. Within these globules, there might still be solvent molecules; then, the scaling inside the thermal blobs is still ideal. Only in the limit of a non-solvent, all solvent is expelled from the globule, ξ_T becomes the monomer size b, and the chain is compact on all scales. Also, when the binodal is crossed, the collapse is accompanied by phase separation (see sections 2.2 and 2.4). Then the system decomposes into a polymer-rich phase where the chains essentially form a melt and a dilute phase with some globules suspended in the solvent.

In principle, mixtures of solvents can also be considered. This can lead to interesting effects like the swelling of chains in a mixture of two poor solvents in which the chains would normally be collapsed [33].

2.1.2 Dynamics

To describe dynamics, we first consider single particles in a solution. The particles undergo stochastic kicks from the surrounding solvent, leading to diffusive motion. On time scales much larger than the average time between collisions, the mean squared displacement of a particle is linear in time, $\langle(\mathbf{r}(t) - \mathbf{r}(0))^2\rangle = 2dDt$. Here, D is the diffusion constant and d the spacial dimension. According to the fluctuation-dissipation theorem, there must be a compensating friction force with coefficient

$$\zeta = k_{\mathrm{B}}T/D \tag{2.7}$$

acting on the particle. Equation (2.7) is known as the Einstein relation.

In *concentrated* systems, hydrodynamic interactions are screened, and the Rouse model is a good choice to describe the dynamics of polymer chains. Here, polymers are modeled as beads connected by springs. One bead can be thought of as a Kuhn segment in a real chain, i.e. a group of consecutive monomers large enough that the resulting chain of Kuhn segments can be considered ideal. The surrounding solvent exerts a friction force with coefficient ζ onto all N beads. Thus, the full dynamics is described by a Langevin equation with a diffusion constant that scales like $D_{\mathrm{R}} \propto 1/N$. The time it takes for a chain to cover a distance equal to its own size R is thus given by

$$\tau_{\mathrm{R}} \sim \frac{R^2}{D} = \frac{\zeta R^2}{k_{\mathrm{B}}T}N \sim \frac{\zeta b^2}{k_{\mathrm{B}}T}N^2, \tag{2.8}$$

which is also called the Rouse time [34].

Hydrodynamic interactions cannot be neglected in *dilute* systems. Moving beads introduce flow in the solvent which propagates through the system with a velocity field that decays like $\propto 1/r$ and interacts with other beads. This is accounted for in the Zimm model which has a diffusion constant $D \propto 1/N^{\nu}$. It can be shown [35] that the Zimm time scales like

$$\tau_{\mathrm{Z}} \sim \frac{\zeta b^2}{k_{\mathrm{B}}T} N^{3\nu}. \tag{2.9}$$

In *semidilute* systems, both Rouse and Zimm behavior can occur. While it is still commonly assumed that both regimes can be separated purely by length scale, it has been shown [36] that the time scale must be taken into consideration as well. Zimm behavior is always observed on short time scales. Only on time scales larger than the time needed for a correlation blob to move a distance equal to its own size, the chain begins to feel the constraint by the surrounding molecules. Then, on length scales larger than the hydrodynamic screening length ξ_h, there are no more effects of hydrodynamic interaction, and Rouse dynamics applies.

In the limit of very low solvent concentration, i.e. *melts*, surrounding chains form a network of tubes with a diameter of d_{t}, through which the molecules can move via reptation. Inside the tubes the chain is able to form blobs of size $d_{\mathrm{t}} = bN_{\mathrm{e}}^{1/2}$ containing N_{e} beads that describe a random walk. Hence, the average contour length of a tube is

$$L = \frac{N}{N_{\mathrm{e}}} d_{\mathrm{t}} = b \frac{N}{N_{\mathrm{e}}^{1/2}}. \tag{2.10}$$

Assuming that inside a tube the chains can diffuse freely with diffusion constant $D = k_{\mathrm{B}}T/(N\zeta)$, the reptation time scales with the cube of the chain length:

$$\tau_{\mathrm{r}} \sim \frac{L^2}{D} = \frac{\zeta L^2}{k_{\mathrm{B}}T} N = \frac{\zeta b^2}{k_{\mathrm{B}}T} \frac{N^3}{N_{\mathrm{e}}}. \tag{2.11}$$

2.2 Flory-Huggins theory

Flory-Huggins theory [30, 31, 37–40] is a classic theory describing the phase behavior of mixtures.

Most often in the derivation of Flory-Huggins theory, the configurational entropy of a mixture is estimated by attempting to count the number of ways in which its components can be arranged on a regular lattice. Together with a mean-field approximation, this is then used to construct the free energy of the mixture.

Here we will start by deriving the Edwards equation for a random walk on a lattice and in an external potential. The solution of this equation for constant potentials allows us to construct an expression for the free energy of the mixture and a continuum limit removes the need for a lattice interpretation.

2.2.1 Random walk in an external potential

This derivation of the free energy for a random walk in an external potential is based on the work of de Gennes in [30], as well as the lecture notes by B. Dünweg [31]. We start by defining the Green's function for a M-step random walk on a d-dimensional cubic lattice with lattice constant a in an external potential $U(\mathbf{r})$, starting at position $\mathbf{r}_0$ and ending at position $\mathbf{r}_M$:

$$G_M(\mathbf{r}_0, \mathbf{r}_M) = \frac{\mathrm{e}^{\beta U(\mathbf{r}_0)}}{z^M} \sum_{W_M(\mathbf{r}_0, \mathbf{r}_M)} \exp[-\beta(U(\mathbf{r}_0) + U(\mathbf{r}_1) + \cdots + U(\mathbf{r}_M))], \tag{2.12}$$

$$G_0(\mathbf{r}_0, \mathbf{r}_M) = \delta_{\mathbf{r}_0, \mathbf{r}_M}. \tag{2.13}$$

Here $\beta = 1/(k_\mathrm{B}T)$, $W_M(\mathbf{r}_0, \mathbf{r}_M)$ is the set of all possible M-step random walks from $\mathbf{r}_0$ to $\mathbf{r}_M$ and z is the number of nearest neighbors of a lattice site. For $\mathbf{s}$ being a nearest neighbor of $\mathbf{r}$, $\mathbf{s} \in \mathrm{nn}(\mathbf{r})$, we can expand

$$G_M(\mathbf{r}_0, \mathbf{s}) \approx G_M(\mathbf{r}_0, \mathbf{r}) + (s_\alpha - r_\alpha)\partial_\alpha G_M + \frac{1}{2}(s_\alpha - r_\alpha)(s_\beta - r_\beta)\partial_\alpha \partial_\beta G_M, \tag{2.14}$$

where Einstein sum convention is applied for repeated Greek indexes and the derivatives $\partial_\alpha = \partial/\partial r_\alpha$ are always acting on the second argument of G_M. Because of lattice symmetry, the expression

$$\sum_{\mathbf{s} \in \mathrm{nn}(\mathbf{r})} (s_\alpha - r_\alpha)(s_\beta - r_\beta) = \delta_{\alpha\beta} \frac{z a^2}{d} \tag{2.15}$$

has no off-diagonal components and we can write the sum over all nearest neighbors of $\mathbf{r}$ as

$$\sum_{\mathbf{s} \in \mathrm{nn}(\mathbf{r})} G_M(\mathbf{r}_0, \mathbf{s}) \approx z G_M(\mathbf{r}_0, \mathbf{r}_M) + \frac{z a^2}{2d} \nabla^2 G_M. \tag{2.16}$$

We now express the Green's function of a chain that is extended by one bead by multiplying with the one-bead Green's function and summing over the nearest neighbors of $\mathbf{r}$:

$$\begin{aligned} G_{M+1}(\mathbf{r}_0,\mathbf{r}) &= \sum_{\mathbf{s}\in\mathrm{nn}(\mathbf{r})} G_M(\mathbf{r}_0,\mathbf{s})G_1(\mathbf{s},\mathbf{r}) = \frac{\mathrm{e}^{-\beta U(\mathbf{r})}}{z}\sum_{\mathbf{s}\in\mathrm{nn}(\mathbf{r})} G_M(\mathbf{r}_0,\mathbf{s}) \\ &\approx \frac{\mathrm{e}^{-\beta U(\mathbf{r})}}{z}\left[zG_M(\mathbf{r}_0,\mathbf{r}) + \frac{za^2}{2d}\nabla^2 G_M\right] \\ &\approx \frac{1-\beta U(\mathbf{r})}{z}\left[zG_M(\mathbf{r}_0,\mathbf{r}) + \frac{za^2}{2d}\nabla^2 G_M\right] \\ &\approx G_M(\mathbf{r}_0,\mathbf{r}) + \frac{a^2}{2d}\nabla^2 G_M - \beta U(\mathbf{r})G_M(\mathbf{r}_0,\mathbf{r}). \end{aligned} \tag{2.17}$$

The expansion of the exponential is done under the assumption that the interaction with the potential is small compared to $k_\mathrm{B}T$ and thus $\beta U \ll 1$. Also we consider G_M to be slowly varying on the scale of a lattice constant a. Therefore, terms proportional to $\beta U a^2\nabla^2 G_M$ are discarded as they are small of second order. Note that we have switched to a continuum interpretation for the spacial variables. By forming the difference quotient we switch to a continuum picture in M as well and retrieve the differential equation

$$\frac{\partial G_M(\mathbf{r}_0,\mathbf{r})}{\partial M} = \left[\frac{a^2}{2d}\nabla^2 - \beta U(\mathbf{r})\right]G_M(\mathbf{r}_0,\mathbf{r}), \tag{2.18}$$

which has the same general form as the Schrödinger equation of quantum mechanics. Here we shall call it Edwards equation. In the continuum limit the initial condition eq. (2.13) becomes

$$G_0(\mathbf{r}_0,\mathbf{r}_M) = a^d\delta(\mathbf{r}_M - \mathbf{r}_0). \tag{2.19}$$

Together with eq. (2.18) it determines G_M uniquely. When taking the continuum limit, the parameter a can be reinterpreted as a bond length. This also implies that the continuum picture is only meaningful on length scales much larger than a.

For potentials that are constant in space, G_M must be translationally invariant. We may then write $G_M(\mathbf{r}_0,\mathbf{r}_M) =: G_M(\mathbf{r}_M - \mathbf{r}_0)$ and set $\mathbf{r}_0 = 0$ without loss of generality. The Edwards equation can then be solved in Fourier space. With the Fourier transform defined by

$$\tilde{G}_M(\mathbf{q}) = \int \mathrm{d}^d\mathbf{r}\,\mathrm{e}^{-\mathrm{i}\mathbf{q}\cdot\mathbf{r}}G_M(\mathbf{r}), \tag{2.20}$$

$$G_M(\mathbf{r}) = \frac{1}{(2\pi)^d}\int \mathrm{d}^d\mathbf{q}\,\mathrm{e}^{\mathrm{i}\mathbf{q}\cdot\mathbf{r}}\tilde{G}_M(\mathbf{q}), \tag{2.21}$$

eq. (2.18) and the initial condition eq. (2.19) can be transformed to

$$\frac{\partial \tilde{G}_M(\mathbf{q})}{\partial M} = -\left(\frac{a^2}{2d}q^2 + \beta U\right)\tilde{G}_M(\mathbf{q}), \tag{2.22}$$

$$\tilde{G}_0 = a^d. \tag{2.23}$$

It is easily seen that the function

$$\tilde{G}_M(\mathbf{q}) = a^d \exp\left[-M\left(\frac{a^2}{2d}q^2 + \beta U\right)\right] \tag{2.24}$$

is the unique solution of this problem. Transformation back into real space via Gaussian integration results in the final expression for the Green's function in a constant external potential

$$G_M(\mathbf{r}) = \left(\frac{d}{2\pi M}\right)^{d/2} \mathrm{e}^{-M\beta U} \exp\left[-\frac{d}{2Ma^2}\mathbf{r}^2\right] =: \mathrm{e}^{-\beta M U} G_M^{(0)}(\mathbf{r}). \tag{2.25}$$

2.2.2 Free energy

We can now use the results for the Green's function to determine the partition function of a single chain in some volume V. Using a continuum limit $\sum_{\mathbf{r}} \to a^{-d}\int \mathrm{d}^d\mathbf{r}$ and dropping numerical prefactors we find

$$\begin{aligned}\mathscr{Z}_{\mathrm{c}}(1,V,T) &\sim \sum_{\mathbf{r}_0,\mathbf{r}} G_M(\mathbf{r}_0,\mathbf{r}) = \frac{1}{a^{2d}}\int \mathrm{d}^d\mathbf{r}\int \mathrm{d}^d\mathbf{r}_0\, G_M(\mathbf{r}_0,\mathbf{r}) \\ &= \frac{1}{a^{2d}}\int \mathrm{d}^d\mathbf{r}\int \mathrm{d}^d\mathbf{r}_0\, G_M(\mathbf{r}-\mathbf{r}_0) = \frac{V}{a^{2d}}\int \mathrm{d}^d\mathbf{r}\, G_M(\mathbf{r}) \\ &= \frac{V}{a^{2d}}\mathrm{e}^{-\beta U M}\int \mathrm{d}^d\mathbf{r}\, G_M^{(0)}(\mathbf{r}) = \frac{V}{a^d}\mathrm{e}^{-\beta U M}.\end{aligned} \tag{2.26}$$

For N non-interacting, indistinguishable random walks, the states are multiplicative and the partition function becomes

$$\mathscr{Z}_{\mathrm{c}}(N,V,T) = \frac{1}{N!}[\mathscr{Z}_{\mathrm{c}}(1,V,T)]^N = \frac{1}{N!}\left[\frac{V}{a^d}\exp(-\beta M U)\right]^N. \tag{2.27}$$

The factor $1/N!$ originates from the fact that chains are indistinguishable, and exchanging individual chains generates an equivalent configuration. We now use eq. (2.27) to calculate the Helmholtz free energy

$$\beta F = -\log \mathscr{Z}_{\mathrm{c}} = \log(N!) + \beta N M U - N\log\left(\frac{V}{a^d}\right). \tag{2.28}$$

For large N we can use Stirling's formula $\log(N!) \approx N\log(N) - N$ and approximate

$$\begin{aligned}\beta F(N,V,T) &\approx N\log(N) - N + \beta NMU - N\log\left(\frac{V}{a^d}\right)\\ &= -N + \beta NMU + N\log\left(\frac{Na^d}{V}\right).\end{aligned} \tag{2.29}$$

With the volume fraction $\phi(N,V) = NMa^d/V$ and the free energy per lattice site $f = Fa^d/V$, we can write

$$\beta f(N,V,T) = \beta U\phi + \frac{\phi}{M}\left(\log\frac{\phi}{M} - 1\right). \tag{2.30}$$

For a two-component mixture of polymers 1 and 2 with corresponding chain lengths M_1, M_2 and volume fractions ϕ_1, ϕ_2, the free energies are additive:

$$\begin{aligned}\beta f_{\mathrm{FH}}(N_1,N_2,V,T) &= \beta f_1(N_1,V,T) + \beta f_2(N_2,V,T)\\ &= \beta U_1\phi_1 + \frac{\phi_1}{M_1}\left(\log\frac{\phi_1}{M_1} - 1\right) + \beta U_2\phi_2 + \frac{\phi_2}{M_2}\left(\log\frac{\phi_2}{M_2} - 1\right).\end{aligned} \tag{2.31}$$

The internal energies U_1 and U_2 contain both equal-species contributions and different-species contributions. By mean-field approximation, both contributions to the U_i are assumed to be proportional to the respective volume fractions:

$$U_1 = \phi_1 u_{11} + \phi_2 u_{12}, \qquad U_2 = \phi_1 u_{21} + \phi_2 u_{22}. \tag{2.32}$$

The interaction coefficients u_{ij} are symmetrical and proportional to the typical number of nearest neighbors in the dense phase as well as to the strength of interaction at distance a. The free energy then becomes

$$\begin{aligned}\beta f_{\mathrm{FH}}(N_1,N_2,V,T) = \frac{\phi_1}{M_1}\left(\log\frac{\phi_1}{M_1} - 1\right) + \frac{\phi_2}{M_2}\left(\log\frac{\phi_2}{M_2} - 1\right)\\ + \beta\left(\phi_1^2 u_{11} + \phi_2^2 u_{22} + 2\phi_1\phi_2 u_{12}\right).\end{aligned} \tag{2.33}$$

Equation (2.33) can easily be generalized to an arbitrary number of components:

$$\beta f_{\mathrm{FH}}(\{N_k\},V,T) = \sum_i \left[\frac{\phi_i}{M_i}\left(\log\frac{\phi_i}{M_i} - 1\right) + \beta\phi_i\sum_j \phi_j u_{ij}\right]. \tag{2.34}$$

From eq. (2.34) we can directly calculate an equation of state for the pressure by taking the derivative

$$\begin{aligned}\beta P(\{N_k\}, V, T) &= -\beta\left(\frac{\partial F_{\mathrm{FH}}}{\partial V}\right)_{\{N_k\},T} = -\frac{1}{a^d}\frac{\partial}{\partial V}(V\beta f_{\mathrm{FH}})_{\{N_k\},T} \\ &= -\frac{1}{a^d}\left[\beta f_{\mathrm{FH}} + \beta V\left(\frac{\partial f_{\mathrm{FH}}}{\partial V}\right)_{\{N_k\},T}\right].\end{aligned} \tag{2.35}$$

With $(\partial\phi_i/\partial V)_{\{N_k\},T} = -\phi_i/V$ the derivative is

$$\beta V\left(\frac{\partial f_{\mathrm{FH}}}{\partial V}\right)_{\{N_k\},T} = -\sum_i\left[\frac{\phi_i}{M_i}\log\frac{\phi_i}{M_i} + 2\beta\phi_i\sum_j\phi_j u_{ij}\right] \tag{2.36}$$

and the pressure becomes

$$\beta a^d P(\{N_k\}, V, T) = \sum_i\left[\frac{\phi_i}{M_i} + \beta\phi_i\sum_j\phi_j u_{ij}\right]. \tag{2.37}$$

The first term is just the expression from the ideal gas, while the second term accounts for interaction effects.

Of course, the total volume fraction cannot exceed one, $\sum_i\phi_i \leq 1$, and we can define the pressure to be infinity at that point. However, this does not reflect the behavior of a Lennard-Jones-like fluid very well. Here one would expect a continuous rise of the pressure towards higher and higher densities. We thus conclude the equation of state from Flory-Huggins theory as derived here is not well suited for our purposes. There is also the possibility to determine the equation of state of a compressible binary mixture by introducing a third, non-interacting component termed voids (c.f. appendix A.4). In this model, the pressure has a divergence at the finite volume $V = NMa^d$; however, the physical interpretation of the voids is somewhat obscure.

2.2.3 Phase behavior

In the following, we consider an incompressible binary mixture of components 1 and 2 and investigate its phase behavior. With the incompressibility condition $\phi_1 = 1 - \phi_2 =: \phi$, the sum of interaction energies can be rewritten as

$$U_1\phi_1 + U_2\phi_2 = (2u_{12} - u_{11} - u_{22})\phi(1-\phi) + \phi u_{11} + (1-\phi)u_{22}. \tag{2.38}$$

With the definition of the Flory-Huggins interaction parameter

$$\chi = \beta(2u_{12} - u_{11} - u_{22}), \tag{2.39}$$

the free energy becomes

$$\beta f_{\mathrm{FH}}(N_1, V, T) = \frac{\phi}{M_1}\left[\log\left(\frac{\phi}{M_1}\right) - 1\right] + \frac{1-\phi}{M_2}\left[\log\left(\frac{1-\phi}{M_2}\right) - 1\right] + \chi\phi(1-\phi) + \beta(\phi u_{11} + (1-\phi)u_{22}). \tag{2.40}$$

As we shall see below, constant contributions to the free energy and contributions linear in ϕ do not alter phase behavior. Hence it is sufficient to use the reduced form

$$\beta f_{\mathrm{FH}}(N_1, V, T) = \frac{\phi}{M_1}\log(\phi) + \frac{1-\phi}{M_2}\log(1-\phi) + \chi\phi(1-\phi), \tag{2.41}$$

which is plotted in (fig. 2.3).

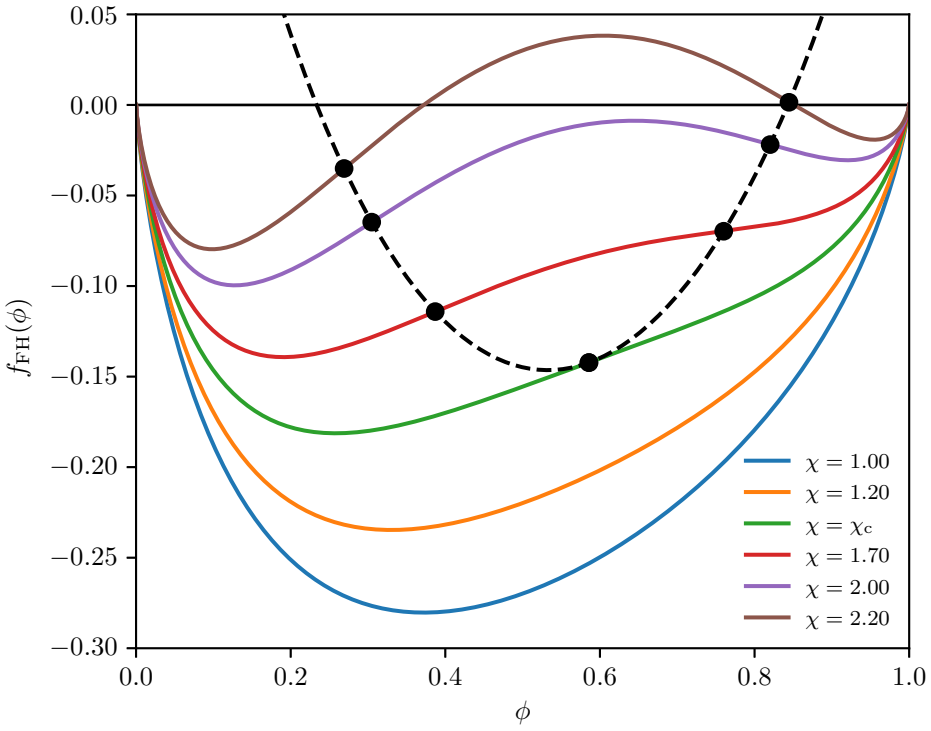


Figure 2.3: Flory-Huggins free energy eq. (2.41) with $M_1 = 1$, $M_2 = 2$ for varying value of the interaction parameter χ. The critical value for the interaction parameter is $\chi_c \approx 1.46$. The dashed line follows the inflection points and defines the spinodal.

In case of phase separation, a system with an overall species-1 volume fraction of ϕ_0 will separate into two subsystems I and II: One subsystem of volume $V_{\mathrm{I}} = \lambda V$, $0 < \lambda < 1$, where species 1, without loss of generality, takes a fraction of $\phi_{\mathrm{I}} \leq \phi_0$ of the subsystem

volume, and one subsystem of volume $V_{\mathrm{II}} = (1-\lambda)V$, where species 1 has a volume fraction of $\phi_{\mathrm{II}} \geq \phi_0$. The global volume fraction of species 1 is then $\phi_0 = \lambda\phi_{\mathrm{I}} + (1-\lambda)\phi_{\mathrm{II}}$. In the phase-separated state, the total free energy of the system is then given by the sum of the free energies of both subsystems:

$$f_{\mathrm{FH,sep}}(\phi_{\mathrm{I}}, \phi_{\mathrm{II}}) = \lambda f_{\mathrm{FH}}(\phi_{\mathrm{I}}) + (1-\lambda) f_{\mathrm{FH}}(\phi_{\mathrm{II}}). \tag{2.42}$$

For phase separation to occur in the first place, the free energy of the unmixed state must be lower than that of the mixed state. We can write this as a condition for the excess free energy of mixing

$$\Delta f_{\mathrm{mix}} := f_{\mathrm{FH}}(\phi_0) - f_{\mathrm{FH,sep}}(\phi_{\mathrm{I}}, \phi_{\mathrm{II}}) \overset{!}{>} 0. \tag{2.43}$$

This condition can always be fulfilled in intervals where the function $f_{\mathrm{FH}}(\phi)$ is concave, i.e.

$$\left.\frac{\partial^2 f_{\mathrm{FH}}}{\partial \phi^2}\right|_{\phi_0} < 0, \tag{2.44}$$

as it is illustrated in fig. 2.4. It is never fulfilled in regions where f_{FH} is convex. Points

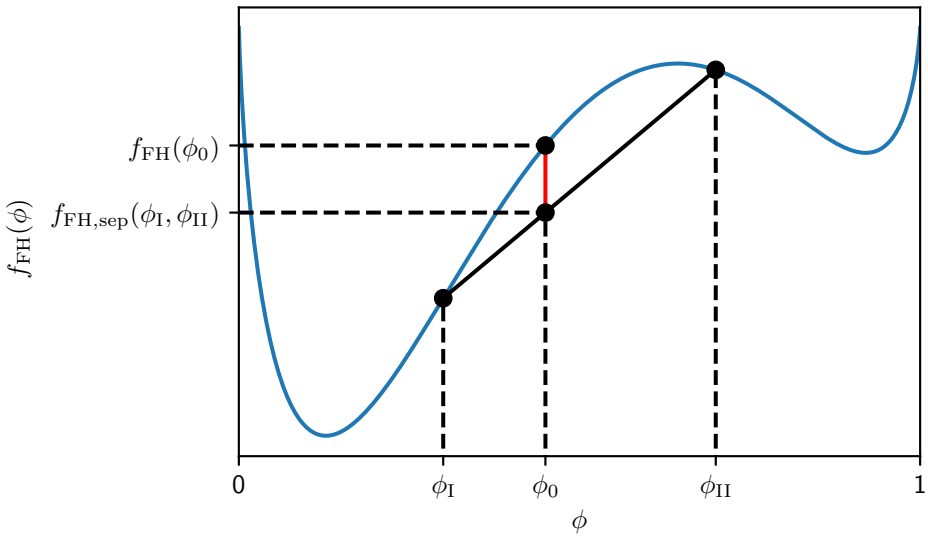


Figure 2.4: Flory-Huggins free energy from eq. (2.41) with $M_1 = 1$, $M_2 = 2$ and $\chi = 3$ (blue). The red line indicates the free energy difference between mixed and separated state. The solid black line is parametrized by the equation $f(\lambda) = \lambda f_{\mathrm{FH}}(\phi_{\mathrm{I}}) + (1-\lambda) f_{\mathrm{FH}}(\phi_{\mathrm{II}})$.

at which the free energy is concave are unstable, i.e. even small fluctuations in ϕ lead to a reduction in free energy and thus phase separation. In this case, the process is called *spinodal decomposition* (c.f. section 2.4). The set of inflection points is called the spinodal

(dashed line in fig. 2.3). It is the boundary of the unstable regime and is usually drawn in the ϕ-χ or ϕ-T plane.

Furthermore, there exists a metastable regime outside the spinodal. In this regime, small fluctuations are not enough to generate a state with lower energy. However, the free energy barrier can be overcome via nucleation and growth once the bulk energy of a droplet surpasses its surface energy. Once a critical droplet size is reached, phase separation takes place. The metastable regime is separated from the stable, one-phase regime by the binodal.

Generally the optimal compositions in the sense that Δf_{mix} is maximal, are those ϕ_{I}, ϕ_{II} which have a common tangent

$$\left.\frac{\partial f_{\mathrm{FH}}}{\partial \phi}\right|_{\phi_{\mathrm{I}}} = \left.\frac{\partial f_{\mathrm{FH}}}{\partial \phi}\right|_{\phi_{\mathrm{II}}} = \frac{f_{\mathrm{FH}}(\phi_{\mathrm{II}}) - f_{\mathrm{FH}}(\phi_{\mathrm{I}})}{\phi_{\mathrm{II}} - \phi_{\mathrm{I}}}. \tag{2.45}$$

This condition is equivalent to requiring the chemical potential to be the same in both phases. The binodal can then be generated by finding $\phi_{\mathrm{I}}, \phi_{\mathrm{II}}$ for all χ. If now $f_{\mathrm{FH}} \to f_{\mathrm{FH}} + A\phi + B$, an extra term '$+A$' appears both on the left hand side and right hand side of eq. (2.45), leaving the condition invariant. This proves that it is safe to neglect constant and linear contributions to the free energy when studying the phase behavior.

The spinodal curve can easily be calculated by setting the second derivative of the free energy equal to zero,

$$\frac{1}{k_{\mathrm{B}}T}\frac{\partial^2 f_{\mathrm{FH}}}{\partial \phi^2} = \frac{1}{M_1\phi} + \frac{1}{M_2(1-\phi)} - 2\chi \stackrel{!}{=} 0, \tag{2.46}$$

and solving for χ. We find:

$$\chi(\phi) = \frac{1}{2}\left(\frac{1}{M_1\phi} + \frac{1}{M_2(1-\phi)}\right). \tag{2.47}$$

The critical value of the interaction parameter χ_{c} is the value of χ at the minimum of the spinodal, below which mixtures of all compositions are stable. We begin by calculating the derivative

$$2\frac{\partial \chi}{\partial \phi} = -\frac{1}{M_1\phi^2} + \frac{1}{M_2(1-\phi)^2} \stackrel{!}{=} 0 \tag{2.48}$$

and find that the critical volume fraction is

$$\phi_{\mathrm{c}} = \frac{\sqrt{M_2}}{\sqrt{M_1} + \sqrt{M_2}}. \tag{2.49}$$

Therefore, the critical interaction parameter is

$$\chi_c = \chi(\phi_c) = \frac{1}{2}\Big(\frac{1}{M_1} + \frac{1}{M_2}\Big) + \frac{1}{\sqrt{M_1 M_2}}. \tag{2.50}$$

For a symmetric mixture of polymers $M_1 = M_2 = M$ this becomes

$$\chi_c = \frac{2}{M}. \tag{2.51}$$

Because the interaction parameter is inversely proportional to the temperature, this defines a critical temperature at the same time: $T_c \propto 1/\chi_c$ and therefore

$$T_c \propto M, \tag{2.52}$$

explaining why polymers typically mix rather poorly with other polymers.

If one components has a chain length of $M_2 = 1$ we basically have a polymer solution and eq. (2.50) becomes

$$\chi_c = \frac{1}{2M} + \frac{1}{\sqrt{M}} + \frac{1}{2}. \tag{2.53}$$

In the limit that the other component is very long, $M \gg 1$, we find

$$\phi_c \approx \frac{1}{\sqrt{M}}, \tag{2.54}$$

$$\chi_c \approx \frac{1}{2} + \frac{1}{\sqrt{M}}. \tag{2.55}$$

This means that the critical temperature $T_c \propto (1/2 + 1/\sqrt{M})^{-1}$ increases with M but levels off at some constant value as M approaches infinity.

Note that assuming that each segment carries the same, average internal energy U neglects among others correlations due to connectivity and makes this a mean-field theory. There are ways to include these higher-order correlations. However, they lead to a much more complicated expression for the free energy, and the benefits are slim [41–43].

2.3 Van der Waals equation of state for polymers

In this section, we derive an equation of state for a system of polymer chains in continuous space while taking into account interaction energy and finite volume of the individual polymer beads. This leads to a more realistic model at high densities as compared to the equation of state derived from Flory-Huggins theory.

2.3.1 Monoatomic molecules

We start by considering a system of monoatomic molecules, i.e. simple particles. For a single point particle of mass m in a volume V, the canonical partition function is

$$\mathscr{Z}_{\mathrm{c}}(1,V,T) = z_{\mathrm{id}}(V,T) = \frac{1}{h^d}\int \mathrm{d}^d\mathbf{r}\,\mathrm{d}^d\mathbf{p}\,\exp\left(-\beta\frac{\mathbf{p}^2}{2m}\right) = \frac{V}{\Lambda^d} \tag{2.56}$$

with the thermal De-Broglie wavelength $\Lambda = \sqrt{h^2/(2\pi m k_{\mathrm{B}}T)}$. For N indistinguishable, non-interacting particles this becomes

$$\mathscr{Z}_{\mathrm{c}}(N,V,T) = \frac{\mathscr{Z}_{\mathrm{c}}(1,V,T)^N}{N!} = \frac{z_{\mathrm{id}}^N}{N!}. \tag{2.57}$$

Using Stirling's approximation, this allows us to calculate the Helmholtz free energy of the ideal gas

$$\begin{aligned}\beta F_{\mathrm{id}}(N,V,T) &= -\log\mathscr{Z}_{\mathrm{c}} = -N\log z_{\mathrm{id}} + \log(N!)\\ &\approx -N\log z_{\mathrm{id}} + N\log(N) - N.\end{aligned} \tag{2.58}$$

From here, the pressure is calculated via

$$P_{\mathrm{id}} = -\frac{\partial F_{\mathrm{id}}}{\partial V} = \frac{Nk_{\mathrm{B}}T}{z_{\mathrm{id}}}\frac{\partial z_{\mathrm{id}}}{\partial V} = \frac{Nk_{\mathrm{B}}T}{V}, \tag{2.59}$$

which is the well-known equation of state of the ideal gas.

For particles with a finite volume $B > 0$, the total volume available to the gas reduces to $V - NB$. The number of available states in the phase space reduces accordingly and in the absence of additional interactions, the single-particle partition function is

$$z = \frac{V-NB}{\Lambda^d} = z_{\mathrm{id}}\left(1-\frac{NB}{V}\right). \tag{2.60}$$

We also want to take into account attractive interactions, which come into play at finite temperatures. In a mean-field context, we expect the partition function to vary exponentially with β:

$$z = z_{\mathrm{id}}\left(1-\frac{NB}{V}\right)\mathrm{e}^{\beta U}. \tag{2.61}$$

Here $U > 0$ is the effective energy from attractive interactions per particle. The Helmholtz free energy becomes

$$F(N,V,T) = -k_\mathrm{B}T\log\left(\frac{z^N}{N!}\right) = F_\mathrm{id}(N,V,T) - k_\mathrm{B}TN\log\left(1-\frac{NB}{V}\right) - NU. \quad (2.62)$$

To further specify the interaction energy U, we make the following arguments:

(i) It must be proportional to the average particle density, $U \propto \frac{N}{V}$.

(ii) Based on a Lennard-Jones-like interaction, it must be proportional to the depth of the potential, $U \propto \varepsilon$.

(iii) It must be proportional to the interaction volume determined by the range of the interaction, $U \propto V_\mathrm{int}$.

Hence, the interaction energy takes the form

$$U = \alpha\varepsilon V_\mathrm{int}\frac{N}{V} =: A\frac{N}{V}, \quad (2.63)$$

where α is some proportionality constant and a parameter A has been defined. With these considerations, the expression for the pressure is

$$\begin{aligned} P &= P_\mathrm{id}\frac{1}{1-NB/V} + N\frac{\partial U}{\partial V} = \frac{Nk_\mathrm{B}T}{V-NB} + N\frac{\partial U}{\partial V} \\ &= \frac{Nk_\mathrm{B}T}{V-NB} - A\left(\frac{N}{V}\right)^2, \end{aligned} \quad (2.64)$$

for a plot see fig. 5.5.

2.3.2 Polymers

We will now generalize the previous derivations going from single particles to chain molecules. From eq. (2.26) in the section on Flory-Huggins theory, we know the partition function for a single random walk of length M:

$$\mathscr{Z}_\mathrm{c} = \frac{V}{a^d}\exp(-\beta MU). \quad (2.65)$$

With monomer number density $n = NM/V$ for N chains of length M, analogous argumentation for the interaction energy yields

$$MU = -\alpha\varepsilon V_\mathrm{int}\frac{NM^2}{V} = -\frac{NM^2}{V}A. \quad (2.66)$$

Taking into account a finite volume of the polymer beads, the single-chain partition function then takes the form

$$z(V,T) = \frac{1}{a^d}(V - NMB)\exp\left(\beta\frac{NM^2}{V}A\right), \tag{2.67}$$

and the total partition function becomes

$$\mathscr{Z}_{\mathrm{c}}(N,V,T) = \frac{z^N}{N!}. \tag{2.68}$$

From there, we can now calculate the Helmholtz free energy for a polymer system with N chains of length M:

$$F = Nk_{\mathrm{B}}T\left[\log\left(\frac{Na^d}{V - NMB}\right) - 1\right] - \frac{(NM)^2}{V}A. \tag{2.69}$$

With the bead number density $n(N,V) = NM/V$ the corresponding free energy density

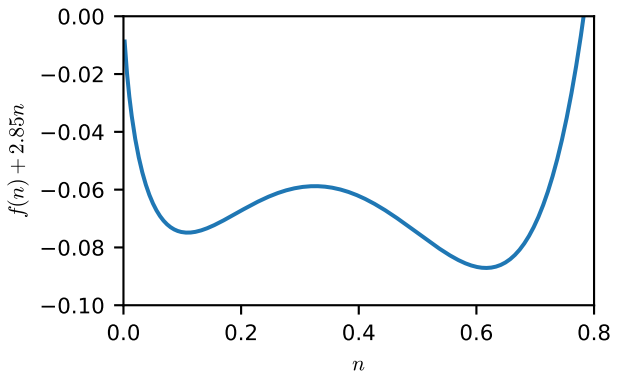


Figure 2.5: Van-der-Waals free energy density eq. (2.70) with $M = k_{\mathrm{B}}T = B = a = 1$ and $A = 4$. A linear term of $2.85n$ has been added in order to make the possibility of a common tangent construction better visible.

is defined by

$$f(N,V,T) = \frac{F}{V} = k_{\mathrm{B}}T\frac{n}{M}\left[\log\left(\frac{a^d n}{M(1-nB)}\right) - 1\right] - An^2. \tag{2.70}$$

As has been shown in section 2.2, additions to the free energy constant or linear in n do not alter the phase behavior. With a suitable linear addition, the free energy density has two local minima in some parameter range (fig. 2.5). This means that there a common tangent construction is possible, and the system is indeed in a separated state.

The associated pressure is

$$\begin{aligned} P &= -\left(\frac{\partial F}{\partial V}\right)_{N,T} = \frac{N k_{\mathrm{B}} T}{V - NMB} - A\Big(\frac{MN}{V}\Big)^2 \\ &= \frac{N k_{\mathrm{B}} T}{MN} \frac{n}{1 - Bn} - An^2. \end{aligned} \tag{2.71}$$

We see that the Van der Waals pressure has a singularity at $V = NMB$, accounting for a minimum possible volume of the system. This is a key feature that the pressure expression as derived from standard Flory-Huggins theory eq. (2.37) is lacking. Hence the Van der Waals equation of state is a good candidate to close the system of equations of the viscoelastic phase separation model derived in section 4.1. Values for the parameters A and B are estimated by numerical simulation in section 5.3.

For small n we have $n/(1 - Bn) \approx n$, $n^2 \approx 0$ and recover the equation of state of the ideal gas

$$P \approx N k_{\mathrm{B}} T \frac{n}{MN} = \frac{N k_{\mathrm{B}} T}{V}, \tag{2.72}$$

which is the limiting case of large volumes when the number of molecules is kept constant.

2.4 Phase separation dynamics

In this section, some standard dissipative models for phase separation dynamics are summarized. In doing so, information from the sources [3, 5, 6, 44] is compiled.

Model A motivates the structure of the equations but treats a non-conserved order parameter. In spinodal composition, the order parameter can, for example, be taken to be the volume fraction or density of one of the components. Because of the conservation of mass, the integral of this order parameter must likewise be conserved, and thus Model A is not a good choice to describe spinodal decomposition. The conservation of the order parameter is considered in Model B, which describes diffusive transport of the order parameter and results in the famous Cahn-Hilliard equation. In Model H, convection due to hydrodynamic flow is accounted for as well.

2.4.1 Energy criterion for spinodal decomposition

Let us consider a mixture of two fluids where the order parameter is given by the field $\phi(\mathbf{r}, t)$. This could, for example, be the density, volume fraction, concentration, etc. of one of the components. By $f_0(\phi)$ we denote the free energy density of a homogeneous configuration without any interfaces or other variations in ϕ. We are interested in isothermal systems, hence the Hamiltonian $\mathscr{H}$ must be interpreted as a Helmholtz free energy. We can in a first approximation write the free energy density as

$$f(\phi) = f_0(\phi) + \frac{\kappa}{2}(\boldsymbol{\nabla}\phi)^2, \tag{2.73}$$

where a gradient term with coefficient $\kappa > 0$ was added to energetically penalize variations in ϕ, and therefore penalize interfaces, which are sharp variations in ϕ in particular. This is a crucial ingredient since minimization of surface energy is a central driving force in the dynamics of spinodal decomposition.

We now investigate the effect of a periodic disturbance on a homogeneous system. The difference in energy density between a homogeneous system with $\phi(\mathbf{r}) = \phi_0 = \text{const.}$ and a system where ϕ is described by a plane periodic variation $\phi(\mathbf{r}) = \phi_0 + A\cos(qx)$ with x being one component of $\mathbf{r}$ is then

$$\begin{aligned} \Delta f(\phi_0, x) &= f(\phi_0 + A\cos(qx)) - f(\phi_0) \\ &\approx \frac{\kappa}{2} q^2 A^2 \sin^2(qx) + A\cos(qx) f_0'(\phi_0) + \frac{A^2}{2}\cos^2(qx) f_0''(\phi_0). \end{aligned} \tag{2.74}$$

Here we have used eq. (2.73) and expanded f_0 up to second order. Now suppose that the system has size L^d and there are periodic boundary conditions. Therefore $q = 2\pi n/L$ with

$n = 1, 2, \ldots$ The total difference in free energy is then given by the integral

$$\begin{aligned}\Delta F(\phi_0) &\approx \frac{L^{d-1}A^2}{2}\int_0^L \mathrm{d}x\left[\frac{2}{A}\cos(qx)f_0'(\phi_0) + \cos^2(qx)f_0''(\phi_0) + \kappa q^2\sin^2(qx)\right] \\ &= \frac{L^d A^2}{4}\Big[f_0''(\phi_0) + \kappa q^2\Big],\end{aligned} \tag{2.75}$$

where the trigonometric terms drop out due to the periodic boundary conditions. The homogeneous state is unstable if the variation results in a decrease in free energy. This is the case if

$$f_0''(\phi_0) < -\kappa q^2. \tag{2.76}$$

Note that this criterion does not depend on the amplitude A of the variation at all. It means that via κ, the surface tension acts as a resistance against fluctuations. This is an extension to the criterion in eq. (2.44), which was derived in the framework of Flory-Huggins mean-field theory where fluctuations are neglected. There, merely a negative second derivative was enough to pose an instability. Thermodynamically, the second derivative of the free energy with respect to the volume fraction corresponds to a compressibility. A negative compressibility means that increasing density is favorable, leading to collapse. This, in turn, leads to a further increase in density, thus rendering the system unstable. Additionally, the dependence on q shows that the system is more susceptible to variations with small wavenumbers or large wavelengths. In particular, the system is unstable wrt. to variations with a wavenumber smaller than a critical wavenumber

$$q_\mathrm{c}(\phi_0) = \sqrt{\frac{-f_0''(\phi_0)}{\kappa}} \tag{2.77}$$

as was first shown by Cahn and Hilliard in [1].

2.4.2 Model B or the Cahn-Hilliard equation

Dissipative dynamics for a non-conserved order parameter ϕ is described by the equation [3, 6, 44]

$$\frac{\partial\phi}{\partial t} = -M\frac{\delta\mathscr{H}}{\delta\phi}, \tag{2.78}$$

where by $\delta\mathscr{H}/\delta\phi$ we denote the functional derivative of the Hamiltonian (see appendix A.1) and M is an Onsager coefficient (c.f. section 2.6). At this point, one can also incorporate thermal fluctuations by using the corresponding Langevin equation; however, we shall neglect fluctuations for now. Equation (2.78) is also classified as Model A, following the nomenclature in the review of Hohenberg and Halperin [3].

In spinodal decomposition of an incompressible binary system, however, the order parameter $\phi(\mathbf{r}, t)$ can e.g. be identified with the volume fraction of one of the species. It is conserved in the sense that the integral

$$\int \mathrm{d}^d\mathbf{r}\, \phi(\mathbf{r}, t) = \phi_0 \tag{2.79}$$

is constant in time. The flux of the order parameter, $\mathbf{j}_\phi$ must therefore satisfy a continuity equation

$$\frac{\partial \phi}{\partial t} = -\boldsymbol{\nabla} \cdot \mathbf{j}_\phi. \tag{2.80}$$

According to linear Onsager theory (c.f. section 2.6 eq. (2.178)), a natural form for the current is then

$$\mathbf{j}_\phi = -M\boldsymbol{\nabla}\mu(\phi), \tag{2.81}$$

where the chemical potential μ is the functional derivative of the Hamiltonian,

$$\mu(\phi) = \frac{\delta \mathscr{H}}{\delta \phi}. \tag{2.82}$$

The 'force' generating order parameter flow is thus the chemical-potential gradient. Then, the dynamics is described by the equation

$$\frac{\partial \phi}{\partial t} = M\nabla^2\mu(\phi), \tag{2.83}$$

where ∇^2 is the Laplacian operator. Equation (2.83) is classified as Model B. For the free energy we use the ansatz from eq. (2.73):

$$\mathscr{H} = \int \mathrm{d}^d\mathbf{r} \left[f_0(\phi) + \frac{\kappa}{2}(\boldsymbol{\nabla}\phi)^2 \right]. \tag{2.84}$$

We can now use the relation for functional derivatives eq. (A.6), to evaluate

$$\frac{\delta \mathscr{H}}{\delta \phi} = \mu(\phi) = f_0'(\phi) - \kappa\nabla^2\phi. \tag{2.85}$$

Equation (2.85) together with the Model B eq. (2.83), gives us

$$\frac{\partial \phi}{\partial t} = -M\nabla^2\left[\kappa\nabla^2\phi - f_0'(\phi)\right] \tag{2.86}$$

which is the famous Cahn-Hilliard equation. Note that with the current

$$\mathbf{j}_\phi = M\boldsymbol{\nabla}\left[\kappa\nabla^2\phi - f_0'(\phi)\right], \tag{2.87}$$

eq. (2.86) has the form of a continuity equation.

We can linearize eq. (2.86) by taking the second order expansion of f_0 in order to approximate its derivative

$$\frac{\partial f_0}{\partial \phi} \approx f_0'(\phi_0) + (\phi - \phi_0) f_0''(\phi_0) = f_0'(\phi_0) - \kappa q_\mathrm{c}^2 (\phi - \phi_0), \tag{2.88}$$

resulting in the linearized form of the Cahn-Hilliard equation

$$\frac{\partial \phi}{\partial t} = -M\kappa \nabla^2 \Big[\nabla^2 \phi + q_\mathrm{c}^2 \phi\Big]. \tag{2.89}$$

Taking the Fourier transform of eq. (2.89) allows us to turn the partial differential equation into a differential equation solely in time, which can be solved in $\mathbf{q}$-space.

$$\frac{\partial}{\partial t}\tilde{\phi}(\mathbf{q}, t) = -M\kappa\Big(q^4 - q^2 q_\mathrm{c}^2\Big)\tilde{\phi}(\mathbf{q}, t), \tag{2.90}$$

$$\tilde{\phi}(\mathbf{q}, t) \propto \exp\Big[-M\kappa\Big(q^4 - q^2 q_\mathrm{c}^2\Big)t\Big]. \tag{2.91}$$

With the definition of the growth rate

$$\omega(q) = -M\kappa\Big(q^4 - q^2 q_\mathrm{c}^2\Big), \tag{2.92}$$

the amplitudes of fluctuations grow as $\exp(\omega(q)t)$. Hence, for $f_0''(\phi_0)$ positive, q_c is imaginary and they decay exponentially. For negative $f_0''(\phi_0)$ however, q_c is real and there is an interval $0 < q < q_\mathrm{c}$ for which the growth rate is positive. It is highest at the wave vector $q = q_\mathrm{c}/\sqrt{2}$ and since the growth is exponential, the corresponding length scale will quickly start to dominate the morphology of the system.

Growth law

For the dynamic structure factor, which we define as

$$\begin{aligned} S(\mathbf{q}, t) &= V \int \mathrm{d}^d\mathbf{r}\, \mathrm{e}^{-\mathrm{i}\mathbf{q}\cdot\mathbf{r}} \left\langle \phi(0, t)\phi(\mathbf{r}, t)\right\rangle \\ &= \left\langle \tilde{\phi}(\mathbf{q}, t)\tilde{\phi}(-\mathbf{q}, t)\right\rangle = \left\langle \left|\tilde{\phi}(\mathbf{q}, t)\right|^2 \right\rangle, \end{aligned} \tag{2.93}$$

eq. (2.91) leads to

$$S(\mathbf{q}, t) \propto \mathrm{e}^{2\omega(q)t}. \tag{2.94}$$

Equation (2.94) correctly shows that in the initial stages of phase separation the magnitude of the structure factor peak grows with time. However, it fails to account for the fact that the system coarsens, which would cause the position of the peak $q_{\max}$ to move towards smaller values of q.

The scaling hypothesis states that, at sufficiently late times, the distribution of domain sizes does not change when lengths are scaled by a function $L(t)$. In other words: L is the only characteristic length scale at that point. For the equal time pair correlation function inside a domain that implies the relation

$$C(\mathbf{r},t) = \langle \phi(0,t)\phi(\mathbf{r},t)\rangle \propto g\left(\frac{r}{L(t)}\right) \tag{2.95}$$

with the scaling function g. Then, eq. (2.95) translates to a structure factor of the form

$$S(q,t) \propto L(t)^d \tilde{g}(qL(t)), \tag{2.96}$$

where $\tilde{g}$ is the Fourier transform of g. With the characteristic length scale defined by

$$L(t) = \frac{2\pi}{q_{\max}(t)}, \tag{2.97}$$

we have $S(q_{\max},t) \propto L(t)^d$. When the wave vector is given in units of $q_{\max}$, $q = xq_{\max}$ and we divide by the maximum scattering intensity $S(q_{\max},t)$, we obtain a master curve that is independent of time

$$\frac{S(xq_{\max},t)}{S(q_{\max},t)} \sim \frac{\tilde{g}(2\pi x)}{\tilde{g}(2\pi)}. \tag{2.98}$$

This collapse to a master curve is shown by the simulation of a simple Lennard-Jones fluid in section 5.4. For viscoelastic fluids, however, significant deviations can be observed; see fig. 5.11.

To rigorously derive a scaling law for $L(t)$ is quite involved; however one can argue [5] that the chemical potential μ is related to the surface tension σ by $\mu \sim \sigma/L$. The current eq. (2.87) scales like $\mathbf{j}_\phi = -M\boldsymbol{\nabla}\mu \sim M\sigma/L^2$ and must be related to the interface velocity L/t. This leads us to the scaling law

$$L(t) \sim (M\sigma t)^{1/3}, \tag{2.99}$$

which is well-known as the Lifshitz-Slyozov-Wagner growth law [45, 46].

2.4.3 Model H

While Model B does take into account the conservation of the order parameter, it does not account for its hydrodynamic convection. It is thus not able to explain some of the effects observed in fluids. Consequently, the order parameter $\phi(\mathbf{r},t)$ has to be coupled to the velocity field $\mathbf{v}(\mathbf{r},t)$. Augmenting the Model B eq. (2.83) by a suitable convection term results in the Model H equation

$$\frac{\partial\phi}{\partial t} + \mathbf{v}\cdot\boldsymbol{\nabla}\phi = M\nabla^2\mu(\phi), \tag{2.100}$$

where $\mu(\phi) = \delta\mathscr{H}/\delta\phi$ is the chemical potential and the flow field $\mathbf{v}$ is governed by the incompressible Navier-Stokes equations

$$\boldsymbol{\nabla}\cdot\mathbf{v} = 0, \tag{2.101}$$

$$\rho\left(\frac{\partial\mathbf{v}}{\partial t} + (\mathbf{v}\cdot\boldsymbol{\nabla})\mathbf{v}\right) = \eta_s\nabla^2\mathbf{v} - \boldsymbol{\nabla}P - \phi\boldsymbol{\nabla}\mu, \tag{2.102}$$

with shear viscosity η_s and pressure P. The assumption that the mass density ρ is constant in time and space results in the incompressibility condition eq. (2.101). The last term in eq. (2.102) can be interpreted as a driving force on the fluid, which we denote by the force density

$$\mathbf{f} = -\phi\boldsymbol{\nabla}\mu. \tag{2.103}$$

We shall now derive a dynamic equation for ϕ in the overdamped limit where viscous forces dominate over inertial forces. The Navier-Stokes eq. (2.102) then becomes the Stokes equation

$$0 = \eta_s\nabla^2\mathbf{v}(\mathbf{r}) - \boldsymbol{\nabla}P(\mathbf{r}) + \mathbf{f}(\mathbf{r}), \tag{2.104}$$

which has the following form in Fourier space:

$$\eta_s q^2\tilde{\mathbf{v}}(\mathbf{q}) = -\mathrm{i}\mathbf{q}\tilde{P}(\mathbf{q}) + \tilde{\mathbf{f}}(\mathbf{q}). \tag{2.105}$$

The incompressibility condition $\boldsymbol{\nabla}\cdot\mathbf{v}(\mathbf{r}) = 0$ in Fourier space is just $\mathbf{q}\cdot\tilde{\mathbf{v}}(\mathbf{q}) = 0$, which allows us to determine the pressure:

$$\tilde{P} = -\frac{\mathrm{i}}{q^2}\mathbf{q}\cdot\tilde{\mathbf{f}}. \tag{2.106}$$

Inserting the expression for the pressure into eq. (2.105) results in (sum convention for repeated Greek indexes implied)

$$\eta_s q^2\tilde{v}_\alpha = -\frac{q_\alpha}{q^2}\left(q_\beta\tilde{f}_\beta\right) + \tilde{f}_\alpha = \left(\delta_{\alpha\beta} - \frac{q_\alpha q_\beta}{q^2}\right)\tilde{f}_\beta. \tag{2.107}$$

By defining the Oseen tensor in $\mathbf{q}$-space

$$\tilde{T}_{\alpha\beta}(\mathbf{q}) = \frac{1}{\eta_\mathrm{s} q^2}\left(\delta_{\alpha\beta} - \frac{q_\alpha q_\beta}{q^2}\right), \tag{2.108}$$

we can simply write

$$\tilde{v}_\alpha = \tilde{T}_{\alpha\beta}\tilde{f}_\beta. \tag{2.109}$$

The corresponding Oseen tensor in real space can be retrieved via the reverse Fourier transform

$$T_{\alpha\beta}(\mathbf{r}) = \frac{1}{(2\pi)^d \eta_\mathrm{s}} \int \mathrm{d}^d\mathbf{q}\, \frac{1}{q^2}\left(\delta_{\alpha\beta} + \frac{\partial_\alpha\partial_\beta}{q^2}\right)\mathrm{e}^{\mathrm{i}\mathbf{q}\cdot\mathbf{r}}. \tag{2.110}$$

The integral can be evaluated by using the general formula for $n > 0$[†]

$$\frac{1}{(2\pi)^d}\int \mathrm{d}^d\mathbf{q}\, \mathrm{e}^{\mathrm{i}\mathbf{q}\cdot\mathbf{r}}\frac{1}{q^n} = \frac{r^{n-d}}{c_{d,n}}, \qquad c_{d,n} = \pi^{d/2}2^n\frac{\Gamma(n/2)}{\Gamma((d-n)/2)}, \tag{2.111}$$

which in $d = 3$ dimensions results in

$$\frac{1}{(2\pi)^3}\int \mathrm{d}^3\mathbf{q}\, \frac{\mathrm{e}^{\mathrm{i}\mathbf{q}\cdot\mathbf{r}}}{q^2} = \frac{1}{4\pi r}, \qquad \frac{1}{(2\pi)^3}\int \mathrm{d}^3\mathbf{q}\, \frac{\mathrm{e}^{\mathrm{i}\mathbf{q}\cdot\mathbf{r}}}{q^4} = -\frac{r}{8\pi}. \tag{2.112}$$

Equation (2.110) then becomes

$$\begin{aligned} T_{\alpha\beta}(\mathbf{r}) &= \frac{1}{4\pi\eta_\mathrm{s}}\left(\frac{\delta_{\alpha\beta}}{r} - \frac{1}{2}\partial_\alpha\partial_\beta r\right) = \frac{1}{4\pi\eta_\mathrm{s}}\left(\frac{\delta_{\alpha\beta}}{r} - \frac{1}{2}\left[\frac{\delta_{\alpha\beta}}{r} - \frac{r_\alpha r_\beta}{r^3}\right]\right) \\ &= \frac{1}{8\pi\eta_\mathrm{s} r}\left(\delta_{\alpha\beta} + \frac{r_\alpha r_\beta}{r^2}\right). \end{aligned} \tag{2.113}$$

By using the convolution theorem for Fourier transforms, eq. (2.109) becomes

$$v_\alpha(\mathbf{r}) = \int \mathrm{d}^3\mathbf{r}'\, T_{\alpha\beta}(\mathbf{r}-\mathbf{r}')f_\beta(\mathbf{r}'), \tag{2.114}$$

meaning that the overdamped flow field is given by the convolution of the Oseen tensor with the force density. We recall that the force density is given by $f_\beta = -\phi\partial_\beta\mu$ with $\mu = \delta\mathscr{H}/\delta\phi$ and insert the above expression for the velocity into eq. (2.100). This results in the dynamic equation for ϕ,

$$\begin{aligned} \frac{\partial\phi}{\partial t} &= M\nabla^2\mu + \partial_\alpha\phi(\mathbf{r})\int \mathrm{d}^3\mathbf{r}'\, T_{\alpha\beta}(\mathbf{r}-\mathbf{r}')\phi(\mathbf{r}')\partial'_\beta\mu(\mathbf{r}') \\ &= M\nabla^2\mu - \partial_\alpha\phi(\mathbf{r})\int \mathrm{d}^3\mathbf{r}'\, \mu(\mathbf{r}')\partial'_\beta[\phi(\mathbf{r}')T_{\alpha\beta}(\mathbf{r}-\mathbf{r}')] \\ &= M\nabla^2\mu(\mathbf{r}) - \partial_\alpha\phi(\mathbf{r})\int \mathrm{d}^3\mathbf{r}'\, \mu(\mathbf{r}')T_{\alpha\beta}(\mathbf{r}-\mathbf{r}')\partial'_\beta\phi(\mathbf{r}'), \end{aligned} \tag{2.115}$$

where $\partial'_\alpha = \partial/\partial r'_\alpha$. In the first step, integration by parts was used and in the second step we exploited the fact that because of $q_\beta\tilde{T}_{\alpha\beta}(\mathbf{q}) = 0$ also $\partial_\beta T_{\alpha\beta}(\mathbf{r}) = 0$.

[†]This can be obtained by interpreting the Riesz potential as a Fourier multiplier [47].

Growth laws

Let us now consider a characteristic length scale in the coarsening process $L(t)$. Dimensional analysis for unitless ϕ can again give us an idea about how this length scale evolves in time. One can argue [5] that

$$\mu \sim \frac{\sigma}{L}, \qquad T_{\alpha\beta} \sim \frac{1}{\eta_s L}, \qquad \partial_\alpha \phi \sim \frac{1}{L}, \tag{2.116}$$

and do an estimation of the terms in eq. (2.115), which is itself $\sim t^{-1}$:

$$M\nabla^2 \mu(\mathbf{r}) \sim \frac{M\sigma}{L^3}, \tag{2.117}$$

$$\int \mathrm{d}^3\mathbf{r}'\, \mu(\mathbf{r}') T_{\alpha\beta}(\mathbf{r}-\mathbf{r}') \partial'_\beta \phi(\mathbf{r}') \partial_\alpha \phi(\mathbf{r}) \sim \frac{\sigma}{\eta_s L}. \tag{2.118}$$

The first term is diffusive in nature while the second term is convective. Hence, diffusive transport of the order parameter dominates if $L \ll \sqrt{\eta_s M}$. In this case eq. (2.117) is relevant and the length scale grows with the Lifshitz-Slyozov law

$$L(t) \sim (M\sigma t)^{1/3}, \qquad L \ll \sqrt{\eta_s M}. \tag{2.119}$$

If $L \gg \sqrt{\eta_s M}$ on the other hand, convective transport of the order parameter dominates and eq. (2.118) leads to the relation

$$L(t) \sim \frac{\sigma}{\eta_s} t, \qquad \sqrt{\eta_s M} \ll L \ll \frac{{\eta_s}^2}{\rho\sigma}, \tag{2.120}$$

also see the paper by Siggia [48]. The upper bound in eq. (2.120) comes about when the overdamped limit in eq. (2.102) is no longer valid, and the inertial terms become important. Note that the interval defined by the bounds in eq. (2.120) is only nonzero for sufficiently large viscosities. For small viscosities, the viscous hydrodynamic regime is left out, and there is a crossover to inertial dynamics right away. In this regime, the driving term $-\phi\partial_\alpha \mu \sim \sigma/L^2$ has no longer to be balanced against the viscous terms, but we instead have to compare it with the inertial terms $\rho\partial_t v_\alpha \sim \rho L/t^2$. This leads to

$$L(t) \sim \left(\frac{\sigma}{\rho}\right)^{1/3} t^{2/3}, \qquad L \gg \frac{{\eta_s}^2}{\rho\sigma}, \tag{2.121}$$

a prediction that was first made by Furukawa in [49].

To summarize, we expect a diffusion-driven $L \propto t^{1/3}$ growth in the initial phase of demixing. As the characteristic length scale grows, convection becomes more and more dominant, and there is a crossover to t^1 scaling. At the very end, inertia dominates, and the characteristic length grows as $t^{2/3}$.

2.5 Oldroyd-B model

In the previous section, we have seen a theory for the phase separation of regular, Newtonian fluids. This section presents one of the most basic models for viscoelastic fluids, the Oldroyd-B model [50–53], which is based on kinetic theory. It is used in the construction of various models for viscoelastic phase separation. However, we shall see that it is not without its problems.

In the Oldroyd-B model, a polymer molecule is represented by two beads at positions $\mathbf{r}_1$ and $\mathbf{r}_2$ with respective velocities $\mathbf{v}_1$ and $\mathbf{v}_2$. The beads are connected via springs that produce antisymmetric forces $\mathbf{F}(-\mathbf{r}) = -\mathbf{F}(\mathbf{r})$. In this way, both the extension and orientation of the molecules are captured. We assume that the individual dumbbells do not interact with each other directly. Instead, we imagine a solvent flow field $\mathbf{v}^{(\mathrm{s})}(\mathbf{r}, t)$ in the background (Figure 2.6), which interacts with the beads via Stokes friction forces with constant ζ. Thermal fluctuations are modeled by random forces, which means that the

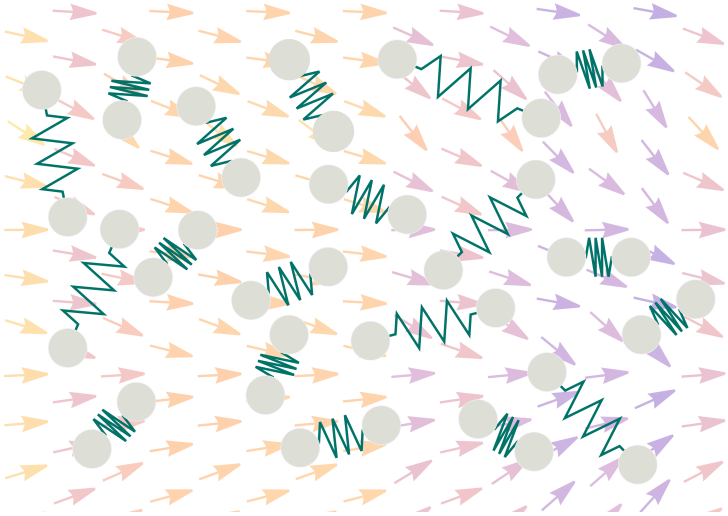


Figure 2.6: Dumbbell molecules with solvent flow field in the background.

dynamics of the dumbbell ends is described by the Langevin equations

$$m\partial_t \mathbf{v}_1 = -\zeta[\mathbf{v}_1 - \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_1)] + \mathbf{F}(\mathbf{r}_1 - \mathbf{r}_2) + \sqrt{k_\mathrm{B}T\zeta}\,\mathbf{w}_1(t), \tag{2.122}$$

$$m\partial_t \mathbf{v}_2 = -\zeta[\mathbf{v}_2 - \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_2)] - \mathbf{F}(\mathbf{r}_1 - \mathbf{r}_2) + \sqrt{k_\mathrm{B}T\zeta}\,\mathbf{w}_2(t). \tag{2.123}$$

The random variables $\mathbf{w}_i$ obey the relations

$$\langle \mathbf{w}_i \rangle = 0, \tag{2.124}$$

$$\langle \mathbf{w}_{i\alpha}\mathbf{w}_{j\beta}(t') \rangle = 2\delta_{ij}\delta_{\alpha\beta}\delta(t - t'). \tag{2.125}$$

We now transform eqs. (2.122) and (2.123) to center of mass $\mathbf{R} = (\mathbf{r}_1+\mathbf{r}_2)/2$ and end-to-end $\mathbf{q} = \mathbf{r}_1 - \mathbf{r}_2$ coordinates. By taking sum and difference of eqs. (2.122) and (2.123) we find

$$m\partial_t\dot{\mathbf{R}} = -\zeta\Big[\dot{\mathbf{R}} - \frac{1}{2}\Big(\mathbf{v}^{(\mathrm{s})}(\mathbf{r}_1) + \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_2)\Big)\Big] + \sqrt{\frac{k_\mathrm{B}T\zeta}{2}}\mathbf{w}^+, \tag{2.126}$$

$$m\partial_t\dot{\mathbf{q}} = -\zeta\Big[\dot{\mathbf{q}} - \Big(\mathbf{v}^{(\mathrm{s})}(\mathbf{r}_1) - \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_2)\Big)\Big] + 2\mathbf{F}(\mathbf{q}) + \sqrt{2k_\mathrm{B}T\zeta}\mathbf{w}^-. \tag{2.127}$$

Here we have defined the new random variables

$$\mathbf{w}^+ := \frac{1}{\sqrt{2}}(\mathbf{w}_1 + \mathbf{w}_2), \tag{2.128}$$

$$\mathbf{w}^- := \frac{1}{\sqrt{2}}(\mathbf{w}_1 - \mathbf{w}_2). \tag{2.129}$$

It follows directly from eqs. (2.124) and (2.125) that mean and covariance are retained:

$$\left\langle w_\alpha^+ \right\rangle = \left\langle w_\alpha^- \right\rangle = \left\langle w_\alpha^+ w_\beta^- \right\rangle = 0, \tag{2.130}$$

$$\left\langle w_\alpha^+ w_\beta^+ \right\rangle = \left\langle w_\alpha^- w_\beta^- \right\rangle = 2\delta_{\alpha\beta}\delta(t-t'). \tag{2.131}$$

Now, the following approximations are made:

(i) We approximate the velocity field around the center of mass by first order Taylor expansion

$$\mathbf{v}^{(\mathrm{s})}(\mathbf{r}_i) \approx \mathbf{v}^{(\mathrm{s})}(\mathbf{R}) + (\mathbf{r}_i - \mathbf{R}) \cdot \boldsymbol{\nabla}\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}}. \tag{2.132}$$

Thus we can write

$$\mathbf{v}^{(\mathrm{s})}(\mathbf{r}_1) + \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_2) \approx 2\mathbf{v}^{(\mathrm{s})}(\mathbf{R}) \tag{2.133}$$

and

$$\mathbf{v}^{(\mathrm{s})}(\mathbf{r}_1) - \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_2) \approx \mathbf{q} \cdot \boldsymbol{\nabla}\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}}. \tag{2.134}$$

(ii) One typically assumes (see e.g. [52]) that the effect of noise on $\mathbf{R}$ is small on relevant time scales τ, i.e.

$$\sqrt{\frac{k_\mathrm{B}T}{2\zeta}}\int_0^\tau \mathrm{d}t\,\mathbf{w}^+ \ll \frac{1}{2}\int_0^\tau \mathrm{d}t\,\Big(\mathbf{v}^{(\mathrm{s})}(\mathbf{r}_1) + \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_2)\Big). \tag{2.135}$$

(iii) We furthermore assume overdamped dynamics by neglecting inertial terms ($m \to 0$).

Then, eqs. (2.126) and (2.127) can be rewritten as

$$0 = -\zeta\Big[\dot{\mathbf{R}} - \mathbf{v}^{(\mathrm{s})}(\mathbf{R})\Big], \tag{2.136}$$

$$0 = -\zeta\Big[\dot{\mathbf{q}} - \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}^{(\mathrm{s})}\Big] + 2\mathbf{F}(\mathbf{q}) + \sqrt{2k_{\mathrm{B}}T\zeta}\mathbf{w}^{-}, \tag{2.137}$$

or

$$\dot{\mathbf{R}} = \mathbf{v}^{(\mathrm{s})}(\mathbf{R}), \tag{2.138}$$

$$\dot{\mathbf{q}} = \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}^{(\mathrm{s})} + \frac{2}{\zeta}\mathbf{F}(\mathbf{q}) + \sqrt{\frac{2k_{\mathrm{B}}T}{\zeta}}\mathbf{w}^{-} \tag{2.139}$$

The center of mass coordinate of the dumbbell is thus locked with the solvent flow field, while the equation of motion for $\mathbf{q}$ is still stochastic. We may therefore equate $\mathbf{v}^{(\mathrm{s})} = \mathbf{v}$ with the mass-average velocity of the solution. It is known that a Langevin equation with constant noise coefficient g,

$$\partial_t\mathbf{q} = \mathbf{h}(\mathbf{q}) + g\mathbf{w}, \tag{2.140}$$

and the properties eqs. (2.130) and (2.131) corresponds to a Fokker-Planck equation

$$\partial_t\Psi(\mathbf{q},t) = -\frac{\partial}{\partial\mathbf{q}}\cdot\left[\mathbf{h}(\mathbf{q}) + g^2\frac{\partial}{\partial\mathbf{q}}\right]\Psi(\mathbf{q},t) =: \mathscr{L}_{\mathrm{FP}}\Psi. \tag{2.141}$$

for the probability density of the end-to end vector $\Psi(\mathbf{q},t)$ [54], where the Fokker-Planck operator $\mathscr{L}_{\mathrm{FP}}$ was introduced to reduce notation. Here we assume that the total probability density in the position-velocity space factorizes and that the positional part does not depend on the center of mass of the molecule but only its end-to-end vector:

$$f(\mathbf{r}_1,\mathbf{r}_2,\dot{\mathbf{r}}_1,\dot{\mathbf{r}}_2,t) = \Psi(\mathbf{q},t)\Xi(\dot{\mathbf{r}}_1,\dot{\mathbf{r}}_2,t). \tag{2.142}$$

In the present case, the Fokker-Planck equation for the probability density of the end-to-end vector looks like

$$\partial_t\Psi = -\frac{\partial}{\partial\mathbf{q}}\cdot\left[\left(\frac{2}{\zeta}\mathbf{F}(\mathbf{q}) + \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}\right)\Psi\right] + \frac{2k_{\mathrm{B}}T}{\zeta}\frac{\partial^2}{\partial\mathbf{q}^2}\Psi. \tag{2.143}$$

With a Hookean spring force $\mathbf{F}(\mathbf{q}) = -k\mathbf{q}$ this can further be simplified to

$$\partial_t\Psi = \frac{\partial}{\partial\mathbf{q}}\cdot\left[\frac{2k}{\zeta}\mathbf{q} - \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v} + \frac{2k_{\mathrm{B}}T}{\zeta}\frac{\partial}{\partial\mathbf{q}}\right]\Psi. \tag{2.144}$$

By introducing $\tau_q = \zeta/(2k)$, the corresponding Fokker-Planck operator takes the form

$$\mathscr{L}_{\mathrm{FP}}(\mathbf{q}) = \frac{\partial}{\partial\mathbf{q}}\cdot\left[\left(\frac{1}{\tau_q}\mathbf{q} - \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}\right) + \frac{k_{\mathrm{B}}T}{k\tau_q}\frac{\partial}{\partial\mathbf{q}}\right]. \tag{2.145}$$

The time derivative of the thermal average of a quantity $A(\mathbf{q})$, that does not have any explicit time dependence, can then be evaluated by

$$\partial_t \langle A \rangle = \int \mathrm{d}^d\mathbf{q} A \partial_t \Psi = \int \mathrm{d}^d\mathbf{q}\, \Psi \mathscr{L}_{\mathrm{FP}}^\dagger A = \left\langle \mathscr{L}_{\mathrm{FP}}^\dagger A \right\rangle, \tag{2.146}$$

where the adjoint Fokker-Planck operator has the form

$$\begin{aligned} \mathscr{L}_{\mathrm{FP}}^\dagger(\mathbf{q}) &= \left(-\frac{1}{\tau_q}\mathbf{q} + \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}\right)\cdot\frac{\partial}{\partial\mathbf{q}} + \frac{k_\mathrm{B}T}{k\tau_q}\frac{\partial^2}{\partial\mathbf{q}^2} \\ &= \left(-\frac{1}{\tau_q}q_\lambda + q_\mu\partial_\mu v_\lambda\right)\frac{\partial}{\partial q_\lambda} + \frac{k_\mathrm{B}T}{k\tau_q}\frac{\partial}{\partial q_\lambda}\frac{\partial}{\partial q_\lambda}. \end{aligned} \tag{2.147}$$

Here we use the Einstein sum convention for double Greek indexes. This can now be used to calculate

$$\begin{aligned} \mathscr{L}_{\mathrm{FP}}^\dagger q_\alpha q_\beta &= \left(-\frac{1}{\tau_q}q_\lambda + q_\mu\partial_\mu v_\lambda\right)(q_\alpha\delta_{\lambda\beta} + q_\beta\delta_{\lambda\alpha}) + 2\frac{k_\mathrm{B}T}{k\tau_q}\delta_{\alpha\beta} \\ &= -\frac{2}{\tau_q}q_\alpha q_\beta + q_\alpha q_\mu\partial_\mu v_\beta + q_\beta q_\mu\partial_\mu v_\alpha + 2\frac{k_\mathrm{B}T}{k\tau_q}\delta_{\alpha\beta}. \end{aligned} \tag{2.148}$$

With the conformation tensor defined by

$$C_{\alpha\beta} = \langle q_\alpha q_\beta \rangle, \tag{2.149}$$

and the velocity gradient tensor $K_{\alpha\beta} = \partial_\beta v_\alpha$ we now find the equation of motion

$$\begin{aligned} \partial_t C_{\alpha\beta} &= -\frac{2}{\tau_q}C_{\alpha\beta} + C_{\alpha\mu}K^\mathsf{T}_{\mu\beta} + K_{\alpha\mu}C_{\mu\beta} + 2\frac{k_\mathrm{B}T}{k\tau_q}\delta_{\alpha\beta} \\ &= C_{\alpha\mu}K^\mathsf{T}_{\mu\beta} + K_{\alpha\mu}C_{\mu\beta} - \frac{2}{\tau_q}\left(C_{\alpha\beta} - \frac{k_\mathrm{B}T}{k}\delta_{\alpha\beta}\right). \end{aligned} \tag{2.150}$$

The relaxation equation eq. (2.150) is derived with a Fokker-Planck equation only in the end-to-end vector. According to eq. (2.138) however, the center of mass of a dumbbell is convected by the flow field $\mathbf{v}$. For an accurate treatment of a continuum of dumbbells one needs to account for this convection. This can be achieved by replacing ∂_t with the convective derivative

$$\mathrm{D}_t = \partial_t + \mathbf{v}\cdot\boldsymbol{\nabla}. \tag{2.151}$$

It is useful to then introduce the upper convected derivative, which accounts for convection as well as stretching and rotation of tensor quantities $\mathbf{X}(\mathbf{r}, t)$ under flow [50],

$$\overset{\triangledown}{\mathbf{X}} = \mathrm{D}_t\mathbf{X} - \mathbf{K}\mathbf{X} - \mathbf{X}\mathbf{K}^\mathsf{T}, \tag{2.152}$$

also see the books by Bird [51] and Larson [55]. The relaxation equation for the field $\mathbf{C}(\mathbf{r},t)$ then has the simple form

$$\overset{\triangledown}{\mathbf{C}} = -\frac{2}{\tau_q}\left(\mathbf{C} - \frac{k_\mathrm{B}T}{k}\mathbb{1}\right), \tag{2.153}$$

which must be zero at thermal equilibrium. Hence, $\mathbf{C}_\mathrm{eq} = k_\mathrm{B}T\mathbb{1}/k$, which is a direct consequence of the equipartition theorem. Note that this differs from the mechanical equilibrium value, which is of course zero.

2.5.1 Coupling to hydrodynamics

The conformation of the dumbbells couples to the macroscopic velocity field $\mathbf{v}(\mathbf{r},t)$ via the stress that occurs in the Navier-Stokes equations:

$$\rho \mathrm{D}_t\mathbf{v} = -\boldsymbol{\nabla}\cdot(P\mathbb{1} + \boldsymbol{\sigma}). \tag{2.154}$$

Here $P = P^{(\mathrm{d})} + P^{(\mathrm{s})}$ is the sum of partial pressures of the dumbbell and solvent phases, and $\boldsymbol{\sigma}$ is the deviatoric stress tensor of the mixture. It can be split up into contributions from the dumbbells and the solvent by

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^{(\mathrm{d})} + \boldsymbol{\sigma}^{(\mathrm{s})}. \tag{2.155}$$

The deviatoric stress of the solvent, which is considered to be a Newtonian fluid, is given by the fourth-order viscosity tensor

$$\begin{aligned}\eta_{\alpha\beta\gamma\delta} &= \eta_\mathsf{s}\left(\delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma} - \frac{2}{3}\delta_{\alpha\beta}\delta_{\gamma\delta}\right) + \eta_\mathsf{b}\delta_{\alpha\beta}\delta_{\gamma\delta} \\ &= \eta_\mathsf{s}(\delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}) + \left(\eta_\mathsf{b} - \frac{2}{3}\eta_\mathsf{s}\right)\delta_{\alpha\beta}\delta_{\gamma\delta}\end{aligned} \tag{2.156}$$

contracted by the velocity gradient tensor. For incompressible flows $\operatorname{tr}\mathbf{K} = 0$ this is

$$\boldsymbol{\sigma}^{(\mathrm{s})} = \boldsymbol{\eta} : \mathbf{K} = \eta_\mathsf{s}\left(\mathbf{K} + \mathbf{K}^\mathsf{T}\right) =: 2\eta_\mathsf{s}\mathbf{D}, \tag{2.157}$$

where the deformation rate tensor

$$\mathbf{D} = \frac{1}{2}\left(\mathbf{K} + \mathbf{K}^\mathsf{T}\right) \tag{2.158}$$

is the symmetric part of $\mathbf{K}$.

Typically, the total stress of the dumbbell portion is written as a virial (also see appendix A.5) with a component resulting from intramolecular forces as well as a contribution form relative motion with respect to $\mathbf{v}$:

$$\boldsymbol{\Pi}^{(\mathrm{d})} = \boldsymbol{\sigma}^{(\mathrm{d})} + P^{(\mathrm{d})}\mathbb{1} = \frac{\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}}\left[\langle\mathbf{q}\mathbf{F}(\mathbf{q})\rangle + m\sum_{i=1}^{2}\langle(\dot{\mathbf{r}}_i - \mathbf{v})(\dot{\mathbf{r}}_i - \mathbf{v})\rangle\right], \tag{2.159}$$

where $m^{(\mathrm{d})} = 2m$ is the mass of one dumbbell and $\rho^{(\mathrm{d})}(\mathbf{r}, t)$ is the dumbbell mass density. This expression can be derived by considering arbitrary planes in the system and adding up effects of dumbbells with beads on both sides of these planes, as well as the effect of momentum transport, see [51]. If one accepts eq. (2.159) as a premise and assumes a Maxwell-Boltzmann distribution of velocities, the average of the kinetic term can be evaluated and, with the ideal gas equation of state $P^{(\mathrm{d})} = \rho^{(\mathrm{d})} k_{\mathrm{B}} T / m^{(\mathrm{d})}$, the deviatoric part of the dumbbell stress be written as

$$\boldsymbol{\sigma}^{(\mathrm{d})} = \frac{\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}}(\langle \mathbf{qF}(\mathbf{q}) \rangle + 2k_{\mathrm{B}}T\mathbb{1}) - P^{(\mathrm{d})}\mathbb{1} = -\frac{k\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}}\left(\mathbf{C} - \frac{k_{\mathrm{B}}T}{k}\mathbb{1}\right). \tag{2.160}$$

Here, the term $k_{\mathrm{B}}T\rho^{(\mathrm{d})}(\mathbf{r}, t)/m^{(\mathrm{d})}$ can be interpreted as density dependent elastic modulus. The constitutive equation eq. (2.160) is called the Kramers form of the stress tensor.

2.5.2 Ensemble Problems

The above approach is standard in kinetic theory but seems problematic for the following reason: In hydrodynamics, one commonly works under the assumption of local equilibrium. This means that fast variables can be averaged over whereas slow variables occurring in the macroscopic description, like the mass, energy, momentum, and in this case also the conformation tensor, define the local fluid element's thermodynamic ensemble. Therefore, they should enter as constraints in any statistical average taken. In particular, the averages in eq. (2.159) should not simply be taken over all chain conformations. Rather they should be constrained to the local conformation tensor, making the process of averaging the configurations superfluous in the first place.

We shall try to further illuminate this point by considering a general polymer system. The following deliberations are published in reference [56]. Let the system be decomposed into subcells inside which a description of the internal degrees of freedom of the polymer chains is given by a set of variables ξ_i, $i = 1, \ldots, n$. As above, the center of mass contribution can be accounted for by adding convection 'by hand' and shall not be treated explicitly. In the absence of external driving forces, let the dynamics of the probability distribution $\Psi(\{\xi_i\})$ be described by the Fokker-Planck equation

$$\partial_t \Psi = \mathscr{L}_{\mathrm{FP}} \Psi. \tag{2.161}$$

Apart from the standard hydrodynamic variables like mass and momentum density, we introduce additional observables $A_i(\{\xi_k\})$, $j = 1, \ldots, m$ to describe the conformational state of the polymer molecules. These may be transformed into a macroscopic interpretation by taking the thermal averages

$$A_i^{(\mathrm{mac})}(\mathbf{r}) := \langle A_i \rangle = \int \mathrm{d}^n \xi \, A_i \Psi, \tag{2.162}$$

where the integral is carried out over the local subcell and $\mathbf{r}$ is the position of that subcell. Note that if on the microscopic scale a property is evaluated at the molecule's center of mass the evaluation at the macroscale must be done at the subcell position $\mathbf{r}$. The time derivative of the macroscopic quantities is then given by

$$\partial_t A_i^{(\mathrm{mac})} = \left\langle \mathscr{L}_{\mathrm{FP}}^{\dagger} A_i \right\rangle, \tag{2.163}$$

also see eq. (2.146). The center of mass motion can again be included by switching to the convective derivative $\partial_t \to \mathrm{D}_t = \partial_t + \mathbf{v} \cdot \boldsymbol{\nabla}$.

The backcoupling to macroscopic hydrodynamics happens via an additional viscoelastic stress in the Navier-Stokes equations. On the microscopic scale, the stress is a function of the microscopic variables, $\tilde{\Pi}_{\alpha\beta}(\{\xi_i\})$, while on the macroscopic scale it is expressed in terms of the $A_j^{(\mathrm{mac})}$, $\bar{\Pi}_{\alpha\beta}\Big(\Big\{A_j^{(\mathrm{mac})}\Big\}\Big)$. It is now crucial to correctly perform the continuum limit $\tilde{\Pi}_{\alpha\beta} \to \bar{\Pi}_{\alpha\beta}$. To do this, Bird et al. pursue the following strategy: First, they evaluate the full thermal average of the microscopic stress,

$$\left\langle \tilde{\Pi}_{\alpha\beta} \right\rangle = \int \mathrm{d}^n \xi \Psi \tilde{\Pi}_{\alpha\beta}. \tag{2.164}$$

This average–only by coincidence–happens to be a function of the $A_i^{(\mathrm{mac})}$,

$$\left\langle \tilde{\Pi}_{\alpha\beta} \right\rangle = \Sigma_{\alpha\beta}(\{\langle A_i \rangle\}). \tag{2.165}$$

The transfer to the macroscale is then performed by just equating the thus found function to the macroscopic form of the stress:

$$\bar{\Pi}_{\alpha\beta}\Big(\Big\{A_i^{(\mathrm{mac})}\Big\}\Big) = \Sigma_{\alpha\beta}\Big(\Big\{A_i^{(\mathrm{mac})}\Big\}\Big). \tag{2.166}$$

This however neglects that, as slow variables occurring on the macroscale, the $A_i^{(\mathrm{mac})}$ define the local thermodynamic ensemble of the subcell. This must be reflected by a constraint in the averaging, which implies only averaging over the fast degrees of freedom. We denote this constrained average by:

$$\langle X \rangle_{\mathrm{fast}} := \frac{\int \mathrm{d}^n \xi \Psi \Big[\prod_{i=1}^m \delta\Big(A_i(\{\xi_j\}) - A_i^{(\mathrm{mac})}(t)\Big)\Big] X(\{\xi_i\})}{\int \mathrm{d}^n \xi \Psi \prod_{i=1}^m \delta\Big(A_i(\{\xi_j\}) - A_i^{(\mathrm{mac})}(t)\Big)}, \tag{2.167}$$

which then immediately implies the relation

$$\langle A_i \rangle_{\mathrm{fast}}(t) = A_i^{(\mathrm{mac})}(t) = \langle A_i \rangle(t). \tag{2.168}$$

This relation however does not hold true in general. In particular, the macroscopic stress must be obtained by evaluating the right-hand side of

$$\bar{\Pi}\Big(\Big\{A_i^{(\mathrm{mac})}\Big\}\Big) = \Big\langle \tilde{\Pi}_{\alpha\beta}(\{\xi_i\}) \Big\rangle_{\mathrm{fast}}. \tag{2.169}$$

By construction, this form of the macroscopic stress is then guaranteed to be a function of the $A_i^{(\mathrm{mac})}$.

In case of the present dumbbell model, the microscopic stress (omitting the kinetic contribution $\propto \delta_{\alpha\beta}$) is given by

$$\tilde{\Pi}_{\alpha\beta}(\mathbf{q}) = -k\frac{\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}} q_\alpha q_\beta = -k\frac{\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}} C_{\alpha\beta}. \tag{2.170}$$

Because $C_{\alpha\beta} = q_\alpha q_\beta$ really has only three independent components, it seems natural to choose $\{A_i\} = \{q_\alpha\}$ rather than $\{A_i\} = \{C_{\alpha\beta}\}$ in order not to artificially introduce any additional degrees of freedom. In this case $m = n = d$ and eq. (2.167) reduces to the simple form

$$\begin{aligned} \langle X \rangle_{\mathrm{fast}} &= \frac{\int \mathrm{d}^d\mathbf{q}\Psi(\mathbf{q})\delta\Big(\mathbf{q} - \mathbf{q}^{(\mathrm{mac})}\Big)X(\mathbf{q})}{\int \mathrm{d}^d\mathbf{q}\Psi(\mathbf{q})\delta(\mathbf{q} - \mathbf{q}^{(\mathrm{mac})})} \\ &= \frac{\Psi\Big(\mathbf{q}^{(\mathrm{mac})}\Big)X\Big(\mathbf{q}^{(\mathrm{mac})}\Big)}{\Psi\big(\mathbf{q}^{(\mathrm{mac})}\big)} = X\Big(\mathbf{q}^{(\mathrm{mac})}\Big). \end{aligned} \tag{2.171}$$

In particular, this implies the macroscopic form of the stress

$$\bar{\Pi}_{\alpha\beta} = -k\frac{\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}} q_\alpha^{(\mathrm{mac})} q_\beta^{(\mathrm{mac})} = -k\frac{\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}} \langle q_\alpha \rangle \langle q_\beta \rangle \tag{2.172}$$

instead of

$$\bar{\Pi}_{\alpha\beta} = k\frac{\rho^{(\mathrm{d})}}{m^{(\mathrm{d})}} \langle q_\alpha q_\beta \rangle , \tag{2.173}$$

as it is commonly proposed in the rheological literature. Therefore, instead of eq. (2.150), we must consider

$$\partial_t \langle \mathbf{q} \rangle = \left\langle \mathscr{L}_{\mathrm{FP}}^\dagger \mathbf{q} \right\rangle = -\frac{1}{\tau_q} \langle \mathbf{q} \rangle + \langle \mathbf{q} \rangle \cdot \boldsymbol{\nabla}\mathbf{v} \tag{2.174}$$

and the related equation

$$\partial_t(\langle q_\alpha \rangle \langle q_\beta \rangle) = \langle q_\alpha \rangle \langle q_\gamma \rangle \partial_\gamma v_\beta + \langle q_\beta \rangle \langle q_\gamma \rangle \partial_\gamma v_\alpha - \frac{2}{\tau_q} \langle q_\alpha \rangle \langle q_\beta \rangle , \tag{2.175}$$

which is symmetric with respect to a flip of dumbbell orientation. The conclusion must be, that $\langle \mathbf{q} \rangle \langle \mathbf{q} \rangle$ should be the dynamical field variable entering the stress and not $\langle \mathbf{qq} \rangle$ as prescribed in the standard Oldroyd-B model. It therefore seems more consistent to use a description without noise first, resulting in a relaxation of the conformation tensor toward its mechanical equilibrium value (zero) rather than towards the thermal equilibrium value given by equipartition. The equations of motion can then be thermalized with a Langevin noise at the very end. This is then in line with the novel model derived in section 4.1.

2.6 Onsager principle

In this section the Onsager reciprocal relations [57, 58] are discussed. This symmetry of transport coefficients is a basic requirement in nonequilibrium thermodynamics and therefore an essential ingredient for the theoretical derivations made in section 4.1.

For the purpose of illustration consider the fundamental thermodynamic relation

$$\mathrm{d}U = T\mathrm{d}S - P\mathrm{d}V + \mu\,\mathrm{d}M \tag{2.176}$$

with internal energy U, temperature T, entropy S, pressure P, chemical potential μ and mass M. The differential quantities are extensive in nature while the non-differential quantities are intensive. At constant volume $\mathrm{d}V = 0$, transforming to the volume normalized versions of the intensive variables, u for U, s for S and the mass density ρ for M and solving for the entropy density yields the new relation

$$\mathrm{d}s = \frac{1}{T}\mathrm{d}u - \frac{\mu}{T}\mathrm{d}\rho. \tag{2.177}$$

This defines the *entropic conjugate* variable pairs $1/T$ and u as well as $-\mu/T$ and ρ.

In his famous paper [57], Lars Onsager considers currents of a quantity X, $\mathbf{j}_X$ that result from the gradients in the respective entropic conjugate variable Y,

$$\mathbf{j}_X = -M_{XY}\boldsymbol{\nabla}Y. \tag{2.178}$$

Examples for such currents include heat flow $\mathbf{j}_u$ that results from a temperature gradient $\boldsymbol{\nabla}(1/T)$ or a Fickian diffusion current $\mathbf{j}_\rho$ resulting from the gradient $\boldsymbol{\nabla}(-\mu/T)$.

The Onsager reciprocal relations state that if a current results from multiple gradients, i.e.

$$\mathbf{j}_X = -\sum_Y M_{XY}\boldsymbol{\nabla}Y, \tag{2.179}$$

the matrix of transport coefficients M_{XY} must be symmetric:

$$M_{XY} = M_{YX}. \tag{2.180}$$

In other words, if a diffusion current induces a heat flow with some transport coefficient, a heat flow must likewise be accompanied by a diffusion current with the same transport coefficient. This is a crucial symmetry that dissipative nonequilibrium processes need to satisfy. It is automatically encoded in the GENERIC formalism (section 2.8), which also allows for conservative contributions to the dynamics.

2.7 Hamiltonian field theory with Poisson brackets

In this section, the field-theoretic Poisson bracket formulation of Hamiltonian mechanics is described. It is central for the construction of the conservative part of dynamics of the viscoelastic phase separation model derived in section 4.1. If the fields describing the system have a microscopic, particle-based interpretation, the Poisson brackets can be calculated on the microscopic level and used to formulate the dynamic equations in the continuum.

We consider a system that, on the microscopic level, is described by the generalized coordinates $\mathbf{x}_i$ and their respective conjugate momenta $\mathbf{p}_i$. Furthermore we assume that there is a set of fields $\phi_i(\mathbf{x}, \{\mathbf{x}_k(t)\}, \{\mathbf{p}_k(t)\})$ which give a complete representation of the system and can be expressed in terms of the microscopic coordinates and momenta. As an example for such a field, consider the mass density of a system of particles at positions $\mathbf{r}_i$ and with masses m_i defined by

$$\rho(\mathbf{r}, \{\mathbf{r}_k\}, t) = \sum_i m_i \delta(\mathbf{r} - \mathbf{r}_i), \tag{2.181}$$

where δ is the Dirac delta function. This has a well-defined microscopic interpretation and a continuum limit can be performed by 'smearing out' the delta peaks. Similar expressions can be defined for the momentum density as well, see section 4.1 eq. (4.11).

The Hamiltonian is then written as a functional of the general fields $\mathscr{H}(\{\phi_k(\mathbf{x}, t)\})$. We can now use Hamilton's equations of motion on the microscopic level,

$$\frac{\partial \mathbf{x}_i}{\partial t} = \frac{\partial \mathscr{H}}{\partial \mathbf{p}_i}, \qquad \frac{\partial \mathbf{p}_i}{\partial t} = -\frac{\partial \mathscr{H}}{\partial \mathbf{x}_i}, \tag{2.182}$$

to determine the time derivative of any field $\psi(\mathbf{x}, \{\mathbf{x}_k(t)\}, \{\mathbf{p}_k(t)\})$. If ψ has no explicit time dependence, $\partial\psi/\partial t = 0$, the total time derivative is

$$\begin{aligned} \frac{\mathrm{d}\psi}{\mathrm{d}t} &= \sum_i \left(\frac{\partial \psi}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial t} + \frac{\partial \psi}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial t} \right) = \sum_i \left(\frac{\partial \psi}{\partial \mathbf{x}_i} \frac{\partial \mathscr{H}}{\partial \mathbf{p}_i} - \frac{\partial \psi}{\partial \mathbf{p}_i} \frac{\partial \mathscr{H}}{\partial \mathbf{x}_i} \right) \\ &= \sum_{ij} \int \mathrm{d}^d \mathbf{x}' \left(\frac{\partial \psi(\mathbf{x})}{\partial \mathbf{x}_i} \frac{\partial \phi_j(\mathbf{x}')}{\partial \mathbf{p}_i} - \frac{\partial \psi(\mathbf{x})}{\partial \mathbf{p}_i} \frac{\partial \phi_j(\mathbf{x}')}{\partial \mathbf{x}_i} \right) \frac{\delta \mathscr{H}}{\delta \phi_j(\mathbf{x}')}, \end{aligned} \tag{2.183}$$

where the chain rule for functionals eq. (A.11) was used in the last step. With the definition of the Poisson bracket

$$\{\psi, \phi\} = \sum_i \left(\frac{\partial \psi}{\partial x_i} \frac{\partial \phi}{\partial p_i} - \frac{\partial \psi}{\partial p_i} \frac{\partial \phi}{\partial x_i} \right), \tag{2.184}$$

this becomes

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = \sum_j \int \mathrm{d}^d\mathbf{x}' \, \{\psi(\mathbf{x}), \phi_j(\mathbf{x}')\} \frac{\delta \mathscr{H}}{\delta \phi_j(\mathbf{x}')}. \tag{2.185}$$

In particular, eq. (2.185) also holds if ψ is one of the ϕ_i.

This formalism can be used to coarse-grain from a particle-based description to a field-theoretic representation of a system. For the hydrodynamics of a simple fluid, for example, the $\mathbf{x}_i$ and $\mathbf{p}_i$ be taken as the coordinates and momenta of the particles that constitute the fluid, and the fields ϕ_i be chosen to represent mass density and momentum density $\rho(\mathbf{x}, t)$ and $\mathbf{j}(\mathbf{x}, t)$. With a suitable microscopic representation of ρ and $\mathbf{j}$, the Poisson brackets may be evaluated on the microscopic level, while the dynamic equations of the continuum can be calculated from eq. (2.185), resulting in the Euler equations, c.f. section 4.1.

Generally, the choice of the ϕ_i must not be unique, and choosing can pose a nontrivial task. For example, the evaluation of the brackets $\{\phi_i, \phi_j\}$ could result in new fields $\tilde{\phi}_i$, which can not be expressed in terms of the original fields. A different set of fields must be chosen, or the new fields be incorporated into the original set, and the additional brackets be evaluated. This procedure could, in the worst case, continue infinitely. Hence, care must be taken in the choice of ϕ_i such that the final set of fields is closed. The existence of such a closed set is by no means guaranteed but depends on the nature of the system and its description. For the very simple dumbbell model in section 4.1, finding such a set of fields turns out to be possible. Typically, it is advisable to make sure that none of the variables that occur in the particle description are discarded.

Finally we take note of some mathematical properties of the Poisson brackets that can be proven in a straightforward way and will be useful for calculations later on:

$$\{x_i, x_j\} = \{p_i, p_j\} = 0, \tag{2.186}$$
$$\{x_i, p_j\} = \delta_{ij}, \tag{2.187}$$
$$\{f, g\} = -\{g, f\}, \tag{2.188}$$
$$\{\lambda(f+g), h\} = \lambda\{f, h\} + \lambda\{g, h\}, \tag{2.189}$$
$$\{f, \lambda(g+h)\} = \lambda\{f, g\} + \lambda\{f, h\}, \tag{2.190}$$
$$\{fg, h\} = \{f, h\}g + f\{g, h\}, \tag{2.191}$$
$$\{f, g(h)\} = \{f, h\}\frac{\partial g}{\partial h}. \tag{2.192}$$

2.8 GENERIC formalism

The acronym GENERIC [27, 28] stands for 'general equation for the nonequilibrium reversible-irreversible coupling'. It is a formalism that can be used to derive the dynamic equations of abstract state variables while taking into account fundamental principles of nonequilibrium thermodynamics such as energy dissipativity $\mathrm{d}\mathscr{H}/\mathrm{d}t \leq 0$, the Onsager reciprocal relations (c.f. section 2.6), as well as the Poisson bracket structure of the conservative part of the dynamics (c.f. section 2.7). It gives a general framework for the derivation of consistent dynamic equations describing nonequilibrium processes and proved to be particularly useful in the determination of the dissipative part of dynamics of the phase separation model derived in section 4.1.

Fundamental ingredients for the GENERIC formalism in the microcanonic ensemble are total energy E, entropy S and two linear operators $\mathbf{L}$ and $\mathbf{M}$. The time evolution equations are written as

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{L}\frac{\partial E}{\partial \mathbf{x}} + \mathbf{M}\frac{\partial S}{\partial \mathbf{x}}, \tag{2.193}$$

where $\mathbf{x}$ is a vector of state variables that fully describe the system. The operator $\mathbf{L} = -\mathbf{L}^\dagger$ is skew-adjoined and describes the conservative (reversible) part of the dynamics. Therefore it must not couple to the entropy, which is expressed by the relation

$$\mathbf{L}\frac{\partial S}{\partial x} = 0. \tag{2.194}$$

The operator $\mathbf{M} = \mathbf{M}^\dagger$ is self-adjoint and positive semi-definite i.e. $\mathbf{x}^\mathsf{T}\mathbf{M}\mathbf{x} \geq 0$. It represents the dissipative (irreversible) part of the dynamics, hence

$$\mathbf{M}\frac{\partial E}{\partial x} = 0. \tag{2.195}$$

Equation (2.193) is easily generalized to a system that is entirely described by *fields* $\phi_i(\mathbf{r}, t)$:

$$\frac{\mathrm{d}\phi_i(\mathbf{r}, t)}{\mathrm{d}t} = \sum_j \int \mathrm{d}^d\mathbf{r}' \left[L_{ij}(\mathbf{r}, \mathbf{r}')\frac{\delta E}{\delta \phi_j(\mathbf{r}')} + M_{ij}(\mathbf{r}, \mathbf{r}')\frac{\delta S}{\delta \phi_j(\mathbf{r}')} \right]. \tag{2.196}$$

If we disregard dissipative effects by setting $\mathbf{M} = 0$, and compare eq. (2.196) with eq. (2.185) we discover that $\mathbf{L}$ must be the matrix of Poisson brackets,

$$L_{ij}(\mathbf{r}, \mathbf{r}') = \{\phi_i(\mathbf{r}), \phi_j(\mathbf{r}')\}. \tag{2.197}$$

With the Helmholtz free energy

$$F(N, V, T) = E - TS \tag{2.198}$$

and eqs. (2.194) and (2.195), we find that the evolution equations for the isothermal ensemble have the form

$$\frac{\mathrm{d}\phi_i(\mathbf{r},t)}{\mathrm{d}t} = \sum_j \int \mathrm{d}^d\mathbf{r}' \left(L_{ij}(\mathbf{r},\mathbf{r}') - \frac{1}{T} M_{ij}(\mathbf{r},\mathbf{r}') \right) \frac{\delta F}{\delta \phi_j(\mathbf{r}')}. \tag{2.199}$$

From here it is obvious that the dynamic equation can be separated into conservative and dissipative parts

$$\left(\frac{\mathrm{d}\phi_i(\mathbf{r},t)}{\mathrm{d}t} \right)_{\mathrm{cons}} = \sum_j \int \mathrm{d}^d\mathbf{r}'\, L_{ij}(\mathbf{r},\mathbf{r}') \frac{\delta F}{\delta \phi_j(\mathbf{r}')}, \tag{2.200}$$

$$\left(\frac{\mathrm{d}\phi_i(\mathbf{r},t)}{\mathrm{d}t} \right)_{\mathrm{diss}} = -\frac{1}{T} \sum_j \int \mathrm{d}^d\mathbf{r}'\, M_{ij}(\mathbf{r},\mathbf{r}') \frac{\delta F}{\delta \phi_j(\mathbf{r}')}. \tag{2.201}$$

In the following we will absorb T by a redefinition of the dissipative operator $\mathbf{M}/T \to \mathbf{M}$.

Let us now examine the time derivative of the free energy. With the symmetry relation of $\mathbf{L}$ and the positive definiteness of $\mathbf{M}$, we can determine

$$\begin{aligned} \frac{\mathrm{d}F}{\mathrm{d}t} &= \sum_i \int \mathrm{d}^d\mathbf{r}\, \frac{\delta F}{\delta \phi_i(\mathbf{r})} \frac{\mathrm{d}\phi_i}{\mathrm{d}t} \\ &= \sum_{ij} \int \mathrm{d}^d\mathbf{r}\, \mathrm{d}^d\mathbf{r}'\, \frac{\delta F}{\delta \phi_i(\mathbf{r})} (L_{ij}(\mathbf{r},\mathbf{r}') - M_{ij}(\mathbf{r},\mathbf{r}')) \frac{\delta F}{\delta \phi_j(\mathbf{r}')} \\ &= -\sum_{ij} \int \mathrm{d}^d\mathbf{r}\, \mathrm{d}^d\mathbf{r}'\, \frac{\delta F}{\delta \phi_i(\mathbf{r})} M_{ij}(\mathbf{r},\mathbf{r}') \frac{\delta F}{\delta \phi_j(\mathbf{r}')} \leq 0, \end{aligned} \tag{2.202}$$

proving that the free energy of an isothermal system is monotonically decreasing under GENERIC dynamics.

We note that if one is able to write the energy dissipation rate of a system in the form of eq. (2.202), this at the same time determines the dissipative matrix $\mathbf{M}$. If, in addition to that, the set of all Poisson brackets can be evaluated and is closed, $\mathbf{L}$ is known from eq. (2.197) and the complete dynamics can be written down with eq. (2.199).

Chapter 3

Numerical methods and implementation details

When simulating viscoelastic phase separation, exact knowledge of the conformations of the polymer molecules at all points in time is of high value. While hydrodynamic interactions with the solvent are important (c.f. section 2.4.3 about Model H), we do not believe that the molecular structure of the solvent is critical. Thus, to be as efficient as possible, we take a multiscale approach to the numerical simulation. For the polymers, we use Molecular Dynamics (MD, see section 3.1) to integrate the equations of motion of bead-spring chain molecules. The bonded interaction is modeled with the FENE potential, which limits the maximum possible extension of the chains. Interaction with the solvent is taken into account implicitly by a non-bonded potential with tunable interaction strength.

The MD particles are coupled to a Lattice Boltzmann (LB, see section 3.2) solvent background via dissipative forces. The fluid is interpreted as mass densities on a discrete lattice. Lattice sites can only exchange mass with a defined set of neighbors. An integration step can be divided into a streaming of mass and a collision process that facilitates relaxation to local equilibrium. Asymptotically, the LB fluid turns into a Navier-Stokes continuum. The dissipative and random forces resulting from friction with the MD particle correspond to a force density, which can be incorporated by an extension of the collision process. In doing so, momentum conservation must not be violated.

This coupled LB/MD scheme was used to produce the numerical results presented in chapter 5. At the same time, it provides the microscopic picture from which the continuum viscoelastic phase separation model in section 4.1 is derived. This strategy will allow for good comparability between simulations of the mesoscopic model presented here and future simulations of the new continuum model.

3.1 Molecular Dynamics simulations

Classical Molecular Dynamics (MD) is a simulation method for simulating a set of particles that have continuous coordinates $\mathbf{r}_i$ inside a simulation box. The particles interact with each other, and possibly with the external surrounding, via potentials $U_j(\{\mathbf{r}_i\})$. Here MD methods are used to simulate the polymer molecules coupled to the Lattice Boltzmann solvent. The molecules are modeled by chains of beads, which are connected by FENE springs with a finite maximum extension. The non-bonded interaction is given by the Lennard-Jones Cosine potential, which has the advantage that the strength of attraction can be adjusted. This corresponds to an implicit adjustment of solvent quality and is used to introduce phase separation. This combination of potentials is a modification of the Kremer-Grest model [59] which is widely used for the simulation of polymer melts. It was introduced by Soddemann et al. [60] and used by Steinhauser [61] to study polymer solutions at different solvent qualities.

The MD simulation works iteratively by solving Newton's equations of motion where the forces are given by the gradients of the potentials

$$\frac{\mathbf{p}_i}{m_i} = \dot{\mathbf{r}}_i = \mathbf{v}_i, \qquad \dot{\mathbf{p}}_i = -\sum_j \frac{\partial U_j}{\partial \mathbf{r}_i} = F_i, \tag{3.1}$$

where $\mathbf{p}_i$ are the particle momenta conjugate to the positions $\mathbf{r}_i$ and m_i the particle masses. The text following is inspired by references [62, 63] where more details can be found.

3.1.1 Velocity Verlet algorithm

In this work, we use the velocity Verlet algorithm [64, 65] to integrate Newton's equations of motion eq. (3.1). An integration of coordinates and velocities corresponding to one MD timestep δt_{MD} in the velocity Verlet algorithm, $\mathbf{r}_i(t) \to \mathbf{r}(t + \delta t_{\mathrm{MD}})$, $\mathbf{v}_i(t) \to \mathbf{v}_i(t + \delta t_{\mathrm{MD}})$, can be divided into three steps,

$$\begin{aligned}
\mathbf{v}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) &= \mathbf{v}_i(t) + \frac{1}{2m_i}\delta t_{\mathrm{MD}}\mathbf{F}_i(t), \\
\mathbf{r}_i(t + \delta t_{\mathrm{MD}}) &= \mathbf{r}(t) + \delta t_{\mathrm{MD}}\mathbf{v}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right), \\
\mathbf{v}(t + \delta t_{\mathrm{MD}}) &= \mathbf{v}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) + \frac{1}{2m_i}\delta t_{\mathrm{MD}}\mathbf{F}_i(t + \delta t_{\mathrm{MD}}),
\end{aligned} \tag{3.2}$$

which are often termed 'kick-drift-kick'. This split can be motivated as follows: Consider an observable $A(\{\mathbf{r}_k(t)\}, \{\mathbf{p}_k(t)\}, t)$ on phase space. The total time derivative of A can

then be written as

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \frac{\partial A}{\partial t} + \sum_i \left(\mathbf{v}_i \cdot \frac{\partial}{\partial \mathbf{r}_i} + \mathbf{F}_i \cdot \frac{\partial}{\partial \mathbf{p}_i} \right) A = \left(\frac{\partial}{\partial t} + \mathrm{i}L \right) A \tag{3.3}$$

where we have introduced the Liouville operator

$$\mathrm{i}L = \sum_i \left(\mathbf{v}_i \cdot \frac{\partial}{\partial \mathbf{r}_i} + \mathbf{F}_i \cdot \frac{\partial}{\partial \mathbf{p}_i} \right). \tag{3.4}$$

Hence, as long as A has no explicit time dependence, $\partial A/\partial t = 0$, we can express the time derivative by $\mathrm{d}A/\mathrm{d}t = \mathrm{i}LA$, a differential equation which has the formal solution

$$A(\{\mathbf{r}_k(t)\}, \{\mathbf{p}_k(t)\}) = \exp(\mathrm{i}Lt) A(\{\mathbf{r}_k(0)\}, \{\mathbf{p}_k(0)\}). \tag{3.5}$$

We can split the Liouville operator into partial operators acting on coordinates and momenta respectively,

$$L = L_{\mathbf{r}} + L_{\mathbf{p}}, \tag{3.6}$$

$$\mathrm{i}L_{\mathbf{r}} = \sum_i \mathbf{v}_i \cdot \frac{\partial}{\partial \mathbf{r}_i}, \tag{3.7}$$

$$\mathrm{i}L_{\mathbf{p}} = \sum_i \mathbf{F}_i \cdot \frac{\partial}{\partial \mathbf{p}_i}. \tag{3.8}$$

However, $L_{\mathbf{r}}$ and $L_{\mathbf{p}}$ do not commute and the exponential in eq. (3.5) does not factorize. Rather, the Trotter formula must be applied [66],

$$\exp(\mathrm{i}Lt) = \lim_{n\to\infty} \left[\exp\left(\mathrm{i}L_{\mathbf{p}} \frac{t}{2n} \right) \exp\left(\mathrm{i}L_{\mathbf{r}} \frac{t}{n} \right) \exp\left(\mathrm{i}L_{\mathbf{p}} \frac{t}{2n} \right) \right]^n \tag{3.9}$$

which for $t = \delta t_{\mathrm{MD}}$ and $n = 1$ becomes

$$\exp(\mathrm{i}L\delta t_{\mathrm{MD}}) \approx \exp\left(\mathrm{i}L_{\mathbf{p}} \frac{\delta t_{\mathrm{MD}}}{2} \right) \exp(\mathrm{i}L_{\mathbf{r}} \delta t_{\mathrm{MD}}) \exp\left(\mathrm{i}L_{\mathbf{p}} \frac{\delta t_{\mathrm{MD}}}{2} \right). \tag{3.10}$$

Equation (3.10) has error terms of $\mathcal{O}({\delta t_{\mathrm{MD}}}^4)$ and therefore becomes exact in the limit of an infinitely small time step. By taking the particle positions and velocities for the observable A, one sees that this factorization is equivalent to the Velocity-Verlet algorithm.

The velocity Verlet algorithm is time-reversible, momentum conserving, and in particular conserves the phase space volume making it a symplectic integrator.

3.1.2 Force calculation

In the first and last step of eq. (3.2), the forces $\mathbf{F}_i$ need to be calculated. Since non-bonded pair interactions depend on all possible pairs of the N particles in the system, evaluating all the forces corresponds to $N(N-1)/2 = \mathcal{O}(N^2)$ operations. By introducing some skin thickness r_s as well as a cutoff distance r_c, beyond which all non-bonded interactions can be neglected, this computational effort can be reduced significantly. This is typically done by generating Verlet lists, which contain, for a given particle i the indices of all other particles $j > i$ within a distance of $r_\mathrm{c} + r_\mathrm{s}$, $V_i = \{j > i : |\mathbf{r}_i - \mathbf{r}_j| < r_\mathrm{c} + r_\mathrm{s}\}$. Using the Verlet lists only takes $\mathcal{O}(N)$ operations. The individual lists stay valid as long as no particle travels a distance larger than $r_\mathrm{s}/2$ and have to be regenerated otherwise, again requiring $\mathcal{O}(N^2)$ operations.

Efficiency can be further improved by dividing the system into subcells that are larger than $r_\mathrm{c} + r_\mathrm{s}$ via a so-called cell list. Then, possible candidates for the Verlet list in three dimensions are limited to the same cell plus the 26 neighboring cells. With the described strategy, one can achieve a scaling which is linear in N [67].

3.1.3 Langevin thermostat

Solving Newton's equations of motion for a fixed set of particles in a box with periodic boundary conditions corresponds to the microcanonical or (N, V, E) ensemble in which particle number, system volume, and total energy are constant and the temperature T is fluctuating around some mean value. Here we are however interested in the canonical, or (N, V, T) ensemble, in which the total energy fluctuates around its average value and the temperature is held constant by a thermostat. The particular thermostat used in the present work is the Langevin thermostat, based on the interplay between thermal motion and friction as required by the fluctuation-dissipation theorem. In this case, the equations of motion are

$$\mathbf{v}_i = \dot{\mathbf{r}}_i, \qquad m_i \dot{\mathbf{v}}_i = -\sum_j \frac{\partial U_j}{\partial \mathbf{r}_i} - \zeta_\mathrm{L} \mathbf{v}_i + \sigma \mathbf{w}_i. \tag{3.11}$$

In the absence of any interaction, $U_j = 0 \forall j$, the particles perform Brownian motion, and the Langevin friction constant ζ_L is related to the diffusion constant D by the Einstein relation

$$\zeta_\mathrm{L} = \frac{k_\mathrm{B} T}{D}. \tag{3.12}$$

The terms $\mathbf{w}_i(t)$ are uncorrelated random variables with zero mean and no memory,

$$\langle \mathbf{w}_i(t) \rangle = 0, \tag{3.13}$$

$$\langle w_{i\alpha}(t) w_{i\beta}(t') \rangle = \delta_{ij} \delta_{\alpha\beta} \delta(t - t'), \tag{3.14}$$

and the strength is given by

$$\sigma = \sqrt{2\zeta k_{\mathrm{B}} T}. \tag{3.15}$$

A robust extension that incorporates the Langevin thermostat into the velocity Verlet algorithm is given by the update scheme

$$\mathbf{v}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) = \mathbf{v}_i(t) + \frac{\delta t_{\mathrm{MD}}}{2m_i}\mathbf{F}_i(t), \tag{3.16}$$

$$\mathbf{r}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) = \mathbf{r}_i(t) + \frac{\delta t_{\mathrm{MD}}}{2}\mathbf{v}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right), \tag{3.17}$$

$$\begin{aligned}\mathbf{v}'_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) &= \exp\left(-\zeta\frac{\delta t_{\mathrm{MD}}}{m_i}\right)\mathbf{v}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) \\ &\quad + \sqrt{\frac{k_{\mathrm{B}}T}{m_i}(1 - \exp[-2\zeta\delta t_{\mathrm{MD}}/m_i])}\mathbf{w}_i,\end{aligned} \tag{3.18}$$

$$\mathbf{r}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) = \mathbf{r}_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) + \frac{\delta t_{\mathrm{MD}}}{2}\mathbf{v}'_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right), \tag{3.19}$$

$$\mathbf{v}_i(t + \delta t_{\mathrm{MD}}) = \mathbf{v}'_i\left(t + \frac{\delta t_{\mathrm{MD}}}{2}\right) + \frac{\delta t_{\mathrm{MD}}}{2m_i}\mathbf{F}_i(t + \delta t_{\mathrm{MD}}), \tag{3.20}$$

where the $\mathbf{w}_i$ are typically chosen as independent Gaussian random variables with

$$\langle w_{i\alpha}\rangle = 0, \qquad \langle w_{i\alpha}w_{j\beta}\rangle = \delta_{ij}\delta_{\alpha\beta}. \tag{3.21}$$

It has been shown, however, that the random variables do not necessarily have to be Gaussian, but only need to satisfy certain moment conditions, see reference [68]. The above stepping scheme can be derived via similar operator splitting techniques and is not unique. However, in this form it has some particularly nice properties with respect to numerical stability, and accuracy [69, 70]. It is quasi-symplectic in the sense that it becomes symplectic for $\zeta = 0$ [71].

3.1.4 FENE potential

The bonds between the individual beads of a chain molecule are modeled with the FENE pair potential [51, 59]:

$$U_{\mathrm{FENE}}(r) = -\frac{K_{\mathrm{F}}}{2}r_{\max}^2 \log\left(1 - \left(\frac{r}{r_{\max}}\right)^2\right). \tag{3.22}$$

The logarithmic term has a singularity at $r = r_{\max}$ (see fig. 3.1), limiting the maximum extension of an N-bead chain to the contour length $Nr_{\max}$. The strength can be adjusted

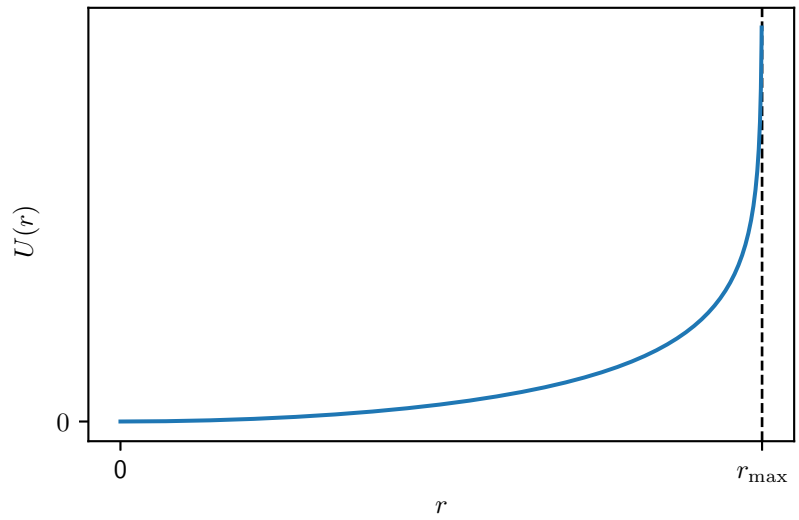


Figure 3.1: FENE potential.

by the parameter K_{F}.

A harmonic bond potential could be used as well, although this may lead to unphysical behavior in certain situations. For example, bonds can be infinitely stretched in an ideal extensional flow, a problem that also affects the harmonic dumbbells in the Oldroyd-B model.

3.1.5 Lennard-Jones Cosine potential

We use a modified version of the Lennard-Jones potential, the Lennard-Jones Cosine (LJC) potential [60, 61], as non-bonded pair interaction. Via the parameter ϕ in

$$U_{\mathrm{LJC}}(r) = \begin{cases} 4\varepsilon_{\mathrm{LJ}}\left[\left(\frac{\sigma_{\mathrm{LJ}}}{r}\right)^{12} - \left(\frac{\sigma_{\mathrm{LJ}}}{r}\right)^{6} + \frac{1}{4}\right] - \varepsilon_{\mathrm{LJ}}\phi & r \leq 2^{1/6}\sigma_{\mathrm{LJ}} \\ \frac{1}{2}\phi\varepsilon_{\mathrm{LJ}}\left(\cos\left[\alpha\left(\frac{r}{\sigma_{\mathrm{LJ}}}\right)^{2} + \beta\right] - 1\right) & 2^{1/6}\sigma_{\mathrm{LJ}} < r \leq r_{\mathrm{c}} \\ 0 & \text{else} \end{cases}, \tag{3.23}$$

it is possible to adjust the depth of the minimum of the potential at $r_{\min} = 2^{1/6}\sigma_{\mathrm{LJ}}$ (c.f. fig. 3.2) and thereby adjust the solvent quality. The constants α and β are chosen such that U_{LJC} is differentiable at any $r > 0$ and in particular at $r = 2^{1/6}\sigma_{\mathrm{LJ}}$ and $r = r_{\mathrm{c}}$, resulting in the conditions

$$\alpha 2^{1/3} + \beta = \pi, \qquad \alpha\left(\frac{r_{\mathrm{c}}}{\sigma_{\mathrm{LJ}}}\right)^{2} + \beta = 2\pi. \tag{3.24}$$

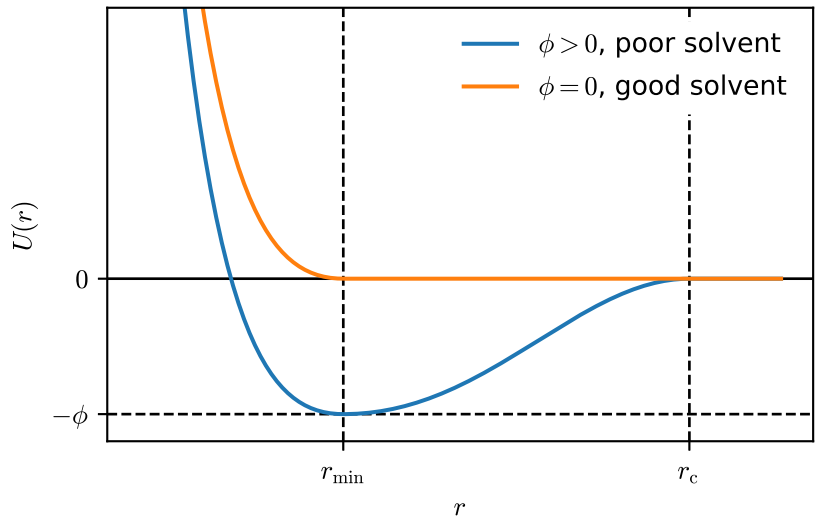


Figure 3.2: Lennard-Jones Cosine potential at different attraction strengths ϕ.

For $r_c = (3/2)\sigma_{LJ}$, which we use here, this system of equations is solved by the approximate numerical values

$$\alpha = 3.1730728678, \qquad \beta = -0.85622864544. \tag{3.25}$$

A polynomial continuation can be used in place of the cosine function. However, this does not improve performance significantly since the implementation of trigonometric functions is usually very efficient [60]. This potential defines natural units for length σ_{LJ}, energy ε_{LJ}, and thereby time $t_{LJ} = \sqrt{(m\sigma_{LJ}^2)/\varepsilon_{LJ}}$, where m is the particle mass.

For the simulation of phase separation, the system is first equilibrated with an athermal solvent $\phi = 0$, i.e. purely repulsive interaction. The quench is then performed by instantaneously turning on the attractive interaction, see e.g. section 5.4. The value of ϕ must, in this case, be chosen large enough for the polymer chains to collapse. The value at which this so called theta collapse occurs in three dimensions is $\phi_\Theta = 0.65 \pm 0.02$ [61]. In two dimensions the value of ϕ_Θ is estimated in section 5.2 to be roughly 1.5.

3.1.6 2D confinement

We use the software package ESPResSo++ [72] for simulating a coupled MD-LB system. At the time of writing, only three-dimensional Lattice Boltzmann (LB) simulations (see section 3.2) are supported by the software. We are mainly interested in 3D systems in the long term, and an implementation of a 2D LB algorithm is associated with substantial effort. Hence we seek to perform 2D simulations with the software at hand without spending much time on code development.

We mimic a two-dimensional system using a box in the form of a thin slab with a small extension in the z-direction, $L_x = L_y \gg L_z$. The particles in the box are confined in the $z = 0$ plane by an external potential U_{zc} that conforms to the periodic boundary conditions and has a minimum at $z = 0$:

$$U_{\mathrm{zc}}(z) = U_{\mathrm{zc}}(z + L_z), \qquad U'_{\mathrm{zc}}(0) = 0, \qquad U''_{\mathrm{zc}}(0) > 0. \tag{3.26}$$

We choose the particular form

$$U_{\mathrm{zc}}(z) = \frac{U_0}{2}\Big(1 - \cos\Big(\frac{2\pi}{L_z}z\Big)\Big) \tag{3.27}$$

with the corresponding force

$$F_{\mathrm{zc}}(z) = -\frac{\partial}{\partial z}U_{\mathrm{zc}}(z) = \frac{\pi U_0}{L_z}\sin\Big(\frac{2\pi}{L_z}z\Big). \tag{3.28}$$

The parameter U_0 gives the maximum height of the potential; we should therefore choose $U_0 \gg k_{\mathrm{B}}T$. This ensures that fluctuations in the z-direction are small and chain molecules can not cross each other. Furthermore, the box must be large enough in the z direction such that the molecules do not interact with their periodic images. With this assumption, $1 - \cos(x) \approx x^2/2$ and the harmonic approximation of the potential

$$\tilde{U}_{\mathrm{zc}}(z) = U_0\Big(\frac{\pi}{L_z}z\Big)^2 \tag{3.29}$$

should be fairly accurate. The oscillation period of a particle of mass m inside this potential is then

$$\tau = L_z\sqrt{\frac{2m}{U_0}}. \tag{3.30}$$

As a rule of thumb, the timestep δt_{MD} of the MD simulation should be chosen such that there are roughly 50 integrations per oscillation period, $\delta t_{\mathrm{MD}} < \tau/50$. This means that the strength of the potential should be chosen according to

$$U_0 < 2m\Big(\frac{L_z}{50\delta t_{\mathrm{MD}}}\Big)^2. \tag{3.31}$$

For the two-dimensional numerical simulations presented in chapter 5 the values $m = 1$, $L_z = 4$ and $\delta t_{\mathrm{MD}} = 0.005$ are used, resulting in a bound for the potential strength of $U_0 = 512$, which is indeed large compared to $k_{\mathrm{B}}T$.

3.2 Lattice Boltzmann simulations

In this section the basic principles of the Lattice Boltzmann (LB) simulation method [73–75] shall be outlined. In particular, the Maxwell-Boltzmann constraints are introduced. For this mathematical problem, a semi-automatic solution method, which was also implemented in a Python script, is provided in section 4.2.

We will show how to introduce thermal fluctuations in terms of a multi relaxation time (MRT) collision scheme [76–78] and how the thermalized fluid can be coupled to soft matter particles. This coupling scheme was used to generate the numerical results shown in chapter 5. It also provides the microscopic picture on which the viscoelastic phase separation model derived in section 4.1 is based. The derivations in the present section are mainly in line with ref. [73].

3.2.1 Introduction

The Lattice Boltzmann method is a kinetic approach to simulate hydrodynamics on a discrete lattice originating from lattice gas automata. Each lattice point can be thought of as containing a certain density of particles. An exchange of particles is only allowed between a finite number of lattice sites in a defined neighborhood. Hence, there is a discrete set of allowed velocities $\{\mathbf{c}_i\}$, which can include the velocity with magnitude zero. With the Lattice Boltzmann timestep δt_{LB}, $\delta t_{\mathrm{LB}}\mathbf{c}_i$ are displacement vectors connecting lattice sites. Larger sets of velocities can increase precision and stability of the numerics at the cost of computational effort, see section 4.2. Each of the velocities $\mathbf{c}_i$ has a separate mass density $n_i(\mathbf{r}, t)$, which we shall also call population, associated with it. The fluid density at the lattice site position $\mathbf{r}$ can then be obtained by taking the sum

$$\rho(\mathbf{r}, t) = \sum_i n_i(\mathbf{r}, t). \tag{3.32}$$

Analogously, the momentum density is obtained by taking the sum weighted by the velocities

$$\mathbf{j}(\mathbf{r}, t) = \sum_i n_i(\mathbf{r}, t)\mathbf{c}_i =: \rho(\mathbf{r}, t)\mathbf{v}(\mathbf{r}, t), \tag{3.33}$$

defining at the same time the local flow velocity $\mathbf{v}(\mathbf{r}, t)$. The population dynamics is described by the Lattice Boltzmann equation (LBE):

$$n_i(\mathbf{r} + \mathbf{c}_i\delta t_{\mathrm{LB}}, t + \delta t_{\mathrm{LB}}) =: n_i' = n_i(\mathbf{r}, t) + \Delta_i(\mathbf{r}, t). \tag{3.34}$$

Here the collision operator Δ_i has been introduced. The collision operator promotes the relaxation of the populations towards their local equilibrium value and most generally

takes the form

$$\Delta_i(\mathbf{r},t) = \sum_j L_{ij}\Big(n_j(\mathbf{r},t) - n_j^{(\mathrm{eq})}(\rho,\mathbf{v})\Big), \tag{3.35}$$

where the matrix $\mathbf{L}$ models dissipative processes ('collisions') that occur between populations i and j. In the especially simple case where $L_{ij} = -\delta_{ij}/\tau$, Δ is called the Bhatnagar-Gross-Krook (BGK) collision operator. Generally, the collisions must conserve mass and momentum:

$$\sum_i \Delta_i = \sum_i \mathbf{c}_i \Delta_i = 0. \tag{3.36}$$

From here one can show via dual time scale Chapman-Enskog expansion [73] that the Lattice Boltzmann eq. (3.34) delivers Navier-Stokes dynamics

$$\partial_t \rho + \partial_\beta(\rho v_\beta) = 0, \tag{3.37}$$
$$\partial_t(\rho v_\alpha) + \partial_\beta(\rho v_\alpha v_\beta) = -\partial_\alpha P + \partial_\beta \eta_{\alpha\beta\gamma\delta}\partial_\gamma v_\delta, \tag{3.38}$$

in the continuum limit of infinitely small timestep δt_{LB} and infinitely small lattice constant a. Alternatively we can write eq. (3.38) as

$$\partial_t v_\alpha + v_\beta \partial_\beta v_\alpha = -\frac{1}{\rho}\partial_\alpha P + \frac{1}{\rho}\eta_{\alpha\beta\gamma\delta}\partial_\beta\partial_\gamma v_\delta \tag{3.39}$$

Here, P is the pressure and $\boldsymbol{\eta}$ is the fourth-order viscosity tensor which can be expressed in terms of shear viscosity η_{s} and bulk viscosity η_{b} with

$$\begin{aligned}\eta_{\alpha\beta\gamma\delta} &= \eta_{\mathrm{s}}\Big(\delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma} - \frac{2}{3}\delta_{\alpha\beta}\delta_{\gamma\delta}\Big) + \eta_{\mathrm{b}}\delta_{\alpha\beta}\delta_{\gamma\delta} \\ &= \eta_{\mathrm{s}}(\delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}) + \Big(\eta_{\mathrm{b}} - \frac{2}{3}\eta_{\mathrm{s}}\Big)\delta_{\alpha\beta}\delta_{\gamma\delta}.\end{aligned} \tag{3.40}$$

Generally the viscosities η_{s}, η_{b} can depend on density, however we shall assume that they are constants for now.

At this point we should note that the model implies the equation of state of an ideal gas. Hence, with a particle mass of m, the pressure reads

$$P = \frac{\rho}{m}k_{\mathrm{B}}T =: \rho c_{\mathrm{s}}^2, \tag{3.41}$$

where the speed of sound c_{s} is determined by the relation $c_{\mathrm{s}}^2 = (\partial P/\partial\rho)_T$. This means that we can express temperature by $k_{\mathrm{B}}T = mc_{\mathrm{s}}^2$.

Determining the equilibrium populations $n_i^{(\mathrm{eq})}$ is non-trivial as they depend, among other factors, on the set of velocities used (c.f. section 4.2). Typically, one takes the Maxwell-Boltzmann distribution of an ideal gas

$$f(\mathbf{u}-\mathbf{v}) = \left(2\pi c_\mathrm{s}^2\right)^{-d/2} \exp\left(-\frac{(\mathbf{u}-\mathbf{v})^2}{2c_\mathrm{s}^2}\right), \tag{3.42}$$

where $\mathbf{u}(\mathbf{r},t)$ is the particle velocity, as a starting point. One then aims at reproducing its velocity moments up to a certain tensor order, i.e.

$$\frac{1}{\rho}\sum_i n_i^{(\mathrm{eq})}(\rho,\mathbf{v}) c_{i\alpha}c_{i\beta}\ldots c_{i\gamma} = \int \mathrm{d}^d\mathbf{u}\, f(\mathbf{u}-\mathbf{v}) u_\alpha u_\beta \ldots u_\gamma. \tag{3.43}$$

The equilibrium populations can then, for example, be expressed by a polynomial expansion in the flow velocity $\mathbf{v}$ where the coefficients are Hermite polynomials in the c_i, or by maximizing a suitably constructed entropy, see e.g. the appendix of [79]. A common ansatz for the equilibrium population is the second order polynomial

$$n_i^{(\mathrm{eq})}(\rho,\mathbf{v}) = \rho w_i\left(1 + A\mathbf{v}\cdot\mathbf{c}_i + B(\mathbf{v}\cdot\mathbf{c}_i)^2 + C\mathbf{v}^2\right), \tag{3.44}$$

where the w_i are weights with $w_i \geq 0$, $\sum_i w_i = 1$ that depend only on the magnitude of $\mathbf{c}_i$, and A, B and C are constant coefficients. For velocities $\mathbf{c}_i$ on a cubic lattice, symmetry dictates the following conditions

$$\sum_i w_i c_{i\alpha} = 0, \tag{3.45a}$$

$$\sum_i w_i c_{i\alpha}c_{i\beta} = C_2\delta_{\alpha\beta}, \tag{3.45b}$$

$$\sum_i w_i c_{i\alpha}c_{i\beta}c_{i\gamma} = 0, \tag{3.45c}$$

$$\sum_i w_i c_{i\alpha}c_{i\beta}c_{i\gamma}c_{i\delta} = C_4(\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}) + C_4'\delta_{\alpha\beta\gamma\delta}, \tag{3.45d}$$

$$\vdots$$

In the last line, the fourth-order Kronecker delta $\delta_{\alpha\beta\gamma\delta}$ was introduced. It is equal to one only if all indexes have the same value and zero otherwise. We shall call the objects on the left hand side of eq. (3.45) the n-th order moment tensors. It is easy to see that the odd-order moments always vanish. The above conditions are symmetric under the exchange of indices. However, the moment tensors also need to satisfy rotation symmetry, which implies $C_4' = 0$ [80]. The highest tensor order M up to which the isotropy conditions can be satisfied depends on the choice of the $\mathbf{c}_i$. For isothermal hydrodynamics, fourth-order isotropy is sufficient, while non-isothermal hydrodynamics or multiphase flows need

isotropy up to sixth order. Generally, more complex physical phenomena require a higher degree of isotropy and thus a larger set of velocities.

The symmetry conditions eq. (3.45) in conjunction with $\sum_i w_i = 1$ now allows us to compute the first three moments of the equilibrium population ansatz eq. (3.44):

$$\sum_i n_i^{(\mathrm{eq})} = \rho\Big(1 + \mathbf{v}^2(BC_2 + C)\Big) \tag{3.46a}$$

$$\sum_i n_i^{(\mathrm{eq})} c_{i\alpha} = \rho A C_2 v_\alpha \tag{3.46b}$$

$$\sum_i n_i^{(\mathrm{eq})} c_{i\alpha} c_{i\beta} = \rho\Big(\delta_{\alpha\beta}[C_2 + \mathbf{v}^2(BC_4 + CC_2)] + 2BC_4 v_\alpha v_\beta\Big) \tag{3.46c}$$

On the other hand, the first three moments of the Maxwell-Boltzmann distribution are

$$\rho \int \mathrm{d}^3\mathbf{u}\, f(\mathbf{u} - \mathbf{v}) = \rho, \tag{3.47a}$$

$$\rho \int \mathrm{d}^3\mathbf{u}\, f(\mathbf{u} - \mathbf{v}) u_\alpha = \rho v_\alpha = j_\alpha, \tag{3.47b}$$

$$\rho \int \mathrm{d}^3\mathbf{u}\, f(\mathbf{u} - \mathbf{v}) u_\alpha u_\beta = \rho c_\mathrm{s}^2 \delta_{\alpha\beta} + \rho v_\alpha v_\beta =: \pi_{\alpha\beta}, \tag{3.47c}$$

where we have introduced the Euler stress $\pi_{\alpha\beta}$ in eq. (3.47c). Matching the moments eq. (3.46) with eq. (3.47) results in the conditions for the parameters

$$BC_2 + C = 0, \tag{3.48}$$

$$AC_2 = 1, \tag{3.49}$$

$$C_2 = c_\mathrm{s}^2, \tag{3.50}$$

$$BC_4 + CC_2 = 0, \tag{3.51}$$

$$2BC_4 = 1, \tag{3.52}$$

which allows us to write down the equilibrium populations

$$n_i^{(\mathrm{eq})}(\rho, \mathbf{v}) = w_i \rho \left(1 + \frac{\mathbf{v} \cdot \mathbf{c}_i}{c_\mathrm{s}^2} + \frac{(\mathbf{v} \cdot \mathbf{c}_i)^2}{2{c_\mathrm{s}}^4} - \frac{u^2}{2c_\mathrm{s}^2}\right). \tag{3.53}$$

One should note that the expansion of the equilibrium populations up to second order in $\mathbf{v}$ is suitable for models with fourth-order isotropy.

3.2.2 The D3Q19 model

One of the most widely used LB models is the three-dimensional model on a cubic lattice with 19 velocities called the D3Q19 model. Here, the velocities are arranged in three shells

of equal magnitude like

$$
\begin{aligned}
S_0 &= \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right\}, && \text{(1 velocity)} \\
S_1 &= \left\{ \begin{pmatrix} \pm 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \pm 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix} \right\}, && \text{(6 velocities)} \\
S_2 &= \left\{ \begin{pmatrix} \pm 1 \\ \pm 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \pm 1 \\ \pm 1 \end{pmatrix}, \begin{pmatrix} \pm 1 \\ 0 \\ \pm 1 \end{pmatrix}, \begin{pmatrix} \pm 1 \\ \mp 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \pm 1 \\ \mp 1 \end{pmatrix}, \begin{pmatrix} \pm 1 \\ 0 \\ \mp 1 \end{pmatrix} \right\}, && \text{(12 velocities)}
\end{aligned}
\tag{3.54}
$$

where a is the lattice spacing the velocities are given in units of $a/\delta t_{\mathrm{LB}}$. This set can be shown to satisfy isotropy up to fourth order for suitably chosen coefficients.

By setting $\mathbf{v} = 0$ in eq. (3.43), we obtain the Maxwell-Boltzmann constraints

$$
\sum_i w_i c_{i\alpha} c_{i\beta} \dots c_{i\gamma} = \int \mathrm{d}^3\mathbf{u}\, f(\mathbf{u}) u_\alpha u_\beta \dots u_\gamma, \tag{3.55}
$$

from which one can determine the weights

$$
w_i = \begin{cases} \frac{1}{3} : & \mathbf{c}_i \in S_0 \\ \frac{1}{18} : & \mathbf{c}_i \in S_1 \\ \frac{1}{36} : & \mathbf{c}_i \in S_2 \end{cases} \tag{3.56}
$$

with a speed of sound of

$$
c_{\mathrm{s}}^2 = \frac{1}{3}\left(\frac{a}{\delta t_{\mathrm{LB}}}\right)^2. \tag{3.57}
$$

Manually determining the weights is typically no easy task and becomes more difficult with larger velocity sets and increasing dimensionality. A way of determining the weights in an automated fashion for arbitrary velocity sets, provided that these sets admit a sufficient degree of isotropy, is presented in section 4.2.

3.2.3 Multiple relaxation time scheme

For the numerical simulations in this work, the multiple relaxation time (MRT) scheme [77, 78] for the collision operator is used.

Here, the dimensionless velocity vectors $\hat{\mathbf{c}}_i := \delta t_{\mathrm{LB}} \mathbf{c}_i / a$ generate an orthogonal basis for the diagonal representation of the collision matrix L_{ij}, which was introduced in eq. (3.35). The 19-dimensional basis vectors $\mathbf{e}_k$ are listed in table 3.1. They can be constructed from

k	e_{ki}	b_k
0	1	1
1	$\hat{c}_{ix}$	1/3
2	$\hat{c}_{iy}$	1/3
3	$\hat{c}_{iz}$	1/3
4	$\hat{c}_i^2 - 1$	2/3
5	$\hat{c}_{ix}^2 - \hat{c}_i^2$	4/3
6	$\hat{c}_{iy}^2 - \hat{c}_{iz}^2$	4/9
7	$\hat{c}_{ix}\hat{c}_{iy}$	1/9
8	$\hat{c}_{iy}\hat{c}_{iz}$	1/9
9	$\hat{c}_{iz}\hat{c}_{ix}$	1/9
10	$(3\hat{c}_i^2 - 5)\hat{c}_{ix}$	2/3
11	$(3\hat{c}_i^2 - 5)\hat{c}_{iy}$	2/3
12	$(3\hat{c}_i^2 - 5)\hat{c}_{iz}$	2/3
13	$(\hat{c}_{iy}^2 - \hat{c}_{iz}^2)\hat{c}_{ix}$	2/9
14	$(\hat{c}_{iz}^2 - \hat{c}_{ix}^2)\hat{c}_{iy}$	2/9
15	$(\hat{c}_{ix}^2 - \hat{c}_{iy}^2)\hat{c}_{iz}$	2/9
16	$3\hat{c}_i^4 - 6\hat{c}_i^2 + 1$	2
17	$(2\hat{c}_i^2 - 3)(3\hat{c}_{ix}^2 - \hat{c}_i^2)$	4/3
18	$(2\hat{c}_i^2 - 3)(\hat{c}_{iy}^2 - \hat{c}_{iz}^2)$	4/9

Table 3.1: MRT basis vectors of the D3Q19 model with associated normalization factors, c.f. [73].

the $\hat{\mathbf{c}}_i$ via Gram-Schmidt orthogonalization and this set is not unique. Orthogonality is understood with respect to the previously defined set of weights w_i:

$$\sum_i w_i e_{ki} e_{li} = b_k \delta_{kl}. \tag{3.58}$$

In principle different weights could be used as well, however the same symmetry conditions must apply. The corresponding normalization factors b_k can easily be retrieved from

$$b_k = \sum_i w_i (e_{ki})^2. \tag{3.59}$$

From the Chapman-Enskog expansion, it becomes evident that the collision process does not touch the equilibrium component of the populations. With $n_i = n_i^{(\mathrm{eq})} + n_i^{(\mathrm{neq})}$, $n_i'(\mathbf{r}, t) = n_i(\mathbf{r} + \mathbf{c}_i \delta t_{\mathrm{LB}}, t + \delta t_{\mathrm{LB}})$ we can thus write

$$n_i'^{(\mathrm{eq})} = n_i^{(\mathrm{eq})}, \tag{3.60}$$

$$n_i'^{(\mathrm{neq})} = \sum_j (\delta_{ij} + L_{ij}) n_i^{(\mathrm{neq})}. \tag{3.61}$$

By defining the transformed populations,

$$m_k = \sum_i e_{ki} n_i, \tag{3.62}$$

the Lattice Boltzmann equation then can be written in the compact form

$$m_k'^{(\mathrm{neq})} = \gamma_k m_k^{(\mathrm{neq})}. \tag{3.63}$$

It can be shown that the modes $m_0, \dots m_3$ correspond to the conserved quantities ρ and $\mathbf{j}$. Thus $\gamma_0, \dots \gamma_3$ do not contribute and can be set to zero. The value $\gamma_4 =: \gamma_\mathrm{b}$ is associated with the bulk stress while $\gamma_5, \dots, \gamma_9 =: \gamma_s$ must have equal value due to symmetry and are associated to shear stress. Bulk and shear viscosities can be expressed by

$$\eta_\mathrm{b} = \frac{\delta t_\mathrm{LB} \rho c_\mathrm{s}^2}{3} \frac{1+\gamma_\mathrm{b}}{1-\gamma_\mathrm{b}}, \tag{3.64}$$

$$\eta_\mathrm{s} = \frac{\delta t_\mathrm{LB} \rho c_\mathrm{s}^2}{2} \frac{1+\gamma_\mathrm{s}}{1-\gamma_\mathrm{s}}. \tag{3.65}$$

One of the key advantages of the MRT scheme becomes apparent here, which is the fact that bulk and shear viscosity can be adjusted independently.

The higher modes are kinetic ones and can be split into three groups with distinct γ-values. These values can be set to zero for most purposes, which means that the corresponding modes relax instantaneously.

3.2.4 Thermal fluctuations and coupling to particles

When coupling the LB fluid to Molecular Dynamics (MD) particles, it is vital that the correct hydrodynamic fluctuations are present in order to reproduce Brownian motion.

The implementation of thermal fluctuations can be done by adding a stochastic term to the collision operator:

$$\Delta_i = \sum_i L_{ij} n_j^{(\mathrm{neq})} + \Delta_i^{(\mathrm{f})}. \tag{3.66}$$

Because the new operator has to satisfy mass and momentum conservation as well, it is required that

$$\sum_i \Delta_i^{(\mathrm{f})} = \sum_i \Delta_i^{(\mathrm{f})} \mathbf{c}_i = 0. \tag{3.67}$$

While the average must be zero, $\left\langle \Delta_i^{(\mathrm{f})} \right\rangle = 0$, the covariance $\left\langle \Delta_i^{(\mathrm{f})} \Delta_j^{(\mathrm{f})} \right\rangle$ must yield the correct relation for the hydrodynamic fluctuating stress $\sigma_{\alpha\beta}^{(\mathrm{f})}$,

$$\left\langle \sigma_{\alpha\beta}^\mathrm{f} \sigma_{\gamma\delta}^\mathrm{f} \right\rangle = \frac{2 k_\mathrm{B} T}{a^3 \delta t_\mathrm{LB}} \eta_{\alpha\beta\gamma\delta}. \tag{3.68}$$

In the continuum limit, this stress enters the dynamic equation for the velocity field like [81]

$$\partial_t(\rho v_\alpha) + \partial_\beta(\rho v_\alpha v_\beta) = -\partial_\alpha P + \eta_{\alpha\beta\gamma\delta} \partial_\beta \partial_\gamma v_\delta + \partial_\beta \sigma_{\alpha\beta}^{(\mathrm{f})}. \tag{3.69}$$

The fluctuating Lattice Boltzmann equation with the desired continuum limit has the form

$$m_k'^{(\mathrm{neq})} = \gamma_k m^{(\mathrm{neq})} + \sqrt{\frac{w_k m \rho}{a^3}(1-\gamma_k^2)} R_k. \tag{3.70}$$

Here, the R_k are Gaussian random variables with $\langle R_k \rangle = 0$ and $\langle R_k R_l \rangle = \delta_{kl}$.

This is now a good starting point to consider coupling to MD particles. Consider a set of particles i with mass m, positions $\mathbf{r}_i$ and momenta $\mathbf{p}_i$. The particles feel conservative forces $\mathbf{F}_i^{(\mathrm{cons})} = -\partial V / \partial \mathbf{r}_i$ as well as random forces $\mathbf{F}_i^{(\mathrm{f})}$ from thermal fluctuations with

$$\left\langle F_{i\alpha}^{\mathrm{f}} \right\rangle = 0, \tag{3.71}$$

$$\left\langle F_{i\alpha}^{\mathrm{f}}(t) F_{j\beta}^{\mathrm{f}}(t') \right\rangle = 2 k_{\mathrm{B}} T \zeta \delta_{ij} \delta_{\alpha\beta} \delta(t-t'). \tag{3.72}$$

Coupling to the surrounding fluid is facilitated by Stokes friction forces with friction coefficient ζ, which are proportional to the relative velocity,

$$\mathbf{F}_i^{(\mathrm{diss})} = -\zeta \Big(\frac{\mathbf{p}_i}{m} - \mathbf{v}(\mathbf{r}_i) \Big), \tag{3.73}$$

where $\mathbf{v}(\mathbf{r}_i)$ denotes the flow velocity at the position of the particle i. While the positions of the MD particles are continuous in space, the velocity of the LB fluid is only defined on the lattice sites at $\mathbf{R}_j$. We may thus view the velocity field as a sum of weighted delta functions

$$\mathbf{v}(\mathbf{r}) = \sum_j \mathbf{v}(\mathbf{R}_j) \delta(\mathbf{r} - \mathbf{R}_j). \tag{3.74}$$

In order to interpolate the fluid velocities from the surrounding lattice sites to the particle positions $\mathbf{r}_i$, we introduce a normalized weighting function with compact support $\omega(\mathbf{r}, \mathbf{r}_i)$ such that the interpolated velocity is given by

$$\mathbf{v}(\mathbf{r}_i) = \int \mathrm{d}^3\mathbf{r}\, \omega(\mathbf{r}, \mathbf{r}_i) \mathbf{v}(\mathbf{r}). \tag{3.75}$$

This then allows the computation of the dissipative force eq. (3.73), which can easily be incorporated in the MD algorithm. As a result of this force and the random force, the momentum of the particles changes and momentum conservation requires that the equal but opposite momentum is likewise transferred to the fluid. This is facilitated by a force density $\mathbf{f}$, which is found by extrapolating both dissipative and random forces to the lattice sites via the same function ω:

$$\mathbf{f}(\mathbf{r}) = -\sum_i \omega(\mathbf{r}, \mathbf{r}_i) \Big(\mathbf{F}_i^{(\mathrm{diss})} + \mathbf{F}_i^{(\mathrm{f})} \Big). \tag{3.76}$$

The equations of motion for the coupled system are then given by

$$\dot{\mathbf{r}}_i = \frac{\mathbf{p}_i}{m_i}, \tag{3.77}$$

$$\dot{\mathbf{p}}_i = \mathbf{F}_i^{(\mathrm{cons})} + \mathbf{F}_i^{(\mathrm{diss})} + \mathbf{F}_i^{(\mathrm{f})}, \tag{3.78}$$

$$\partial_t \rho + \partial_\alpha j_\alpha = 0, \tag{3.79}$$

$$\partial_t(\rho v_\alpha) + \partial_\beta(\rho v_\alpha v_\beta) = -\partial_\alpha P + \eta_{\alpha\beta\gamma\delta}\partial_\beta\partial_\gamma v_\delta + \partial_\beta \sigma^{(\mathrm{f})}_{\alpha\beta} + f_\alpha. \tag{3.80}$$

It can be shown that this set of equations satisfies the fluctuation-dissipation theorem as well as conservation of total mass and momentum.

The force density $\mathbf{f}$ can be incorporated into the LB algorithm by yet another contribution $\Delta^{(c)}$ to the collision operator such that the total operator is given by $\Delta + \Delta^{(\mathrm{f})} + \Delta^{(c)}$. While this additional contribution must still conserve mass, it changes the momentum by $\delta t_{\mathrm{LB}}\mathbf{f}$, hence

$$\sum_i \Delta_i^{(c)} = 0, \qquad \sum_i \Delta_i^{(c)}\mathbf{c}_i = \delta t_{\mathrm{LB}}\mathbf{f}. \tag{3.81}$$

This makes the definition of the instantaneous momentum somewhat arbitrary. Here we shall use

$$\mathbf{j} := \sum_i n_i \mathbf{c}_i + \frac{1}{2}\delta t_{\mathrm{LB}}\mathbf{f}, \qquad \mathbf{v} = \frac{\mathbf{j}}{\rho}, \tag{3.82}$$

which is the optimal definition from a numerical viewpoint. It can be shown via Chapman-Enskog analysis that in the MRT framework $\Delta^{(c)}$ takes the form

$$\Delta_i^{(c)} = w_i\left[\frac{\delta t_{\mathrm{LB}}}{c_{\mathrm{s}}^2} f_\alpha c_{i\alpha} + \frac{\delta t_{\mathrm{LB}}}{2c_{\mathrm{s}}^4}\Sigma_{\alpha\beta}\Big(c_{i\alpha}c_{i\beta} - c_{\mathrm{s}}^2\delta_{\alpha\beta}\Big)\right] \tag{3.83}$$

with

$$\Sigma_{\alpha\beta} = \frac{1}{2}(1+\gamma_{\mathrm{s}})\Big(v_\alpha f_\beta + v_\beta f_\alpha - \frac{2}{3}v_\gamma f_\gamma \delta_{\alpha\beta}\Big) + \frac{1}{3}(1+\gamma_{\mathrm{b}})v_\gamma f_\gamma \delta_{\alpha\beta}. \tag{3.84}$$

The MD-LB coupling scheme that is outlined in this section is implemented in the simulation package ESPResSo++ [72,82] which was used to perform the simulations in the present work. Efficiency can be improved by performing the relatively costly LB steps only every several MD steps and hence choosing the ratio $\delta t_{\mathrm{LB}}/\delta t_{\mathrm{MD}}$ to be a natural number larger than one [83].

Chapter 4

Computational and theoretical developments

This chapter compiles some of the theoretical and computational concepts that have been developed.

In section 4.1, continuum equations for viscoelastic phase separation are derived from a microscopic interpretation that is very close to the LB/MD coupling scheme outlined in section 3.2.4. In the derivation of the equations, conservative and dissipative contributions to the dynamics are always kept separate. While the conservative dynamics is derived with the Poisson bracket formalism introduced in section 2.7, the dissipative coupling is constructed in such a way that it is compatible with the GENERIC formalism presented in section 2.8. Due to the simplicity of the underlying microscopic model, the equations can be derived without taking any approximations other than postulating continuum expressions for the Hamiltonian and dissipation rate. Only at the very end, several approximations are taken, significantly reducing the number and complexity of the dynamic equations. A publication of this found at [56].

Section 4.2 addresses the generation of Lattice Boltzmann models, particularly the determination of the weights needed to construct the equilibrium distributions for the collision process. This is done by mapping the Maxwell-Boltzmann constraints, which are tensor valued equations, onto an equivalent linear system of equations. In doing so, the speed of sound of the model is left variable and, if a solution exists, the weights can be written as polynomials in the speed of sound. It is shown how ranges of values for the speed of sound for which all weights are positive can be identified. Only then a physically consistent Lattice Boltzmann model can be constructed. The procedure was implemented in the publicly available Python script `LBWeights.py` [79, 84]. At this point, it should be mentioned that the main idea of the theoretical development in this particular section is due to

the work of B. Dünweg, and the contribution of the author is mainly the implementation and computational aspects of the Python script as well as the finding of higher-order LB models that were hitherto unknown in the literature.

4.1 Viscoelastic Model H

In the present section, equations for the continuum dynamics of viscoelastic phase separation are derived. Similar to Model H (section 2.4.3), mass is conserved and hydrodynamic flow is taken into account. Furthermore, one component is considered macromolecular and thus viscoelastic, motivating the term Viscoelastic Model H or VEMH. Like in the Oldroyd-B model (section 2.5), polymer molecules in a solvent are approximated by harmonic dumbbells that are coupled to a background velocity field. Instead of formulating the Langevin equations for the dumbbell and solving the corresponding Fokker-Planck equation for the end-to-end distribution function, we here determine the conservative part of dynamics via the Poisson bracket formalism as described in section 2.7. Coupling terms are constructed in such a way that they are compatible with the GENERIC formalism outlined in section 2.8. Only at the very end, several approximations are taken, and the set of equations is significantly reduced and simplified.

In the microscopic interpretation, a molecule is represented by a dumbbell consisting of two particles with mass m located at positions $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$ which are connected by a harmonic spring with spring constant k (fig. 4.1). To describe the position and orientation

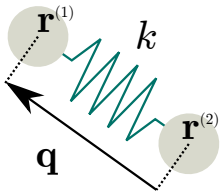


Figure 4.1: Illustration of a dumbbell molecule

of a dumbbell, the center of mass and relative vector

$$\mathbf{R} = \frac{\mathbf{r}^{(1)} + \mathbf{r}^{(2)}}{2}, \qquad \mathbf{q} = \mathbf{r}^{(1)} - \mathbf{r}^{(2)} \tag{4.1}$$

are introduced. Note that the definition of the relative vector is somewhat arbitrary as $\mathbf{q}$ and $-\mathbf{q}$ describe the same dumbbell orientation. This symmetry, which we shall call flip invariance, must be reflected in the final dynamic equations.

In analogy to the LB/MD coupling scheme (section 3.2.4), the solvent background is considered to be a Navier-Stokes continuum that is coupled to the polymer component via friction forces. The dynamic variables of the solvent are its density $\rho^{(\mathrm{s})}(\mathbf{r}, t)$ and momentum density $\mathbf{j}^{(\mathrm{s})}(\mathbf{r}, t)$. The velocity field is then defined by $\mathbf{v}^{(\mathrm{s})} = \mathbf{j}^{(\mathrm{s})}/\rho^{(\mathrm{s})}$. The dynamics is given by the Navier-Stokes equations (sum convention is implied for repeated

Greek indexes)

$$\partial_t \rho^{(\mathrm{s})} = -\partial_\beta j_\beta^{(\mathrm{s})}, \tag{4.2}$$

$$\partial_t j_\alpha^{(\mathrm{s})} = -\partial_\beta \Big(P^{(\mathrm{s})} \delta_{\alpha\beta} + j_\alpha^{(\mathrm{s})} v_\beta^{(\mathrm{s})} \Big) + \eta_{\alpha\beta\gamma\delta} \partial_\beta \partial_\gamma v_\delta^{(\mathrm{s})} + g_\alpha, \tag{4.3}$$

where $P^{(\mathrm{s})}(\mathbf{r}, t)$ is the partial pressure of the solvent and $\eta_{\alpha\beta\gamma\delta}$ is the fourth-order viscosity tensor as defined in eq. (3.40). The term g_α is a force density and is used to model the coupling to the dumbbell component via Strokes-friction forces. In vector notation this takes the more compact form

$$\partial_t \rho^{(\mathrm{s})} = -\boldsymbol{\nabla} \cdot \mathbf{j}^{(\mathrm{s})}, \tag{4.4}$$

$$\partial_t \mathbf{j}^{(\mathrm{s})} = -\boldsymbol{\nabla} P^{(\mathrm{s})} - \boldsymbol{\nabla} \cdot \left(\mathbf{j}^{(\mathrm{s})} \mathbf{v}^{(\mathrm{s})} \right) + \boldsymbol{\eta} \vdots \boldsymbol{\nabla}\boldsymbol{\nabla} \mathbf{v}^{(\mathrm{s})} + \mathbf{g}. \tag{4.5}$$

4.1.1 Hamiltonian

To determine the conservative dynamics of the dumbbell component with the Poisson bracket formalism, we need a continuum Hamiltonian that has a direct and obvious connection to the underlying microscopic picture.

The potential energy of a single dumbbell is given by the energy stored in the harmonic spring, $E_{\mathrm{pot}} = k\mathbf{q}^2/2$. The kinetic energy in terms of center of mass and relative coordinates can be expressed by

$$E_{\mathrm{kin}} = \frac{m}{2} \left[\left(\dot{\mathbf{r}}^{(1)} \right)^2 + \left(\dot{\mathbf{r}}^{(2)} \right)^2 \right] = m \dot{\mathbf{R}}^2 + \frac{m}{4} \dot{\mathbf{q}}^2 = \frac{m^{(\mathrm{d})}}{2} \dot{\mathbf{R}}^2 + \frac{m^{(\mathrm{r})}}{2} \dot{\mathbf{q}}^2. \tag{4.6}$$

Although we could write all equations in terms of just the particle mass m, we introduce the total dumbbell mass $m^{(\mathrm{d})} = 2m$ and the reduced mass $m^{(\mathrm{r})} = m/2$ for the relative motion in order to emphasize the connection with the classical mechanics of two-body systems. We can then formulate the microscopic Lagrangian of a system containing multiple dumbbell molecules:

$$\widehat{\mathscr{L}}^{(\mathrm{d})} = E_{\mathrm{kin}} - E_{\mathrm{pot}} = \frac{m^{(\mathrm{d})}}{2} \sum_i \dot{\mathbf{R}}_i^2 + \frac{m^{(\mathrm{r})}}{2} \sum_i \dot{\mathbf{q}}_i^2 - \frac{k}{2} \sum_i \mathbf{q}_i^2 - U^{(\mathrm{nb})}, \tag{4.7}$$

where $U^{(\mathrm{nb})}$ is the potential energy due to non-bonded interactions. From the Lagrangian follow the conjugate momenta

$$\mathbf{p}_i^{(\mathrm{d})} = \frac{\partial \widehat{\mathscr{L}}^{(\mathrm{d})}}{\partial \dot{\mathbf{R}}_i} = m^{(\mathrm{d})} \dot{\mathbf{R}}_i, \qquad \mathbf{p}_i^{(\mathrm{r})} = \frac{\partial \widehat{\mathscr{L}}^{(\mathrm{d})}}{\partial \dot{\mathbf{q}}_i} = m^{(\mathrm{r})} \dot{\mathbf{q}}_i. \tag{4.8}$$

For the continuum description of the dynamics, suitable fields with microscopic interpretation are needed. To that end, we introduce densities for center of mass and relative component by suitably weighted sums of delta functions centered at the center of mass of the dumbbells

$$\hat{\rho}^{(\mathrm{d})}(\mathbf{r}) = m^{(\mathrm{d})} \sum_i \delta(\mathbf{r} - \mathbf{R}_i), \tag{4.9}$$

$$\hat{\mathbf{k}}^{(\mathrm{r})}(\mathbf{r}) = k \sum_i \mathbf{q}_i \delta(\mathbf{r} - \mathbf{R}_i). \tag{4.10}$$

Analogously, we define the momentum densities for center of mass and relative momenta

$$\hat{\mathbf{j}}^{(\mathrm{d})}(\mathbf{r}) = \sum_i \mathbf{p}_i^{(\mathrm{d})} \delta(\mathbf{r} - \mathbf{R}_i), \tag{4.11}$$

$$\hat{\mathbf{j}}^{(\mathrm{r})}(\mathbf{r}) = \sum_i \mathbf{p}_i^{(\mathrm{r})} \delta(\mathbf{r} - \mathbf{R}_i). \tag{4.12}$$

The corresponding microscopic Hamiltonian is then obtained from the Lagrangian by the Legendre transformation

$$\widehat{\mathscr{H}}^{(\mathrm{d})} = \sum_i \left(\dot{\mathbf{R}}_i \cdot \mathbf{p}_i^{(\mathrm{d})} + \dot{\mathbf{q}}_i \cdot \mathbf{p}_i^{(\mathrm{r})} \right) - \widehat{\mathscr{L}}^{(\mathrm{d})} = \sum_i \left[\frac{\left(\mathbf{p}_i^{(\mathrm{d})}\right)^2}{2m^{(\mathrm{d})}} + \frac{\left(\mathbf{p}_i^{(\mathrm{r})}\right)^2}{2m^{(\mathrm{r})}} + \frac{k}{2}\mathbf{q}_i^2 \right] + U^{(\mathrm{nb})}. \tag{4.13}$$

Note that in the isothermal setting the Hamiltonian must be interpreted as Helmholtz free energy.

When switching to the continuum picture, we imagine to smooth out the delta peaks * such that the quantities eqs. (4.9) to (4.12) become continuous and differentiable fields. We denote this by $\hat{\rho}^{(\mathrm{d})} \to \rho^{(\mathrm{d})}$, $\hat{\mathbf{k}}^{(\mathrm{r})} \to \mathbf{k}^{(\mathrm{r})}$, $\hat{\mathbf{j}}^{(\mathrm{d})} \to \mathbf{j}^{(\mathrm{d})}$ and $\hat{\mathbf{j}}^{(\mathrm{r})} \to \mathbf{j}^{(\mathrm{r})}$, where the hat marks quantities in microscopic interpretation. Finally, with the mass ratio $\Gamma = m^{(\mathrm{r})}/m^{(\mathrm{d})}$, we write the Hamiltonian in terms of the macroscopic variables as

$$\mathscr{H}^{(\mathrm{d})} = \int \mathrm{d}^d\mathbf{r} \left[\frac{1}{2\rho^{(\mathrm{d})}} \left(\left(\mathbf{j}^{(\mathrm{d})}\right)^2 + \frac{1}{\Gamma}\left(\mathbf{j}^{(\mathrm{r})}\right)^2 + \frac{m^{(\mathrm{d})}}{k}\left(\mathbf{k}^{(\mathrm{r})}\right)^2 \right) + f + \frac{\kappa}{2}\left(\boldsymbol{\nabla}\rho^{(\mathrm{d})}\right)^2 \right]. \tag{4.14}$$

The non-bonded interaction is split up into an interface term with interfacial stiffness κ (for a motivation of the interface term see sections 2.4.2 and 2.4.3 or references [85–87]) and a thermodynamic free energy density $f^{(\mathrm{d})}$. The functional derivatives of the Hamiltonian

*In practice, this can, for example, be done by applying the Fourier transformation, discarding higher modes, and then doing the inverse transformation.

are then given by

$$\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\mathbf{k}^{(\mathrm{r})}} = \frac{m^{(\mathrm{d})}\mathbf{k}^{(\mathrm{r})}}{k\rho^{(\mathrm{d})}} =: \mathbf{q}, \tag{4.15}$$

$$\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\mathbf{j}^{(\mathrm{d})}} = \frac{\mathbf{j}^{(\mathrm{d})}}{\rho^{(\mathrm{d})}} =: \mathbf{v}^{(\mathrm{d})}, \tag{4.16}$$

$$\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\mathbf{j}^{(\mathrm{r})}} = \frac{\mathbf{j}^{(\mathrm{r})}}{\Gamma\rho^{(\mathrm{d})}} =: \mathbf{v}^{(\mathrm{r})}, \tag{4.17}$$

$$\begin{aligned}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\rho^{(\mathrm{d})}} &= \frac{\partial f^{(\mathrm{d})}}{\partial\rho^{(\mathrm{d})}} - \frac{1}{2(\rho^{(\mathrm{d})})^2}\left[\left(\mathbf{j}^{(\mathrm{d})}\right)^2 + \frac{1}{\Gamma}\left(\mathbf{j}^{(\mathrm{r})}\right)^2 + \frac{m^{(\mathrm{d})}}{k}\left(\mathbf{k}^{(\mathrm{r})}\right)^2\right] - \kappa\nabla^2\rho^{(\mathrm{d})} \\ &= \frac{\partial f^{(\mathrm{d})}}{\partial\rho^{(\mathrm{d})}} - \frac{1}{2}\left[\left(\mathbf{v}^{(\mathrm{d})}\right)^2 + \Gamma\left(\mathbf{v}^{(\mathrm{r})}\right)^2 + \frac{k}{m^{(\mathrm{d})}}\mathbf{q}^2\right] - \kappa\nabla^2\rho^{(\mathrm{d})}.\end{aligned} \tag{4.18}$$

Note that while the microscopic Hamiltonian $\widehat{\mathscr{H}}^{(\mathrm{d})}$ in eq. (4.13) is expressed as a function of canonically conjugated variables, the macroscopic Hamiltonian $\mathscr{H}^{(\mathrm{d})}$ in eq. (4.14) is a functional on a set of macroscopic fields that are not conjugate. However, by the Poisson bracket formalism, it is nevertheless possible to derive equations of motion since the macroscopic fields can be expressed in terms of the microscopic conjugate variables. Therefore, their Poisson brackets can be evaluated, which is the subject of the next section.

4.1.2 Poisson brackets

We proceed by deriving several auxiliary relations that will aid in the calculation of the Poisson brackets. For ease of notation, we introduce the shorthand $\delta_i = \delta(\mathbf{r} - \mathbf{R}_i)$, resp. $\delta_i' = \delta(\mathbf{r}' - \mathbf{R}_i)$, and take note of the elementary brackets

$$\left\{\delta_i, \delta_j'\right\} = 0, \tag{4.19}$$

$$\left\{R_{i\alpha}, p_{j\beta}^{(\mathrm{d})}\right\} = \left\{q_{i\alpha}, p_{j\beta}^{(\mathrm{r})}\right\} = \delta_{\alpha\beta}\delta_{ij}. \tag{4.20}$$

Furthermore,

$$\begin{aligned}\left\{\delta_i, p_{j\beta}^{(\mathrm{d})}\right\} &= \sum_k \frac{\partial\delta(\mathbf{r} - \mathbf{R}_i)}{\partial R_{k\gamma}}\frac{\partial p_{j\beta}^{(\mathrm{d})}}{\partial p_{k\gamma}^{(\mathrm{d})}} = \frac{\partial\delta(\mathbf{r} - \mathbf{R}_i)}{\partial R_{j\beta}} \\ &= -\delta_{ij}\frac{\partial\delta(\mathbf{r} - \mathbf{R}_i)}{\partial r_\beta} = -\delta_{ij}\partial_\beta\delta_i.\end{aligned} \tag{4.21}$$

Next, we note that for any field A of the form

$$A(\mathbf{r}) = \sum_i a_i\delta(\mathbf{r} - \mathbf{R}_i), \tag{4.22}$$

the relation

$$\begin{aligned}\sum_i a_i \delta_i' f(\mathbf{r}, \mathbf{R}_i) &= \sum_i \int \mathrm{d}^d\mathbf{r}''\, a_i \delta(\mathbf{r}'' - \mathbf{R}_i)\delta(\mathbf{r}' - \mathbf{r}'') f(\mathbf{r}, \mathbf{r}'') \\ &= A(\mathbf{r}') f(\mathbf{r}, \mathbf{r}')\end{aligned} \tag{4.23}$$

holds. Additionally, if the coefficient a_i neither depends on $\mathbf{R}_i$ nor on $\mathbf{p}_i^{(\mathrm{d})}$, we have

$$\begin{aligned}\left\{A(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\} &= \sum_{ij} \left\{a_i \delta_i, p_{j\beta}^{(\mathrm{d})} \delta_j'\right\} = \sum_{ij} a_i \delta_j' \left\{\delta_i, p_{j\beta}^{(\mathrm{d})}\right\} \\ &= -\sum_i a_i \delta(\mathbf{r}' - \mathbf{R}_i)\partial_\beta \delta(\mathbf{r} - \mathbf{R}_i) = -A(\mathbf{r}')\partial_\beta \delta(\mathbf{r} - \mathbf{r}'),\end{aligned} \tag{4.24}$$

where eq. (4.23) was used in the last step. In particular, the relation eq. (4.24) allows for the immediate calculation of the brackets

$$\left\{\rho^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\} = -\rho^{(\mathrm{d})}(\mathbf{r}')\partial_\beta \delta(\mathbf{r} - \mathbf{r}'), \tag{4.25}$$

$$\left\{k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\} = -k_\alpha^{(\mathrm{r})}(\mathbf{r}')\partial_\beta \delta(\mathbf{r} - \mathbf{r}'), \tag{4.26}$$

$$\left\{j_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\} = -j_\alpha^{(\mathrm{r})}(\mathbf{r}')\partial_\beta \delta(\mathbf{r} - \mathbf{r}'). \tag{4.27}$$

The remaining brackets are determined via similar, straightforward calculation:

$$\left\{j_\alpha^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\} = j_\beta^{(\mathrm{d})}(\mathbf{r})\partial_\alpha' \delta(\mathbf{r} - \mathbf{r}') - j_\alpha^{(\mathrm{d})}(\mathbf{r}')\partial_\beta \delta(\mathbf{r} - \mathbf{r}'), \tag{4.28}$$

$$\left\{k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{r})}(\mathbf{r}')\right\} = \delta_{\alpha\beta} \frac{k}{m^{(\mathrm{d})}} \rho^{(\mathrm{d})}(\mathbf{r}')\delta(\mathbf{r} - \mathbf{r}'). \tag{4.29}$$

All other Poisson brackets are simply zero; for more details see appendix A.3.1. We see that all Poisson brackets can be expressed in terms of the previously defined fields, and the set is closed. This is by no means guaranteed but depends on the initial choice of fields.

4.1.3 Conservative equations of motion

Now that Hamiltonian and Poisson brackets are known, all ingredients necessary to determine the conservative equations of motion are present. We recall from section 2.7 that the conservative part of dynamics can be written as

$$\left(\frac{\partial \phi_i(\mathbf{r}, t)}{\partial t}\right)_{\mathrm{cons}} = \sum_k \int \mathrm{d}^d\mathbf{r}' \left\{\phi_i(\mathbf{r}, t), \phi_k(\mathbf{r}', t)\right\} \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta \phi_k(\mathbf{r}', t)}. \tag{4.30}$$

First of all, we shall note that eq. (4.24) allows us to calculate, for an arbitrary function f, the integrals

$$\begin{aligned}\int \mathrm{d}^d\mathbf{r}' \left\{A(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\} f(\mathbf{r}') &= -\partial_\beta \int \mathrm{d}^d\mathbf{r}'\, A(\mathbf{r}')\delta(\mathbf{r} - \mathbf{r}') f(\mathbf{r}') \\ &= -\partial_\beta (A(\mathbf{r}) f(\mathbf{r}))\end{aligned} \tag{4.31}$$

and

$$\int \mathrm{d}^d\mathbf{r}' \left\{ j_\alpha^{(\mathrm{d})}(\mathbf{r}), A(\mathbf{r}') \right\} f(\mathbf{r}') = \int \mathrm{d}^d\mathbf{r}' \, A(\mathbf{r}) f(\mathbf{r}') \partial'_\alpha \delta(\mathbf{r} - \mathbf{r}') = -A(\mathbf{r}) \partial_\alpha f(\mathbf{r}). \tag{4.32}$$

We will also find useful the similar relations

$$\int \mathrm{d}^d\mathbf{r}' \left\{ j_\alpha^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}') \right\} f(\mathbf{r}') = -j_\beta^{(\mathrm{d})}(\mathbf{r}) \partial_\alpha f(\mathbf{r}) - \partial_\beta \Big(j_\alpha^{(\mathrm{d})}(\mathbf{r}) f(\mathbf{r}) \Big), \tag{4.33}$$

$$\int \mathrm{d}^d\mathbf{r}' \left\{ k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{r})}(\mathbf{r}') \right\} f(\mathbf{r}') = \delta_{\alpha\beta} \frac{k}{m^{(\mathrm{d})}} \rho^{(\mathrm{d})}(\mathbf{r}) f(\mathbf{r}), \tag{4.34}$$

$$\int \mathrm{d}^d\mathbf{r}' \left\{ j_\beta^{(\mathrm{r})}(\mathbf{r}), k_\alpha^{(\mathrm{r})}(\mathbf{r}') \right\} f(\mathbf{r}') = -\delta_{\alpha\beta} \frac{k}{m^{(\mathrm{d})}} \rho^{(\mathrm{d})}(\mathbf{r}) f(\mathbf{r}); \tag{4.35}$$

more details are again found in appendix A.3.1.

Densities

We proceed by deriving the equation of motion for the dumbbell density $\rho^{(\mathrm{d})}$ via the Poisson bracket expansion eq. (4.30). Using eqs. (4.16) and (4.31) leads to

$$\begin{aligned} \left(\frac{\partial \rho^{(\mathrm{d})}(\mathbf{r}, t)}{\partial t} \right)_{\mathrm{cons}} &= \sum_k \int \mathrm{d}^d\mathbf{r}' \left\{ \rho^{(\mathrm{d})}(\mathbf{r}), \phi_k(\mathbf{r}') \right\} \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta \phi_k(\mathbf{r}')} \\ &= \int \mathrm{d}^d\mathbf{r}' \left\{ \rho^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}(\mathbf{r}')} \\ &= -\partial_\beta \left(\rho^{(\mathrm{d})}(\mathbf{r}) \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}(\mathbf{r})} \right) = -\partial_\beta j_\beta^{(\mathrm{d})}(\mathbf{r}). \end{aligned} \tag{4.36}$$

This is just the continuity equation reflecting the conservation of dumbbell mass.

For the relative density $\mathbf{k}^{(\mathrm{r})}$ we obtain by using eqs. (4.31) and (4.34) the equation

$$\begin{aligned} \left(\frac{\partial k_\alpha^{(\mathrm{r})}(\mathbf{r})}{\partial t} \right)_{\mathrm{cons}} &= \int \mathrm{d}^d\mathbf{r}' \left[\left\{ k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{r})}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{r})}(\mathbf{r}')} \right. \\ &\quad \left. + \left\{ k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}(\mathbf{r}')} \right] \\ &= \delta_{\alpha\beta} \frac{k}{m^{(\mathrm{d})}} \rho^{(\mathrm{d})}(\mathbf{r}) \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{r})}(\mathbf{r})} - \partial_\beta \left(k_\alpha^{(\mathrm{r})} \frac{\delta \mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}} \right)(\mathbf{r}) \\ &= \frac{k}{m^{(\mathrm{r})}} j_\alpha^{(\mathrm{r})}(\mathbf{r}) - \partial_\beta \Big(k_\alpha^{(\mathrm{r})} v_\beta^{(\mathrm{d})} \Big)(\mathbf{r}). \end{aligned} \tag{4.37}$$

Currents

Next, the equations of motion for the currents are derived. For the dumbbell current $\mathbf{j}^{(\mathrm{d})}$ we apply eqs. (4.32) and (4.33) to calculate

$$\begin{aligned}
\left(\frac{\partial j_\alpha^{(\mathrm{d})}(\mathbf{r})}{\partial t}\right)_{\mathrm{cons}} &= \int \mathrm{d}^d\mathbf{r}' \left[\left\{j_\alpha^{(\mathrm{d})}(\mathbf{r}), \rho^{(\mathrm{d})}(\mathbf{r}')\right\}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\rho^{(\mathrm{d})}(\mathbf{r}')} + \left\{j_\alpha^{(\mathrm{d})}(\mathbf{r}), k_\beta^{(\mathrm{r})}(\mathbf{r}')\right\}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta k_\beta^{(\mathrm{r})}(\mathbf{r}')}\right. \\
&\quad \left. + \left\{j_\alpha^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{r})}(\mathbf{r}')\right\}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{r})}(\mathbf{r}')} + \left\{j_\alpha^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}(\mathbf{r}')}\right] \\
&= -\rho^{(\mathrm{d})}(\mathbf{r})\partial_\alpha\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\rho^{(\mathrm{d})}(\mathbf{r})} - k_\beta^{(\mathrm{r})}(\mathbf{r})\partial_\alpha\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta k_\beta^{(\mathrm{r})}(\mathbf{r})} - j_\beta^{(\mathrm{r})}(\mathbf{r})\partial_\alpha\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{r})}(\mathbf{r})} \\
&\quad - j_\beta^{(\mathrm{d})}(\mathbf{r})\partial_\alpha\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}(\mathbf{r})} - \partial_\alpha\left(j_\alpha^{(\mathrm{d})}(\mathbf{r})\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}(\mathbf{r})}\right) \\
&= -\rho^{(\mathrm{d})}(\mathbf{r})\partial_\alpha\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\rho^{(\mathrm{d})}(\mathbf{r})} - k_\beta^{(\mathrm{r})}(\mathbf{r})\partial_\alpha q_\beta(\mathbf{r}) - j_\beta^{(\mathrm{r})}(\mathbf{r})\partial_\alpha v_\beta^{(\mathrm{r})}(\mathbf{r}) \\
&\quad - j_\beta^{(\mathrm{d})}(\mathbf{r})\partial_\alpha v_\beta^{(\mathrm{d})}(\mathbf{r}) - \partial_\alpha\Big(j_\alpha^{(\mathrm{d})}(\mathbf{r})v_\beta^{(\mathrm{d})}(\mathbf{r})\Big).
\end{aligned} \tag{4.38}$$

In isothermal conditions, the partial pressure of the dumbbell component is given by

$$P^{(\mathrm{d})} = \left(\rho^{(\mathrm{d})}\right)^2\frac{\partial}{\partial\rho^{(\mathrm{d})}}\left(\frac{f^{(\mathrm{d})}}{\rho^{(\mathrm{d})}}\right) = \rho^{(\mathrm{d})}\frac{\partial f^{(\mathrm{d})}}{\partial\rho^{(\mathrm{d})}} - f^{(\mathrm{d})}, \tag{4.39}$$

and therefore

$$\boldsymbol{\nabla}P^{(\mathrm{d})} = \rho^{(\mathrm{d})}\boldsymbol{\nabla}\frac{\partial f^{(\mathrm{d})}}{\partial\rho^{(\mathrm{d})}}. \tag{4.40}$$

This can for example be seen from eq. (A.27) with $\rho^{(2)} = 0$. Then, we can write

$$\begin{aligned}
&-\rho^{(\mathrm{d})}\partial_\alpha\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta\rho^{(\mathrm{d})}} \\
&\quad = -\partial_\alpha P^{(\mathrm{d})} + \frac{\rho^{(\mathrm{d})}}{2}\partial_\alpha\left[\left(\mathbf{v}^{(\mathrm{d})}\right)^2 + \Gamma\left(\mathbf{v}^{(\mathrm{r})}\right)^2 + \frac{k}{m^{(\mathrm{d})}}\mathbf{q}^2\right] + \kappa\rho^{(\mathrm{d})}\partial_\alpha\nabla^2\rho^{(\mathrm{d})} \\
&\quad = -\partial_\alpha P^{(\mathrm{d})} + \left[\mathbf{j}^{(\mathrm{d})}\partial_\alpha\mathbf{v}^{(\mathrm{d})} + \mathbf{j}^{(\mathrm{r})}\partial_\alpha\mathbf{v}^{(\mathrm{r})} + \mathbf{k}^{(\mathrm{r})}\partial_\alpha\mathbf{q}\right] + \kappa\rho^{(\mathrm{d})}\partial_\alpha\nabla^2\rho^{(\mathrm{d})}.
\end{aligned} \tag{4.41}$$

Inserting this in eq. (4.38) and canceling complementary terms finally results in the dynamic equation for the dumbbell current

$$\left(\frac{\partial j_\alpha^{(\mathrm{d})}}{\partial t}\right)_{\mathrm{cons}} = -\partial_\beta\left(j_\alpha^{(\mathrm{d})}v_\beta^{(\mathrm{d})}\right) - \partial_\alpha P^{(\mathrm{d})} + \kappa\rho^{(\mathrm{d})}\partial_\alpha\nabla^2\rho^{(\mathrm{d})}. \tag{4.42}$$

This can be interpreted as an Euler equation for the dumbbell component with a surface term as driving force.

To calculate the equation of motion for the relative current $\mathbf{j}^{(\mathrm{r})}$, we use eqs. (4.31) and (4.35) to find

$$\begin{aligned}
&\left(\frac{\partial j_\alpha^{(\mathrm{r})}(\mathbf{r})}{\partial t}\right)_{\mathrm{cons}} \\
&\quad= \int \mathrm{d}^d\mathbf{r}' \left[\left\{j_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\right\}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}(\mathbf{r}')} + \left\{j_\alpha^{(\mathrm{r})}(\mathbf{r}), k_\beta^{(\mathrm{r})}(\mathbf{r}')\right\}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta k_\beta^{(\mathrm{r})}(\mathbf{r}')}\right] \\
&\quad= -\partial_\beta\left(j_\alpha^{(\mathrm{r})}\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta j_\beta^{(\mathrm{d})}}\right)(\mathbf{r}) - \delta_{\alpha\beta}\frac{k}{m^{(\mathrm{d})}}\rho^{(\mathrm{d})}(\mathbf{r})\frac{\delta\mathscr{H}^{(\mathrm{d})}}{\delta k_\beta^{(\mathrm{r})}(\mathbf{r})} \\
&\quad= -\partial_\beta\left(j_\alpha^{(\mathrm{r})}v_\beta^{(\mathrm{d})}\right)(\mathbf{r}) - k_\alpha^{(\mathrm{r})}(\mathbf{r}).
\end{aligned} \tag{4.43}$$

In summary, the conservative equations of motion are

$$\left(\frac{\partial\rho^{(\mathrm{d})}}{\partial t}\right)_{\mathrm{cons}} = -\partial_\beta j_\beta^{(\mathrm{d})}, \tag{4.44}$$

$$\left(\frac{\partial j_\alpha^{(\mathrm{d})}}{\partial t}\right)_{\mathrm{cons}} = -\partial_\beta\left(j_\alpha^{(\mathrm{d})}v_\beta^{(\mathrm{d})}\right) - \partial_\alpha P^{(\mathrm{d})} + \kappa\rho^{(\mathrm{d})}\partial_\alpha\nabla^2\rho^{(\mathrm{d})}, \tag{4.45}$$

$$\left(\frac{\partial k_\alpha^{(\mathrm{r})}}{\partial t}\right)_{\mathrm{cons}} = -\partial_\beta\left(k_\alpha^{(\mathrm{r})}v_\beta^{(\mathrm{d})}\right) + \frac{k}{m^{(\mathrm{r})}}j_\alpha^{(\mathrm{r})}, \tag{4.46}$$

$$\left(\frac{\partial j_\alpha^{(\mathrm{r})}}{\partial t}\right)_{\mathrm{cons}} = -\partial_\beta\left(j_\alpha^{(\mathrm{r})}v_\beta^{(\mathrm{d})}\right) - k_\alpha^{(\mathrm{r})}, \tag{4.47}$$

$$\left(\frac{\partial\rho^{(\mathrm{s})}}{\partial t}\right)_{\mathrm{cons}} = -\partial_\beta j_\beta^{(\mathrm{s})}, \tag{4.48}$$

$$\left(\frac{\partial j_\alpha^{(\mathrm{s})}}{\partial t}\right)_{\mathrm{cons}} = -\partial_\beta\left(j_\alpha^{(\mathrm{s})}v_\beta^{(\mathrm{s})}\right) - \partial_\beta P^{(\mathrm{s})}. \tag{4.49}$$

4.1.4 Dissipative coupling

In analogy to the LB/MD coupling (c.f. section 3.2.4), we imagine the dumbbell molecules to be in a continuous solvent background to which they are coupled exclusively via friction forces. These forces are proportional to the relative velocity between solvent and polymers with a Stokes friction coefficient ζ_{c}. When determining the dissipation rate, the Hamiltonian part of the dynamics will not enter and can be omitted right away. The dissipative part of the dynamics of a single particle i with mass m and velocity $\mathbf{v}_i$ inside a

solvent with velocity field $\mathbf{v}^{(\mathrm{s})}(\mathbf{r})$ can be expressed by

$$\left(\frac{\partial \mathbf{v}_i}{\partial t}\right)_{\mathrm{diss}} = -\frac{\zeta_{\mathrm{c}}}{m}\Big(\mathbf{v}_i - \Big\langle \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_i)\Big\rangle_\omega\Big) = -\frac{1}{\tau}\Big(\mathbf{v}_i - \Big\langle \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_i)\Big\rangle_\omega\Big), \tag{4.50}$$

where the friction time scale $\tau = m/\zeta_{\mathrm{c}}$ was introduced. The solvent velocity at the particle position is calculated via averaging the solvent velocity around the particle position $\mathbf{r}_i$ wrt. some interpolation kernel ω:

$$\Big\langle \mathbf{v}^{(\mathrm{s})}(\mathbf{r}_i)\Big\rangle_\omega = \int \mathrm{d}^d\mathbf{r}\, \omega(\mathbf{r}-\mathbf{r}_i)\mathbf{v}^{(\mathrm{s})}(\mathbf{r}) = \int \mathrm{d}^d\mathbf{r}'\, \omega(\mathbf{r}')\mathbf{v}^{(\mathrm{s})}(\mathbf{r}'+\mathbf{r}_i). \tag{4.51}$$

We require that ω is isotropic and is normalized, i.e.

$$\omega(\mathbf{r}) = \omega(|\mathbf{r}|), \qquad\qquad \int \mathrm{d}^d\mathbf{r}\, \omega(\mathbf{r}) = 1. \tag{4.52}$$

In the vicinity of the center of mass $\mathbf{R}$ of a dumbbell, the solvent velocity can be expressed by the Taylor expansion

$$\begin{aligned} v_\alpha^{(\mathrm{s})}(\mathbf{r}) &= v_\alpha^{(\mathrm{s})}(\mathbf{R}) + (r_\beta - R_\beta)\partial_\beta v_\alpha^{(\mathrm{s})}(\mathbf{R}) + \frac{1}{2}(r_\beta - R_\beta)(r_\gamma - R_\gamma)\partial_\beta\partial_\gamma v_\alpha^{(\mathrm{s})}(\mathbf{R}) \\ &\quad + \frac{1}{6}(r_\beta - R_\beta)(r_\gamma - R_\gamma)(r_\delta - R_\delta)\partial_\beta\partial_\gamma\partial_\delta v_\alpha^{(\mathrm{s})}(\mathbf{R}) \\ &\quad + \frac{1}{24}(r_\beta - R_\beta)(r_\gamma - R_\gamma)(r_\delta - R_\delta)(r_\epsilon - R_\epsilon)\partial_\beta\partial_\gamma\partial_\delta\partial_\epsilon v_\alpha^{(\mathrm{s})}(\mathbf{R}) \\ &\quad + \dots \\ &=: \Big(1 + \Delta^{(1)} + \frac{1}{2}\Delta^{(2)} + \frac{1}{6}\Delta^{(3)} + \frac{1}{24}\Delta^{(4)} + \dots\Big) v_\alpha^{(\mathrm{s})}(\mathbf{R}). \end{aligned} \tag{4.53}$$

In order to determine the dissipative dynamics for the center of mass and relative component of the dumbbells, it is helpful to know the sum and difference of solvent velocities at the position of the dumbbell ends. Since $\mathbf{r}^{(1)} = \mathbf{R} + \mathbf{q}/2$ and $\mathbf{r}^{(2)} = \mathbf{R} - \mathbf{q}/2$, the estimate of the sum of velocities is

$$\begin{aligned} &\Big\langle \mathbf{v}^{(\mathrm{s})}(\mathbf{r}^{(1)}) + \mathbf{v}^{(\mathrm{s})}(\mathbf{r}^{(2)})\Big\rangle_\omega \\ &\quad = \int \mathrm{d}^d\mathbf{r}'\, \omega(\mathbf{r}')\Big(\mathbf{v}^{(\mathrm{s})}(\mathbf{r}'+\mathbf{R}+\mathbf{q}/2) + \mathbf{v}^{(\mathrm{s})}(\mathbf{r}'+\mathbf{R}-\mathbf{q}/2)\Big) \\ &\quad = \int \mathrm{d}^d\mathbf{r}'\, \omega(\mathbf{r}')\Big(2 + \Delta_+^{(1)} + \frac{1}{2}\Delta_+^{(2)} + \frac{1}{6}\Delta_+^{(3)} + \frac{1}{24}\Delta_+^{(4)} + \dots\Big)\mathbf{v}^{(\mathrm{s})}(\mathbf{R}). \end{aligned} \tag{4.54}$$

Here, the shorthand notation

$$\begin{aligned}
\Delta_+^{(0)}(\mathbf{r}') &= 2,\\
\Delta_+^{(1)}(\mathbf{r}') &= (r'_\alpha + q_\alpha/2 + r'_\alpha - q_\alpha/2)\partial_\alpha = 2r'_\alpha\partial_\alpha,\\
\Delta_+^{(2)}(\mathbf{r}') &= \Big[(r'_\alpha + q_\alpha/2)\Big(r'_\beta + q_\beta/2\Big) + (r'_\alpha - q_\alpha/2)\Big(r'_\beta - q_\beta/2\Big)\Big]\partial_\alpha\partial_\beta,\\
&= \Big[2r'_\alpha r'_\beta + \frac{1}{2}q_\alpha q_\beta\Big]\partial_\alpha\partial_\beta,\\
&\;\;\vdots
\end{aligned} \tag{4.55}$$

was introduced.

Because of the isotropy of ω, all odd moments of the kernel vanish and we can write

$$\begin{aligned}
&\Big\langle \mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}_i^{(1)}\Big) + \mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}_i^{(2)}\Big)\Big\rangle_\omega\\
&\quad = \int \mathrm{d}^d\mathbf{r}'\,\omega(\mathbf{r}')\Big(2 + 2\mathbf{r}'\cdot\boldsymbol{\nabla} + \Big[\mathbf{r}'\mathbf{r}' + \frac{1}{4}\mathbf{q}_i\mathbf{q}_i\Big] : \boldsymbol{\nabla}\boldsymbol{\nabla} + \dots\Big)\,\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\\
&= 2\Big(1 + \frac{1}{2}\Big[\mathbf{M}^{(2)} + \frac{1}{4}\mathbf{q}_i\mathbf{q}_i\Big] : \boldsymbol{\nabla}\boldsymbol{\nabla} + \dots\Big)\,\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\\
&=: 2\Omega_+(\mathbf{q}_i)\,\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i},
\end{aligned} \tag{4.56}$$

where we have introduced the kernel moments for $n \geq 1$

$$M^{(n)}_{\alpha_1\alpha_2\dots\alpha_n} = \int \mathrm{d}^d\mathbf{r}\,\omega(\mathbf{r}) r_{\alpha_1} r_{\alpha_2}\dots r_{\alpha_n}. \tag{4.57}$$

Note that both sides of eq. (4.56) are even under a flip transformation $\mathbf{q} \to -\mathbf{q}$.

The averaged difference in velocities then is

$$\begin{aligned}
&\Big\langle \mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}^{(1)}\Big) - \mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}^{(2)}\Big)\Big\rangle_\omega\\
&\quad = \int \mathrm{d}^d\mathbf{r}'\,\omega(\mathbf{r}')\Big(\Delta_-^{(1)} + \frac{1}{2}\Delta_-^{(2)} + \frac{1}{6}\Delta_-^{(3)} + \frac{1}{24}\Delta_-^{(4)} + \dots\Big)\,\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i},
\end{aligned} \tag{4.58}$$

where

$$\begin{aligned}
\Delta_-^{(0)} &= 0,\\
\Delta_-^{(1)} &= (r'_\alpha + q_\alpha/2 - r'_\alpha + q_\alpha/2)\partial_\alpha = q_\alpha\partial_\alpha,\\
\Delta_-^{(2)} &= \Big[(r'_\alpha + q_\alpha/2)\Big(r'_\beta + q_\beta/2\Big) - (r'_\alpha - q_\alpha/2)\Big(r'_\beta - q_\beta/2\Big)\Big]\partial_\alpha\partial_\beta\\
&= \Big[r'_\alpha q_\beta + q_\alpha r'_\beta\Big]\partial_\alpha\partial_\beta,\\
\Delta_-^{(3)} &= \Big[(r'_\alpha + q_\alpha/2)\Big(r'_\beta + q_\beta/2\Big)\Big(r'_\gamma + q_\gamma/2\Big)\\
&\quad - (r'_\alpha - q_\alpha/2)\Big(r'_\beta - q_\beta/2\Big)\Big(r'_\gamma - q_\gamma/2\Big)\Big]\partial_\alpha\partial_\beta\partial_\gamma\\
&= \Big[\frac{q_\alpha q_\beta q_\gamma}{4} + q_\gamma r'_\alpha r'_\beta + q_\beta r'_\alpha r'_\gamma + q_\alpha r'_\beta r'_\gamma\Big]\partial_\alpha\partial_\beta\partial_\gamma,\\
&\ \ \vdots
\end{aligned} \tag{4.59}$$

We can write

$$\begin{aligned}
&\Big\langle \mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}_i^{(1)}\Big) - \mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}_i^{(2)}\Big)\Big\rangle_\omega\\
&= \int \mathrm{d}^d\mathbf{r}'\,\omega(\mathbf{r}')\Big(q_{i\alpha}\partial_{i\alpha} + \frac{1}{2}\Big[r'_\alpha q_{i\beta} + q_{i\alpha}r'_{i\beta}\Big]\partial_\alpha\partial_\beta\\
&\quad + \frac{1}{6}\Big[\frac{q_{i\alpha}q_{i\beta}q_{i\gamma}}{4} + q_{i\gamma}r'_\alpha r'_\beta + q_{i\beta}r'_\alpha r'_\gamma + q_{i\alpha}r'_\beta r'_\gamma\Big]\partial_\alpha\partial_\beta\partial_\gamma + \dots\Big)\,\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\\
&= \Big(q_{i\alpha}\partial_\alpha + \frac{1}{6}\Big[\frac{q_{i\alpha}q_{i\beta}q_{i\gamma}}{4} + M^{(2)}_{\alpha\beta}q_{i\gamma} + M^{(2)}_{\alpha\gamma}q_{i\beta} + M^{(2)}_{\beta\gamma}q_{i\alpha}\Big]\partial_\alpha\partial_\beta\partial_\gamma + \dots\Big)\,\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\\
&=: \Omega_-(\mathbf{q}_i)\,\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}.
\end{aligned} \tag{4.60}$$

Both sides of eq. (4.60) are odd wrt. flip. The simplest form of the interpolation kernel is just the Dirac delta function $\omega(\mathbf{r}) = \delta(\mathbf{r})$, then all moments $\mathbf{M}^{(n)} = 0$ and the interpolation operators become

$$\Omega_+(\mathbf{q}) = 1 + \frac{1}{8}\mathbf{qq} : \boldsymbol{\nabla\nabla} + \dots, \tag{4.61}$$

$$\Omega_-(\mathbf{q}) = \mathbf{q}\cdot\boldsymbol{\nabla} + \frac{1}{24}\mathbf{qqq} \vdots \boldsymbol{\nabla\nabla\nabla} + \dots \tag{4.62}$$

Dumbbells

The Ω operators can now be used to express the dissipation rate of the dumbbell component. It results from a Stokes friction that is acting on both dumbbell ends located at $\mathbf{r}_i^{(1,2)} =$

$\mathbf{R}_i \pm \mathbf{q}_i/2$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\dot{\mathbf{R}}_i + \frac{\dot{\mathbf{q}}_i}{2}\right)_{\mathrm{diss}} = -\frac{1}{\tau}\left(\dot{\mathbf{R}}_i + \frac{\dot{\mathbf{q}}_i}{2} - \left\langle \mathbf{v}^{(\mathrm{s})}\left(\mathbf{r}_i^{(1)}\right)\right\rangle_\omega\right), \tag{4.63}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\dot{\mathbf{R}}_i - \frac{\dot{\mathbf{q}}_i}{2}\right)_{\mathrm{diss}} = -\frac{1}{\tau}\left(\dot{\mathbf{R}}_i - \frac{\dot{\mathbf{q}}_i}{2} - \left\langle \mathbf{v}^{(\mathrm{s})}\left(\mathbf{r}_i^{(2)}\right)\right\rangle_\omega\right). \tag{4.64}$$

Expressed in terms of the conjugate momenta, this becomes

$$\begin{aligned}\dot{\mathbf{p}}_i^{(\mathrm{d})} &= m^{(\mathrm{d})}\dot{\mathbf{R}}_i = -\frac{m^{(\mathrm{d})}}{2\tau}\left[2\dot{\mathbf{R}}_i - \left\langle \mathbf{v}^{(\mathrm{s})}\left(\mathbf{r}_i^{(1)}\right) + \mathbf{v}^{(\mathrm{s})}\left(\mathbf{r}_i^{(2)}\right)\right\rangle_\omega\right] \\ &= -\frac{m^{(\mathrm{d})}}{\tau}\left[\dot{\mathbf{R}}_i - \Omega_+(\mathbf{q}_i)\ \mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\right] =: -\frac{m^{(\mathrm{d})}}{\tau}\mathbf{u}_i^{(0)},\end{aligned} \tag{4.65}$$

$$\begin{aligned}\dot{\mathbf{p}}_i^{(\mathrm{r})} &= m^{(\mathrm{r})}\dot{\mathbf{q}}_i = -\Gamma\frac{m^{(\mathrm{d})}}{\tau}\left[\dot{\mathbf{q}}_i - \left\langle \mathbf{v}^{(\mathrm{s})}\left(\mathbf{r}_i^{(1)}\right) - \mathbf{v}^{(\mathrm{s})}\left(\mathbf{r}_i^{(2)}\right)\right\rangle_\omega\right] \\ &= -\Gamma\frac{m^{(\mathrm{d})}}{\tau}\left[\dot{\mathbf{q}}_i - \Omega_-(\mathbf{q}_i)\ \mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\right] =: -\Gamma\frac{m^{(\mathrm{d})}}{\tau}\Delta\mathbf{u}_i,\end{aligned} \tag{4.66}$$

where we have defined

$$\mathbf{u}_i^{(0)} = \dot{\mathbf{R}}_i - \Omega_+(\mathbf{q}_i)\ \mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}, \tag{4.67}$$

$$\Delta\mathbf{u}_i = \dot{\mathbf{q}}_i - \Omega_-(\mathbf{q}_i)\ \mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}. \tag{4.68}$$

The total amount of energy dissipated by all dumbbells per unit of time is then given by the time derivative of the microscopic dumbbell Hamiltonian:

$$\begin{aligned}\frac{\mathrm{d}\widehat{\mathscr{H}}^{(\mathrm{d})}}{\mathrm{d}t} &= \sum_i\left[\dot{\mathbf{R}}_i\cdot\dot{\mathbf{p}}_i^{(\mathrm{d})} + \dot{\mathbf{q}}_i\cdot\dot{\mathbf{p}}_i^{(\mathrm{r})}\right] = -\frac{m^{(\mathrm{d})}}{\tau}\sum_i\left[\dot{\mathbf{R}}_i\cdot\mathbf{u}_i^{(0)} + \Gamma\dot{\mathbf{q}}_i\cdot\Delta\mathbf{u}_i\right] \\ &= -\frac{m^{(\mathrm{d})}}{\tau}\sum_i\left[\left(\mathbf{u}_i^{(0)} + \Omega_+\ \mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\right)\cdot\mathbf{u}_i^{(0)} + \Gamma\left(\Delta\mathbf{u}_i + \Omega_-\ \mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\right)\cdot\Delta\mathbf{u}_i\right].\end{aligned} \tag{4.69}$$

Solvent

In the next step, the dissipation rate of the solvent is determined. The viscous dissipation term $\propto \eta_{\alpha\beta\gamma\delta}$ in the solvent dynamics is well known to comply with non-equilibrium thermodynamics and can be omitted. It is thus enough to consider

$$\left(\frac{\partial\rho^{(\mathrm{s})}}{\partial t}\right)_{\mathrm{diss}} = 0, \tag{4.70}$$

$$\left(\frac{\partial\mathbf{j}^{(\mathrm{s})}}{\partial t}\right)_{\mathrm{diss}} = \mathbf{g}, \tag{4.71}$$

where we express the force density $\mathbf{g}$ by point forces exerted by the beads,

$$\begin{aligned}\mathbf{g}(\mathbf{r}) &= -\sum_i \left[\delta\Big(\mathbf{r}-\mathbf{r}_i^{(1)}\Big)\dot{\mathbf{p}}_i^{(1)} + \delta\Big(\mathbf{r}-\mathbf{r}_i^{(2)}\Big)\dot{\mathbf{p}}_i^{(2)}\right] \\ &= \frac{m}{\tau}\sum_i \left[\delta\Big(\mathbf{r}-\mathbf{r}_i^{(1)}\Big)\mathbf{u}_i^{(1)} + \delta\Big(\mathbf{r}-\mathbf{r}_i^{(2)}\Big)\mathbf{u}_i^{(2)}\right] \\ &= \frac{m}{\tau}\sum_i \left[\delta\Big(\mathbf{r}-\mathbf{r}_i^{(1)}\Big)\Big(\mathbf{u}_i^{(0)}+\frac{1}{2}\Delta\mathbf{u}_i\Big) + \delta\Big(\mathbf{r}-\mathbf{r}_i^{(2)}\Big)\Big(\mathbf{u}_i^{(0)}-\frac{1}{2}\Delta\mathbf{u}_i\Big)\right].\end{aligned} \tag{4.72}$$

With the chain rule for functional derivatives eq. (A.11), this corresponds to a dissipation rate of

$$\begin{aligned}\frac{\mathrm{d}\mathscr{H}^{(\mathrm{s})}}{\mathrm{d}t} &= \int \mathrm{d}^d\mathbf{r}\,\frac{\delta\mathscr{H}^{(\mathrm{s})}}{\delta\mathbf{j}^{(\mathrm{s})}(\mathbf{r})}\cdot\partial_t\mathbf{j}^{(\mathrm{s})}(\mathbf{r}) = \int \mathrm{d}^d\mathbf{r}\,\mathbf{v}^{(\mathrm{s})}(\mathbf{r})\cdot\mathbf{g}(\mathbf{r}) \\ &= \frac{m}{\tau}\sum_i \left[\mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}_i^{(1)}\Big)\cdot\Big(\mathbf{u}_i^{(0)}+\frac{1}{2}\Delta\mathbf{u}_i\Big) + \mathbf{v}^{(\mathrm{s})}\Big(\mathbf{r}_i^{(2)}\Big)\cdot\Big(\mathbf{u}_i^{(0)}-\frac{1}{2}\Delta\mathbf{u}_i\Big)\right] \\ &= \frac{m^{(\mathrm{d})}}{\tau}\sum_i \left[\mathbf{u}_i^{(0)}\cdot\Omega_+\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i} + \Gamma\Delta\mathbf{u}_i\cdot\Omega_-\mathbf{v}^{(\mathrm{s})}\Big|_{\mathbf{R}_i}\right].\end{aligned} \tag{4.73}$$

Adding eqs. (4.69) and (4.73) leads to a total dissipation rate of

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\widehat{\mathscr{H}}^{(\mathrm{d})}+\mathscr{H}^{(\mathrm{s})}\Big) = -\frac{m^{(\mathrm{d})}}{\tau}\sum_i\left[\Big(\mathbf{u}_i^{(0)}\Big)^2+\Gamma\Delta\mathbf{u}_i^2\right]. \tag{4.74}$$

A suitable continuum limit of the microscopic dissipation rate is

$$\frac{\mathrm{d}\mathscr{H}}{\mathrm{d}t} = -\int \mathrm{d}^d\mathbf{r}\,\frac{\rho^{(\mathrm{d})}}{\tau}\left[\Big(\mathbf{u}^{(0)}(\mathbf{r})\Big)^2+\Gamma\Delta\mathbf{u}(\mathbf{r})^2\right], \tag{4.75}$$

where

$$\mathbf{u}^{(0)}(\mathbf{r}) = \mathbf{v}^{(\mathrm{d})}(\mathbf{r}) - \Omega_+(\mathbf{q}(\mathbf{r}))\mathbf{v}^{(\mathrm{s})}(\mathbf{r}), \tag{4.76}$$

$$\Delta\mathbf{u}(\mathbf{r}) = \mathbf{v}^{(\mathrm{r})}(\mathbf{r}) - \Omega_-(\mathbf{q}(\mathbf{r}))\mathbf{v}^{(\mathrm{s})}(\mathbf{r}). \tag{4.77}$$

are the continuum analogues of eqs. (4.67) and (4.68).

By inserting eqs. (4.76) and (4.77) into eq. (4.75) the continuum dissipation rate is expressed in terms of the velocities

$$\begin{aligned}\tau\frac{\mathrm{d}\mathscr{H}}{\mathrm{d}t} = -\int \mathrm{d}^d\mathbf{r}\,\rho^{(\mathrm{d})}\Big[&\Big(\mathbf{v}^{(\mathrm{d})}\Big)^2 - 2\mathbf{v}^{(\mathrm{d})}\cdot\Omega_+(\mathbf{q})\mathbf{v}^{(\mathrm{s})} + \Big(\Omega_+(\mathbf{q})\mathbf{v}^{(\mathrm{s})}\Big)^2 \\ &+\Gamma\Big(\mathbf{v}^{(\mathrm{r})}\Big)^2 - 2\Gamma\mathbf{v}^{(\mathrm{r})}\cdot\Omega_-(\mathbf{q})\mathbf{v}^{(\mathrm{s})} + \Gamma\Big(\Omega_-(\mathbf{q})\mathbf{v}^{(\mathrm{s})}\Big)^2\Big].\end{aligned} \tag{4.78}$$

We now introduce adjoint operators $\Omega_+^\dagger$ and $\Omega_-^\dagger$ in the sense that with two functions f and g

$$\int \mathrm{d}^d\mathbf{r}\, f(\mathbf{r})\Omega_\pm(\mathbf{q}(\mathbf{r}))g(\mathbf{r}) = \int \mathrm{d}^d\mathbf{r}\, g(\mathbf{r})\Omega_\pm^\dagger(\mathbf{q}(\mathbf{r}))f(\mathbf{r}). \tag{4.79}$$

It is easily shown via multiple integration by parts that this is satisfied with

$$\Omega_+^\dagger = 1 + \frac{1}{8}\boldsymbol{\nabla}\boldsymbol{\nabla} : \mathbf{q}\mathbf{q} + \dots \tag{4.80}$$

$$\Omega_-^\dagger = -\boldsymbol{\nabla}\cdot\mathbf{q} - \frac{1}{24}\boldsymbol{\nabla}\boldsymbol{\nabla}\boldsymbol{\nabla}\vdots\mathbf{q}\mathbf{q}\mathbf{q} - \dots \tag{4.81}$$

The total dissipation rate thus takes the form

$$\begin{aligned}
&-\tau\frac{\mathrm{d}\mathscr{H}}{\mathrm{d}t}\\
&= \int \mathrm{d}^d\mathbf{r} \begin{pmatrix}\mathbf{v}^{(\mathrm{d})}(\mathbf{r})\\ \mathbf{v}^{(\mathrm{r})}(\mathbf{r})\\ \mathbf{v}^{(\mathrm{s})}(\mathbf{r})\end{pmatrix} \cdot \begin{pmatrix}\rho^{(\mathrm{d})}\mathbf{v}^{(\mathrm{d})} - 2\rho^{(\mathrm{d})}\Omega_+\mathbf{v}^{(\mathrm{s})}\\ \Gamma\rho^{(\mathrm{d})}\mathbf{v}^{(\mathrm{r})} - 2\Gamma\rho^{(\mathrm{d})}\Omega_-\mathbf{v}^{(\mathrm{s})}\\ \Omega_+^\dagger\left(\rho^{(\mathrm{d})}\Omega_+\mathbf{v}^{(\mathrm{s})}\right) + \Gamma\Omega_-^\dagger\left(\rho^{(\mathrm{d})}\Omega_-\mathbf{v}^{(\mathrm{s})}\right)\end{pmatrix}\\
&= \int \mathrm{d}^d\mathbf{r} \begin{pmatrix}\mathbf{v}^{(\mathrm{d})}(\mathbf{r})\\ \mathbf{v}^{(\mathrm{r})}(\mathbf{r})\\ \mathbf{v}^{(\mathrm{s})}(\mathbf{r})\end{pmatrix} \cdot \begin{pmatrix}\rho^{(\mathrm{d})}\mathbf{v}^{(\mathrm{d})} - \rho^{(\mathrm{d})}\Omega_+\mathbf{v}^{(\mathrm{s})}\\ \Gamma\rho^{(\mathrm{d})}\mathbf{v}^{(\mathrm{r})} - \Gamma\rho^{(\mathrm{d})}\Omega_-\mathbf{v}^{(\mathrm{s})}\\ \Omega_+^\dagger\rho^{(\mathrm{d})}\left(\Omega_+\mathbf{v}^{(\mathrm{s})} - \mathbf{v}^{(\mathrm{d})}\right) + \Gamma\Omega_-^\dagger\rho^{(\mathrm{d})}\left(\Omega_-\mathbf{v}^{(\mathrm{s})} - \mathbf{v}^{(\mathrm{r})}\right)\end{pmatrix}\\
&= \int \mathrm{d}^d\mathbf{r} \begin{pmatrix}\mathbf{v}^{(\mathrm{d})}(\mathbf{r})\\ \mathbf{v}^{(\mathrm{r})}(\mathbf{r})\\ \mathbf{v}^{(\mathrm{s})}(\mathbf{r})\end{pmatrix} \cdot \begin{pmatrix}\rho^{(\mathrm{d})}\left(\mathbf{v}^{(\mathrm{d})} - \Omega_+\mathbf{v}^{(\mathrm{s})}\right)\\ \Gamma\rho^{(\mathrm{d})}\left(\mathbf{v}^{(\mathrm{r})} - \Omega_-\mathbf{v}^{(\mathrm{s})}\right)\\ -\Omega_+^\dagger\rho^{(\mathrm{d})}\left(\mathbf{v}^{(\mathrm{d})} - \Omega_+\mathbf{v}^{(\mathrm{s})}\right) - \Gamma\Omega_-^\dagger\rho^{(\mathrm{d})}\left(\mathbf{v}^{(\mathrm{r})} - \Omega_-\mathbf{v}^{(\mathrm{s})}\right)\end{pmatrix}\\
&= \tau\sum_{ij}\int \mathrm{d}^d\mathbf{r}\,\mathrm{d}^d\mathbf{r}'\,\frac{\delta\mathscr{H}}{\delta\phi_i(\mathbf{r})}M_{ij}(\mathbf{r},\mathbf{r}')\frac{\delta\mathscr{H}}{\delta\phi_j(\mathbf{r}')},
\end{aligned} \tag{4.82}$$

where we compare to the GENERIC dissipation rate of the free energy eq. (2.202) in the last step. From section 2.8 we know that the dissipative part of the dynamics is given by

$$\tau\left(\frac{\partial\phi_i(\mathbf{r})}{\partial t}\right)_{\mathrm{diss}} = -\tau\sum_j\int \mathrm{d}^d\mathbf{r}'\, M_{ij}(\mathbf{r},\mathbf{r}')\frac{\delta\mathscr{H}}{\delta\phi_j(\mathbf{r}')}. \tag{4.83}$$

Recalling the functional derivatives of the Hamiltonian

$$\frac{\delta\mathscr{H}}{\delta\mathbf{j}^{(\mathrm{d})}} = \mathbf{v}^{(\mathrm{d})}, \qquad \frac{\delta\mathscr{H}}{\delta\mathbf{j}^{(\mathrm{r})}} = \mathbf{v}^{(\mathrm{r})}, \qquad \frac{\delta\mathscr{H}}{\delta\mathbf{j}^{(\mathrm{s})}} = \mathbf{v}^{(\mathrm{s})}, \tag{4.84}$$

we can read the dissipative equations of motion directly from eq. (4.82):

$$\tau\left(\frac{\partial \mathbf{j}^{(\mathrm{d})}}{\partial t}\right)_{\mathrm{diss}} = -\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{d})} - \Omega_+\mathbf{v}^{(\mathrm{s})}\Big), \tag{4.85}$$

$$\tau\left(\frac{\partial \mathbf{j}^{(\mathrm{r})}}{\partial t}\right)_{\mathrm{diss}} = -\Gamma\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{r})} - \Omega_-\mathbf{v}^{(\mathrm{s})}\Big), \tag{4.86}$$

$$\tau\left(\frac{\partial \mathbf{j}^{(\mathrm{s})}}{\partial t}\right)_{\mathrm{diss}} = \Omega_+^\dagger\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{d})} - \Omega_+\mathbf{v}^{(\mathrm{s})}\Big) + \Gamma\Omega_-^\dagger\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{r})} - \Omega_-\mathbf{v}^{(\mathrm{s})}\Big). \tag{4.87}$$

Equations (4.85) to (4.87) together with the conservative part eqs. (4.44) to (4.49) finally result in the full equations of motion

$$\frac{\partial \rho^{(\mathrm{d})}}{\partial t} = -\partial_\beta j_\beta^{(\mathrm{d})}, \tag{4.88}$$

$$\begin{aligned}\frac{\partial j_\alpha^{(\mathrm{d})}}{\partial t} &= -\partial_\alpha\Big(j_\alpha^{(\mathrm{d})}v_\beta^{(\mathrm{d})}\Big) - \partial_\alpha P^{(\mathrm{d})} + \kappa\rho^{(\mathrm{d})}\partial_\alpha\nabla^2\rho^{(\mathrm{d})} \\ &\quad - \frac{1}{\tau}\rho^{(\mathrm{d})}\Big(v_\alpha^{(\mathrm{d})} - \Omega_+ v_\alpha^{(\mathrm{s})}\Big),\end{aligned} \tag{4.89}$$

$$\frac{\partial k_\alpha^{(\mathrm{r})}}{\partial t} = -\partial_\beta\Big(k_\alpha^{(\mathrm{r})}v_\beta^{(\mathrm{d})}\Big) + \frac{k}{m^{(\mathrm{r})}}j_\alpha^{(\mathrm{r})}, \tag{4.90}$$

$$\frac{\partial j_\alpha^{(\mathrm{r})}}{\partial t} = -\partial_\beta\Big(j_\alpha^{(\mathrm{r})}v_\beta^{(\mathrm{d})}\Big) - k_\alpha^{(\mathrm{r})} - \frac{\Gamma}{\tau}\rho^{(\mathrm{d})}\Big(v_\alpha^{(\mathrm{r})} - \Omega_- v_\alpha^{(\mathrm{s})}\Big), \tag{4.91}$$

$$\frac{\partial \rho^{(\mathrm{s})}}{\partial t} = -\partial_\beta j_\beta^{(\mathrm{s})}, \tag{4.92}$$

$$\begin{aligned}\frac{\partial j_\alpha^{(\mathrm{s})}}{\partial t} &= -\partial_\beta\Big(j_\alpha^{(\mathrm{s})}v_\beta^{(\mathrm{s})}\Big) - \partial_\beta P^{(\mathrm{s})}\delta_{\alpha\beta} + \eta_{\alpha\beta\gamma\delta}\partial_\beta\partial_\gamma v_\delta^{(\mathrm{s})} \\ &\quad + \frac{1}{\tau}\Omega_+^\dagger\rho^{(\mathrm{d})}\Big(v_\alpha^{(\mathrm{d})} - \Omega_+ v_\alpha^{(\mathrm{s})}\Big) + \frac{\Gamma}{\tau}\Omega_-^\dagger\rho^{(\mathrm{d})}\Big(v_\alpha^{(\mathrm{r})} - \Omega_- v_\alpha^{(\mathrm{s})}\Big).\end{aligned} \tag{4.93}$$

Conservation of the total momentum $\mathbf{p}^{(\mathrm{tot})}$ can be explicitly shown by calculating

$$\frac{\partial \mathbf{p}^{(\mathrm{tot})}}{\partial t} = \int \mathrm{d}^d\mathbf{r}\,\frac{\mathrm{d}}{\mathrm{d}t}\Big[\mathbf{j}^{(\mathrm{d})} + \mathbf{j}^{(\mathrm{s})}\Big] \tag{4.94}$$

making use of the relations

$$\int \mathrm{d}^d\mathbf{r}\,\Omega_+^\dagger f(\mathbf{r}) = \int \mathrm{d}^d\mathbf{r}\,f(\mathbf{r}), \tag{4.95}$$

$$\int \mathrm{d}^d\mathbf{r}\,\Omega_-^\dagger f(\mathbf{r}) = 0, \tag{4.96}$$

and the fact that the integrals over divergences vanish. Note that no overall momentum is produced by the relative motion.

Furthermore, it is now possible to write down the complete matrix $\mathbf{M}$,

$$\mathbf{M} = \frac{1}{\tau}\begin{pmatrix} \rho^{(\mathrm{d})} & 0 & -\rho^{(\mathrm{d})}\Omega_+ \\ 0 & \Gamma\rho^{(\mathrm{d})} & -\Gamma\rho^{(\mathrm{d})}\Omega_- \\ -\Omega_+^\dagger\rho^{(\mathrm{d})} & -\Gamma\Omega_-^\dagger\rho^{(\mathrm{d})} & \Omega_+^\dagger\rho^{(\mathrm{d})}\Omega_+ + \Omega_-^\dagger\rho^{(\mathrm{d})}\Omega_- \end{pmatrix}, \tag{4.97}$$

which is obviously self-adjoint.

While the calculation of the Poisson brackets is more easily done via the currents, one might have a better intuition of the equations when they are written in terms of velocities. We can facilitate such a transformation in a general fashion by inspecting currents of the form

$$j_\alpha^{(x)} = c^{(x)}\rho^{(\mathrm{d})}v_\alpha^{(x)}. \tag{4.98}$$

In particular, here we have $c^{(\mathrm{d})} = 1$ and $c^{(\mathrm{r})} = \Gamma$. The relative density $\mathbf{k}^{(\mathrm{r})} = k\rho^{(\mathrm{d})}\mathbf{q}/m^{(\mathrm{d})}$ can be treated analogously by choosing a c-factor of $k/m^{(\mathrm{d})}$. We apply this for the sum of the time derivative of a current and its convection term and find using the continuity equation for $\rho^{(\mathrm{d})}$:

$$\begin{aligned} \partial_t j_\alpha^{(x)} + \partial_\beta\left(\frac{j_\alpha^{(x)}j_\beta^{(\mathrm{d})}}{\rho^{(\mathrm{d})}}\right) &= c^{(x)}\Big[\rho^{(\mathrm{d})}\partial_t v_\alpha^{(x)} - v_\alpha^{(x)}\partial_\beta j_\beta^{(\mathrm{d})} + \partial_\beta\Big(v_\alpha^{(x)}j_\beta^{(\mathrm{d})}\Big)\Big] \\ &= c^{(x)}\Big[\rho^{(\mathrm{d})}\partial_t v_\alpha^{(x)} + j_\beta^{(\mathrm{d})}\partial_\beta v_\alpha^{(x)}\Big] = c^{(x)}\rho^{(\mathrm{d})}\Big[\partial_t v_\alpha^{(x)} + v_\beta^{(\mathrm{d})}\partial_\beta v_\alpha^{(x)}\Big] \\ &= c^{(x)}\rho^{(\mathrm{d})}\mathrm{D}_t^{(\mathrm{d})}v_\alpha^{(x)}, \end{aligned} \tag{4.99}$$

where $\mathrm{D}_t^{(x)} = \partial_t + v_\beta^{(x)}\partial_\beta$ is the convective derivative with respect to the velocity $\mathbf{v}^{(x)}$. The full model is then:

$$\mathrm{D}_t^{(\mathrm{d})}\rho^{(\mathrm{d})} = -\rho^{(\mathrm{d})}\boldsymbol{\nabla}\cdot\mathbf{v}^{(\mathrm{d})}, \tag{4.100}$$

$$\rho^{(\mathrm{d})}\mathrm{D}_t^{(\mathrm{d})}\mathbf{v}^{(\mathrm{d})} = -\boldsymbol{\nabla}P^{(\mathrm{d})} + \kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\nabla^2\rho^{(\mathrm{d})} - \frac{1}{\tau}\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{d})} - \Omega_+\mathbf{v}^{(\mathrm{s})}\Big), \tag{4.101}$$

$$\mathrm{D}_t^{(\mathrm{d})}\mathbf{q} = \mathbf{v}^{(\mathrm{r})}, \tag{4.102}$$

$$\mathrm{D}_t^{(\mathrm{d})}\mathbf{v}^{(\mathrm{r})} = -\frac{k}{m^{(\mathrm{r})}}\mathbf{q} - \frac{1}{\tau}\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{r})} - \Omega_-\mathbf{v}^{(\mathrm{s})}\Big), \tag{4.103}$$

$$\mathrm{D}_t^{(\mathrm{s})}\rho^{(\mathrm{s})} = -\rho^{(\mathrm{s})}\boldsymbol{\nabla}\cdot\mathbf{v}^{(\mathrm{s})}, \tag{4.104}$$

$$\begin{aligned} \rho^{(\mathrm{s})}\mathrm{D}_t^{(\mathrm{s})}\mathbf{v}^{(\mathrm{s})} = &-\boldsymbol{\nabla}P^{(\mathrm{s})} + \boldsymbol{\eta}\vdots\boldsymbol{\nabla}\boldsymbol{\nabla}\mathbf{v}^{(\mathrm{s})} \\ &+ \frac{1}{\tau}\Omega_+^\dagger\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{d})} - \Omega_+\mathbf{v}^{(\mathrm{s})}\Big) + \frac{\Gamma}{\tau}\Omega_-^\dagger\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{r})} - \Omega_-\mathbf{v}^{(\mathrm{s})}\Big). \end{aligned} \tag{4.105}$$

Equation (4.100) is the continuity equation for the dumbbell component. Equation (4.101) is the Euler equation for the dumbbell component augmented with a surface term and

coupling to the solvent as driving forces. Equations (4.102) and (4.103) are the dynamic equations for the relative component of the dumbbell. They essentially describe a dampened harmonic oscillator with the solvent velocity field as driving force. Finally, eqs. (4.104) and (4.105) are the continuity equation of the solvent component and the Navier-Stokes equation of the solvent, driven by coupling to the center of mass and relative component of the dumbbell component. A brief diagrammatic summary of the derivation process is given in fig. 4.2.

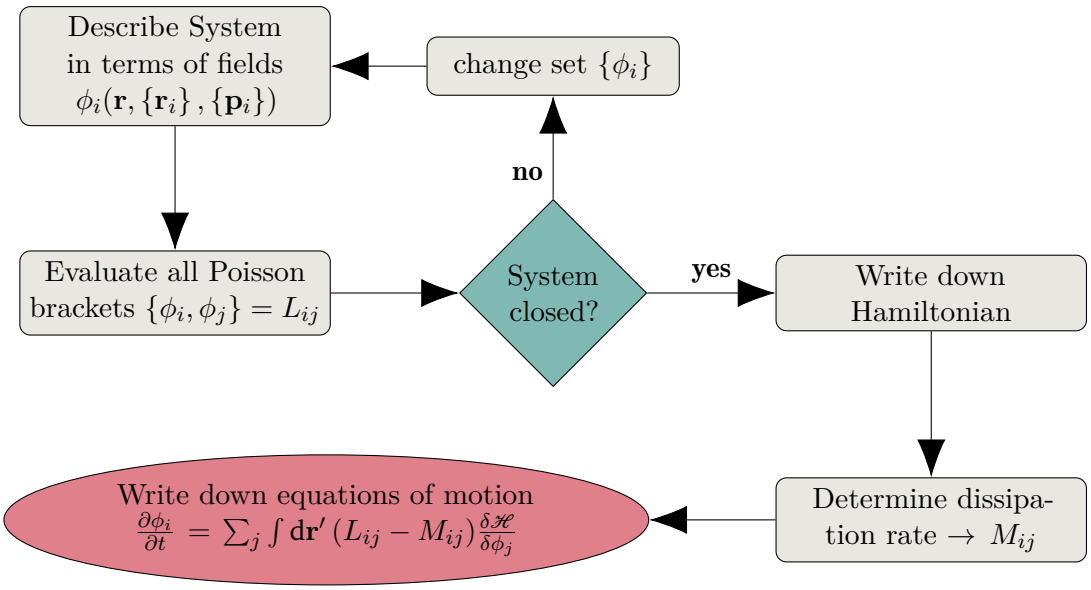


Figure 4.2: Summary of the derivation of the equations of motion.

Equations (4.100) to (4.105) were derived, making no approximations other than postulating a continuum equivalent for the Hamiltonian and the dissipation rate, which is possible only due to the utmost simplicity of the underlying microscopic picture. They comprise two scalar equations and four vector-valued equations and, in d dimensions, $2 + 4d$ variables in total. Parameters are the interfacial stiffness κ, friction time scale τ, the ratio of spring constant to reduced mass $k/m^{(\mathrm{r})}$, as well as the solvent's shear and bulk viscosity η_s and η_b. While in the microscopic interpretation $\Gamma = m^{(\mathrm{r})}/m^{(\mathrm{d})} = 1/4$ has a well-defined constant value, it might be useful to interpret this as a free parameter that can be used to tune the strength of solvent-relative coupling.

For the operators Ω_+ and Ω_-, a maximum order in $\mathbf{q}$ at which the expansion is terminated must be chosen. Furthermore, different interpolation kernels other than $\omega(\mathbf{r}) = \delta(\mathbf{r})$, e.g. Gaussian functions can be chosen. In that way, not only the solvent velocity at the exact point of the particle location is included in the coupling, but a broader range of velocities that depends on the width of the kernel.

The partial pressures of dumbbell and solvent components, $P^{(\mathrm{d})}$ and $P^{(\mathrm{s})}$, follow by their respective thermodynamic equation of state. As the LB/MD coupling scheme (section 3.2.4) is used as an inspiration for these derivations, it is reasonable to use the equation of state of the ideal gas for the solvent

$$P^{(\mathrm{s})} = \rho^{(\mathrm{s})} c_{\mathrm{s}}^2. \tag{4.106}$$

For the dumbbell component, we propose the Van der Waals equation of state derived in section 2.3,

$$P^{(\mathrm{d})} = \frac{N k_{\mathrm{B}} T}{MN} \frac{n}{1 - Bn} - An^2. \tag{4.107}$$

The vectors $\mathbf{q}$ and $-\mathbf{q}$ describe the same dumbbell; therefore, all the equations are covariant under a flip transformation $\mathbf{q} \to -\mathbf{q}$, $\mathbf{v}^{(\mathrm{r})} \to -\mathbf{v}^{(\mathrm{r})}$, meaning that all terms in an equation transform equally. This ambiguity is conceptually somewhat problematic as it allows for discontinuous jumps in the fields $\mathbf{q}(\mathbf{r})$ and $\mathbf{v}^{(\mathrm{r})}(\mathbf{r})$, which is especially problematic in the numerical simulation of the continuum equations. We therefore require the fields to be as smooth as possible. In the microscopic picture, this could be facilitated by a procedure in which the orientation of the first dumbbell is chosen randomly. The orientation of the neighboring dumbbells is then successively chosen in such a way that alignment is as good as possible. This is done until all dumbbells are assigned an orientation.

It would be desirable to have a quantity that is independent of the sign to describe orientation altogether. One candidate satisfying this requirement is the conformation tensor used in the Oldroyd-B model (section 2.5). Since the present model has, in contrast to the Oldroyd-B model, no overdamping approximations and inertia plays a role both in the center of mass and relative motion, one would also need an additional, flip invariant quantity to substitute for $\mathbf{v}^{(\mathrm{r})}$. While starting from a description that is flip invariant right away introduces new complications, c.f. appendix A.3.3, we shall see in the next section that with suitable approximations, it is possible to simplify the equations in such a way that the result is completely flip invariant.

4.1.5 New variables

Before taking any approximations, we first introduce new dynamic variables that better reflect the system's two-fluid nature. Instead of the basic densities $\rho^{(\mathrm{d})}$ and $\rho^{(\mathrm{s})}$ we use the total density

$$\rho = \rho^{(\mathrm{d})} + \rho^{(\mathrm{s})} \tag{4.108}$$

and the normalized density difference

$$c = \frac{\rho^{(\mathrm{d})} - \rho^{(\mathrm{s})}}{\rho}, \tag{4.109}$$

which changes sign when $\rho^{(\mathrm{d})}$ and $\rho^{(\mathrm{s})}$ are exchanged. Furthermore, it is helpful to define the reduced density

$$\rho^{(\mathrm{red})} = \frac{\rho^{(\mathrm{d})}\rho^{(\mathrm{s})}}{\rho} = \frac{\rho}{4}\Big(1 - c^2\Big). \tag{4.110}$$

Note that the reduced density alone would not be sufficient for a complete description as it is not sensitive to the exchange of both basic densities. Therefore we choose the density difference c as a fundamental variable for now and use $\rho^{(\mathrm{red})}$ only to reduce notation.

Instead of the basic velocities $\mathbf{v}^{(\mathrm{d})}$ and $\mathbf{v}^{(\mathrm{s})}$, we introduce the barycentric (mass-average) velocity

$$\mathbf{v} = \frac{\rho^{(\mathrm{d})}\mathbf{v}^{(\mathrm{d})} + \rho^{(\mathrm{s})}\mathbf{v}^{(\mathrm{s})}}{\rho}, \tag{4.111}$$

and the velocity difference

$$\mathbf{w} = \mathbf{v}^{(\mathrm{d})} - \mathbf{v}^{(\mathrm{s})}. \tag{4.112}$$

We also take note of the inverse transformations

$$\mathbf{v}^{(\mathrm{d})} = \mathbf{v} + \frac{\rho^{(\mathrm{s})}}{\rho}\mathbf{w}, \qquad \mathbf{v}^{(\mathrm{s})} = \mathbf{v} - \frac{\rho^{(\mathrm{d})}}{\rho}\mathbf{w}, \tag{4.113}$$

$$\rho^{(\mathrm{d})} = \frac{\rho}{2}(1 + c), \qquad \rho^{(\mathrm{s})} = \frac{\rho}{2}(1 - c). \tag{4.114}$$

Furthermore, we define a new convective derivative with respect to the barycentric velocity

$$\mathrm{D}_t = \partial_t + \mathbf{v} \cdot \boldsymbol{\nabla}. \tag{4.115}$$

Transformation between the different convective derivatives is facilitated by the relations

$$\mathrm{D}_t^{(\mathrm{d})} = \mathrm{D}_t + \frac{\rho^{(\mathrm{s})}}{\rho}\mathbf{w} \cdot \boldsymbol{\nabla}, \qquad \mathrm{D}_t^{(\mathrm{s})} = \mathrm{D}_t - \frac{\rho^{(\mathrm{d})}}{\rho}\mathbf{w} \cdot \boldsymbol{\nabla}. \tag{4.116}$$

Introducing the abbreviations

$$\rho^{(\mathrm{d})}\mathrm{D}_t^{(\mathrm{d})}\mathbf{v}^{(\mathrm{d})} = \mathbf{g}^{(\mathrm{d})}, \qquad \rho^{(\mathrm{s})}\mathrm{D}_t^{(\mathrm{s})}\mathbf{v}^{(\mathrm{s})} = \mathbf{g}^{(\mathrm{s})}, \tag{4.117}$$

for the convective equations of motion in the original variables, the dynamic equations in the new variables can be brought into the form

$$\mathrm{D}_t\rho = -\rho\boldsymbol{\nabla} \cdot \mathbf{v}, \tag{4.118}$$

$$\rho\mathrm{D}_t c = -2\boldsymbol{\nabla} \cdot \Big(\rho^{(\mathrm{red})}\mathbf{w}\Big), \tag{4.119}$$

$$\rho\mathrm{D}_t\mathbf{v} = \mathbf{g}^{(\mathrm{d})} + \mathbf{g}^{(\mathrm{s})} - \boldsymbol{\nabla} \cdot \Big(\rho^{(\mathrm{red})}\mathbf{w}\mathbf{w}\Big), \tag{4.120}$$

$$\mathrm{D}_t\mathbf{w} = \frac{\mathbf{g}^{(\mathrm{d})}}{\rho^{(\mathrm{d})}} - \frac{\mathbf{g}^{(\mathrm{s})}}{\rho^{(\mathrm{s})}} - \mathbf{w} \cdot \boldsymbol{\nabla}\mathbf{v} + c\mathbf{w} \cdot \boldsymbol{\nabla}\mathbf{w} + \frac{\mathbf{w}\mathbf{w}}{2} \cdot \boldsymbol{\nabla}c. \tag{4.121}$$

Furthermore, the conservation of dumbbell mass can be written as

$$\mathrm{D}_t\rho^{(\mathrm{d})} = -\rho^{(\mathrm{d})}\boldsymbol{\nabla}\cdot\mathbf{v} - \boldsymbol{\nabla}\cdot\Big(\rho^{(\mathrm{red})}\mathbf{w}\Big). \tag{4.122}$$

These transformations are somewhat tedious but straightforward; details can be found found in appendix A.2.2.

4.1.6 Approximations

In a first step of simplifying these equations, we assume that inertial effects are negligible in the motion of the harmonic oscillator by making the overdamping approximation $\mathrm{D}_t^{(\mathrm{d})}\mathbf{v}^{(\mathrm{r})} = 0$. Applied to eq. (4.103), this yields

$$\mathbf{v}^{(\mathrm{r})} = \Omega_-\mathbf{v}^{(\mathrm{s})} - \frac{k\tau}{m^{(\mathrm{r})}}\mathbf{q}, \tag{4.123}$$

allowing us to eliminate $\mathbf{v}^{(\mathrm{r})}$ entirely from our set of equations. By defining $\tau_q = m^{(\mathrm{r})}/(k\tau)$, the equation for the orientational field becomes

$$\mathrm{D}_t\mathbf{q} = \Omega_-\mathbf{v}^{(\mathrm{s})} - \frac{1}{\tau_q}\mathbf{q}. \tag{4.124}$$

Furthermore, $\mathbf{v}^{(\mathrm{r})}$ also appears in the coupling to the solvent velocity eq. (4.105). After insertion of eq. (4.123) the equation reads

$$\begin{aligned}\rho^{(\mathrm{s})}\mathrm{D}_t^{(\mathrm{s})}\mathbf{v}^{(\mathrm{s})} = &-\boldsymbol{\nabla}P^{(\mathrm{s})} + \boldsymbol{\eta}\vdots\boldsymbol{\nabla}\boldsymbol{\nabla}\mathbf{v}^{(\mathrm{s})} \\ &+ \frac{1}{\tau}\Omega_+^\dagger\rho^{(\mathrm{d})}\Big(\mathbf{v}^{(\mathrm{d})} - \Omega_+\mathbf{v}^{(\mathrm{s})}\Big) - \frac{\Gamma}{\tau\tau_q}\Omega_-^\dagger\rho^{(\mathrm{d})}\mathbf{q}.\end{aligned} \tag{4.125}$$

As the next approximation, we stop the expansion of the Ω operators after the linear order in $\mathbf{q}$, i.e. $\Omega_+ = 1$ and $\Omega_- = \boldsymbol{\nabla}\cdot\mathbf{q}$. Hence

$$\mathbf{g}^{(\mathrm{d})} = -\boldsymbol{\nabla}P^{(\mathrm{d})} + \kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\boldsymbol{\nabla}^2\rho^{(\mathrm{d})} - \frac{\rho^{(\mathrm{d})}}{\tau}\mathbf{w}, \tag{4.126}$$

$$\mathbf{g}^{(\mathrm{s})} = -\boldsymbol{\nabla}P^{(\mathrm{s})} + \boldsymbol{\eta}\vdots\boldsymbol{\nabla}\boldsymbol{\nabla}\left(\mathbf{v} - \frac{\rho^{(\mathrm{d})}}{\rho}\mathbf{w}\right) + \frac{\rho^{(\mathrm{d})}}{\tau}\mathbf{w} + \frac{\Gamma}{\tau\tau_q}\boldsymbol{\nabla}\cdot\Big(\rho^{(\mathrm{d})}\mathbf{q}\mathbf{q}\Big), \tag{4.127}$$

and the dynamic equation for $\mathbf{q}$ is obtained by transformation of the convective derivative:

$$\mathrm{D}_t^{(\mathrm{d})} = \mathrm{D}_t + \frac{\rho^{(\mathrm{s})}}{\rho}\mathbf{w}\cdot\boldsymbol{\nabla}, \tag{4.128}$$

$$\mathrm{D}_t^{(\mathrm{d})}\mathbf{q} = \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}^{(\mathrm{s})} - \frac{1}{\tau_q}\mathbf{q}, \tag{4.129}$$

$$\mathrm{D}_t\mathbf{q} = \mathbf{q}\cdot\boldsymbol{\nabla}\left(\mathbf{v} - \frac{\rho^{(\mathrm{d})}}{\rho}\mathbf{w}\right) - \frac{\rho^{(\mathrm{s})}}{\rho}\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{q} - \frac{1}{\tau_q}\mathbf{q}. \tag{4.130}$$

Incompressibility is applied to the total mass density ρ, i.e. $\mathrm{D}_t\rho$, $\boldsymbol{\nabla}\rho$ and $\boldsymbol{\nabla}\cdot\mathbf{v}$ all vanish. For the viscous stress wrt., $\mathbf{v}$ only the shear component remains: $\boldsymbol{\eta}\vdots\boldsymbol{\nabla}\boldsymbol{\nabla}\mathbf{v} = \eta_\mathrm{s}\boldsymbol{\nabla}^2\mathbf{v}$. The total pressure, $P = P^{(\mathrm{d})} + P^{(\mathrm{s})}$, is then no longer related to a thermodynamic potential but acts like a constraint force enforcing incompressibility.

As the relative velocity $\mathbf{w}$ is not a hydrodynamic variable, we shall assume that it relaxes rather quickly and is always small. Furthermore, it is assumed that the dynamics of $\mathbf{w}$ is overdamped and that, similar to $\mathbf{v}^{(\mathrm{r})}$, $\mathbf{w}$ can be eliminated adiabatically. Since the elimination of $\mathbf{w}$ by overdamping approximation is not as straightforward as for $\mathbf{v}^{(\mathrm{r})}$, and maybe even impossible, we take the following strategy: All terms containing $\mathbf{w}$ are replaced phenomenological terms which are later on chosen in such a way that they are consistent with the principles of non-equilibrium thermodynamics. The term $\rho^{(\mathrm{red})}\mathbf{w}$ is replaced by the interdiffusion current $\mathbf{j}^{(\mathrm{int})}$ and a vector $\mathbf{Q}$ replaces the terms $-\mathbf{q}\cdot\boldsymbol{\nabla}\left(\rho^{(\mathrm{d})}\mathbf{w}/\rho\right) - \rho^{(\mathrm{s})}\mathbf{w}/\rho\cdot\boldsymbol{\nabla}\mathbf{q}$. The divergence of a stress tensor $\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}$ is introduced to replace $-\boldsymbol{\eta}\vdots\boldsymbol{\nabla}\boldsymbol{\nabla}\left(\rho^{(\mathrm{d})}\mathbf{w}/\rho\right) - \boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{red})}\mathbf{w}\mathbf{w}\right)$. Writing this as a divergence is done in order to preserve momentum conservation. The thus modified equations then read

$$\mathrm{D}_t\rho^{(\mathrm{d})} = -\boldsymbol{\nabla}\cdot\mathbf{j}^{(\mathrm{int})}, \tag{4.131}$$

$$\mathrm{D}_t\mathbf{q} = \mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v} - \frac{1}{\tau_q}\mathbf{q} + \mathbf{Q}, \tag{4.132}$$

$$\rho\mathrm{D}_t\mathbf{v} = -\boldsymbol{\nabla}P + \kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\boldsymbol{\nabla}^2\rho^{(\mathrm{d})} + \eta_\mathrm{s}\boldsymbol{\nabla}^2\mathbf{v} + \frac{\Gamma}{\tau\tau_q}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{q}\mathbf{q}\right) + \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}, \tag{4.133}$$

$$\boldsymbol{\nabla}\cdot\mathbf{v} = 0. \tag{4.134}$$

Whether terms in a dynamic equation are of conservative or dissipative nature can be determined from their sign-changing behavior under time-reversal. If the term is covariant with the time derivative on the left-hand side, it is conservative; if it is contravariant, it is dissipative. For example, $\mathrm{D}_t\mathbf{v}$ does not change its sign when time is reversed, but $\eta_\mathrm{s}\nabla^2\mathbf{v}$ does. Therefore, $\eta_\mathrm{s}\nabla^2\mathbf{v}$ must be of dissipative nature. Note that this procedure can not be applied to the phenomenological terms. Typically however, conservative terms are turned into dissipative ones by such phenomenological replacements.

For the system with the new variables, we define the Hamiltonian by

$$\mathscr{H} = \int \mathrm{d}^d\mathbf{r}\left[\frac{\rho}{2}\mathbf{v}^2 + f + \frac{\kappa}{2}\left(\boldsymbol{\nabla}\rho^{(\mathrm{d})}\right)^2 + \frac{1}{2}\frac{\Gamma}{\tau\tau_q}\rho^{(\mathrm{d})}\mathbf{q}^2\right]. \tag{4.135}$$

The phenomenological terms $\boldsymbol{\nabla}\cdot\mathbf{j}^{(\mathrm{int})}$, $\mathbf{Q}$, and $\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}$ must be chosen in such a way that consistency with non-equilibrium thermodynamic is ensured. Considering the dissipation rate

$$\frac{\mathrm{d}\mathscr{H}}{\mathrm{d}t} = \int \mathrm{d}^d\mathbf{r}\left[\frac{\delta\mathscr{H}}{\delta\mathbf{v}}\partial_t\mathbf{v} + \frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}}\partial_t\rho^{(\mathrm{d})} + \frac{\delta\mathscr{H}}{\delta\mathbf{q}}\partial_t\mathbf{q}\right], \tag{4.136}$$

one obtains after some calculation (see appendix A.3.2)

$$\int \mathrm{d}^d\mathbf{r}\,\frac{\delta\mathscr{H}}{\delta\mathbf{v}}\partial_t\mathbf{v} = -\int \mathrm{d}^d\mathbf{r}\,\partial_\beta v_\alpha(\eta_\mathrm{s}\partial_\beta v_\alpha + \sigma_{\alpha\beta}), \tag{4.137}$$

$$\int \mathrm{d}^d\mathbf{r}\,\frac{\delta\mathscr{H}}{\delta\mathbf{q}}\partial_t\mathbf{q} = -\frac{\Gamma}{\tau\tau_q}\int \mathrm{d}^d\mathbf{r}\,\rho^{(\mathrm{d})}\mathbf{q}\left(\frac{1}{\tau_q}\mathbf{q}+\mathbf{Q}\right), \tag{4.138}$$

$$\int \mathrm{d}^d\mathbf{r}\,\frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}}\partial_t\rho^{(\mathrm{d})} = \int \mathrm{d}^d\mathbf{r}\,\mathbf{j}^{(\mathrm{int})}\cdot\boldsymbol{\nabla}\frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}}. \tag{4.139}$$

This identifies all dissipative terms, in particular the phenomenological quantities $\mathbf{j}^{(\mathrm{int})}$, $\mathbf{Q}$, and $\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}$. For the second law to be satisfied, all of the above terms must be be negative. The simplest approach is to choose $\sigma_{\alpha\beta}\propto\partial_\beta v_\alpha$ and $\mathbf{Q}\propto\mathbf{q}$ with positive proportionality constants, such that their effect can be absorbed by a redefinition of η_s and τ_q respectively. For the interdiffusion current we choose

$$\mathbf{j}^{(\mathrm{int})} = -M\left(\rho^{(\mathrm{d})}\right)\boldsymbol{\nabla}\frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}}, \tag{4.140}$$

where $M(\rho^{(\mathrm{d})})$ is the Onsager transport coefficient for interdiffusion (c.f. eq. (2.81) in section 2.4). With the derivative

$$\frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}} = \frac{\partial f}{\partial\rho^{(\mathrm{d})}} - \kappa\nabla^2\rho^{(\mathrm{d})} + \frac{\Gamma}{2\tau\tau_q}\mathbf{q}^2, \tag{4.141}$$

one sees that the interdiffusion current is driven by bulk, interface, as well as elastic effects. A coupling to elastic stresses was first predicted by Doi and Onuki in [13].

By straightforward calculation, it can be shown that the conservative part of dynamics derived via Poisson brackets is not compatible with the simplified set of eqs. (4.131) to (4.134). Instead, we must suppose that by the adiabatic elimination of $\mathbf{v}^{(\mathrm{r})}$ and $\mathbf{w}$, the Hamiltonian structure of the system is lost.

Equation eq. (4.132) can be transformed into a dynamic equation for the conformation tensor which is invariant wrt. to flip transformation:

$$\begin{aligned}\mathrm{D}_t C_{\alpha\beta} &= \mathrm{D}_t(q_\alpha q_\beta) = q_\alpha \mathrm{D}_t q_\beta + q_\beta \mathrm{D}_t q_\alpha \\ &= q_\alpha q_\gamma\partial_\gamma v_\beta + q_\beta q_\gamma\partial_\gamma v_\alpha - \frac{2}{\tau_q}q_\alpha q_\beta \\ &= C_{\alpha\gamma}\partial_\gamma v_\beta + C_{\beta\gamma}\partial_\gamma v_\alpha - \frac{2}{\tau_q}C_{\alpha\beta}.\end{aligned} \tag{4.142}$$

With the velocity gradient tensor $K_{\alpha\beta}=\partial_\beta v_\alpha$, this becomes

$$\mathrm{D}_t\mathbf{C} = \mathbf{C}\mathbf{K}^\mathsf{T} + \mathbf{K}\mathbf{C} - \frac{2}{\tau_q}\mathbf{C}, \tag{4.143}$$

where the symmetry of the conformation tensor $\mathbf{C} = \mathbf{C}^{\mathsf{T}}$ has been used. With the upper convected derivative introduced in eq. (2.152), this takes the compact form

$$\overset{\triangledown}{\mathbf{C}} = -\frac{2}{\tau_q}\mathbf{C}. \tag{4.144}$$

The upper convected derivative describes the convection and deformation of tensor quantities under flow and here follows naturally from the microscopic description. Compared to the Oldroyd-B constitutive equation eq. (2.153), which reads

$$\overset{\triangledown}{\mathbf{C}} = -\frac{2}{\tau_q}\left(\mathbf{C} - \frac{k_{\mathrm{B}}T}{k}\mathbb{1}\right), \tag{4.145}$$

we see that there is no constant term subtracted on the right-hand side. This is because, in the present derivation, thermal fluctuations are not yet taken into account and the equilibrium value of the conformation tensor is the mechanical one, $\mathbf{C}_{\mathrm{eq}} = 0$. However, in the Oldroyd-B model, fluctuations are taken into account right away in the Langevin equation for the individual particles. Neglecting the ensemble-defining property of the average conformation tensor, this results in a nonzero average of $\mathbf{C}$. As has been described in more detail in section 2.5.2, this approach commonly taken in the derivation of rheological equations can be seen as problematic. This is because statistical averages are taken inside small volume elements in which all fast variables are in local equilibrium. Hence the slow variables, in particular including the conformation tensor $\mathbf{C}$, define a local thermodynamic ensemble, and statistical averages should be constrained to said ensemble. Doing so then leads to eq. (2.175) which is equivalent to eq. (4.143). It therefore seems more consistent to calculate the equations of motion without fluctuations first and only add those at the very end while making sure that the fluctuation-dissipation theorem is satisfied. This will be the subject of future work.

To summarize, the final set of approximated equations is

$$\mathrm{D}_t\rho^{(\mathrm{d})} = -\boldsymbol{\nabla}\cdot\mathbf{j}^{(\mathrm{int})}, \tag{4.146}$$

$$\overset{\triangledown}{\mathbf{C}} = -\frac{2}{\tau_q}\mathbf{C}, \tag{4.147}$$

$$\rho\mathrm{D}_t\mathbf{v} = -\boldsymbol{\nabla}P - \kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\boldsymbol{\nabla}^2\rho^{(\mathrm{d})} + \eta_{\mathrm{s}}\boldsymbol{\nabla}^2\mathbf{v} - \frac{\Gamma}{\tau\tau_q}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{C}\right), \tag{4.148}$$

$$\boldsymbol{\nabla}\cdot\mathbf{v} = 0. \tag{4.149}$$

4.2 Semi-automatic construction of Lattice Boltzmann models

When constructing Lattice Boltzmann (LB) models, the choice of the velocity set and the determination of respective weights for the equilibrium distributions are central steps (c.f. section 3.2). For a given set of velocities $\{\mathbf{c}_i\}$, one way of determining the weights is finding the solution to the Maxwell-Boltzmann constraint (MBC) equations

$$\sum_i w_i c_{i\alpha_1} c_{i\alpha_2} \dots c_{i\alpha_m} = \left(2\pi c_\mathrm{s}^2\right)^{-d/2} \int \mathrm{d}^d\mathbf{v} \exp\left(-\frac{\mathbf{v}^2}{2c_\mathrm{s}^2}\right) v_{\alpha_1} v_{\alpha_2} \dots v_{\alpha_m}, \tag{4.150}$$

$$\sum_i w_i = 1, \tag{4.151}$$

$$w_i \geq 0 \,\forall\, i, \tag{4.152}$$

where $\{w_i\}$ is a normalized set of positive weights and the index i enumerates all velocities.

In this section, which is based on the publication [79], it is shown how this mathematical problem can be recast in the language of linear algebra, which simplifies its numerical solution. The resulting procedure has been automatized in the publicly available Python script `LBWeights.py` [84]; the documentation of this script can be found in appendix B.1. Input parameters that need to be supplied by the user are the spacial dimension d, the maximum tensor order M up to which eq. (4.150) should be satisfied, an integer seed for the random number generator, and a list of $\tilde{c}_i^2$ values to be used to generate the velocity set.

4.2.1 Velocity shells

Here we restrict ourselves to velocities $\mathbf{c}_i$ on a cubic lattice with lattice constant 1. Therefore, the script first verifies that the user-supplied $\tilde{c}_i^2$ values are indeed compatible with the underlying lattice. If this is the case, the velocity 'modulus shells' $S_i^{(\mathrm{m})}$ are generated. These are sets of lattice velocities $\mathbf{c}_j$ that are compatible with a supplied modulus, i.e.

$$S_i^{(\mathrm{m})} = \left\{\mathbf{c}_j \in \mathbb{Z}^d : c_j^2 = \tilde{c}_i^2\right\}. \tag{4.153}$$

Note that the zero velocity $\mathbf{c}_0 = 0$ is included implicitly and forms its own shell $S_0^{(\mathrm{m})}$.

Furthermore, we define an 'equivalence shell' $S_i^{(\mathrm{e})}$ as an equivalence class of velocities $\mathbf{c}_j$ that can be mapped onto each other by transformations $\mathbf{T}_k$ in the cubic group. Each modulus shell consists of one or more of such 'equivalence shells'. With this definition of a shell, a wider range of models can be treated, such as, for example, the D3Q41-ZOT

model by Chikatamarla and Karlin [88]. To exemplify the concept of equivalence shells, consider the modulus shell with $c_i^2 = 9$ in three dimensions which can be divided into two equivalence shells:

$$\left\{ \mathbf{c}_i : \mathbf{c}_i \sim \begin{pmatrix} 0 \\ 0 \\ 3 \end{pmatrix} \right\} \qquad \text{(6 velocities)}, \tag{4.154}$$

$$\left\{ \mathbf{c}_i : \mathbf{c}_i \sim \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \right\} \qquad \text{(24 velocities)}. \tag{4.155}$$

By $\mathbf{c}_i \sim \mathbf{a}$ we mean that $\mathbf{c}_i$ is in the same equivalence class as $\mathbf{a}$, i.e. there exists some k such that $\mathbf{c}_i = \mathbf{T}_k \mathbf{a}$. In particular, there is no transformation $\mathbf{T}_k$ in the cubic group that can transform a vector of class $(0, 0, 3)$ into a vector of class $(1, 2, 2)$, and yet they both have the same absolute value.

The script detects the equivalence shells by explicitly generating all cubic group transformations and applying them to the velocities in the modulus shells, thus spanning the equivalence shells. A transformation in the cubic group can be interpreted as a combination of 90° rotations about the Cartesian coordinate axes combined with a possible reflection about the origin. The columns of a transformation matrix $\mathbf{T}_k$ must therefore be some permutation π_k of the canonical unit vectors in d dimensions, $\pm\hat{\mathbf{e}}_i$ with $e_{i\alpha} = \delta_{i\alpha}$, $i = 1, \ldots, d$. We denote this by

$$\mathbf{T}_k = \Big(s_k(1)\hat{\mathbf{e}}_{\pi_k(1)}, s_k(2)\hat{\mathbf{e}}_{\pi_k(2)}, \ldots, s_k(d)\hat{\mathbf{e}}_{\pi_k(d)} \Big), \tag{4.156}$$

where $s_k(i) = \pm 1$ and each permutation π occurs with every possible combination of signs s. As there are $d!$ possible permutations of d indexes and 2^d possible combination of signs, the cubic group in d dimensions consists of $d!2^d$ transformations in total. The generation of the group is implemented making use of the permutation routines of the `itertools` package [89], which is part of the Python standard library. After the script has generated the complete cubic group, the list of images of the first velocity in a modulus shell is generated by applying all transformations. This generates the first equivalence shell. The same is then done for the first velocity that is not yet in the first equivalence shell and so on, until all velocities are checked. If a modulus shell decomposes into multiple equivalence shells, the user is asked whether or not any of the subshells should be discarded; the remaining shells are processed further in the following steps.

4.2.2 Dimensionality of the tensor space

Equation (4.150) is a tensor equation of rank m, where m is the number of Greek indexes. We want to satisfy this equation for all ranks $m \leq M$, where we call M the degree of

isotropy of the model. Since for uneven m the MBCs are trivially satisfied because of the symmetry of the velocity shells, we require M to be an even number. Then there are $M/2$ tensor equations to be solved in total. For each rank m, we can write the tensors on the left-hand side (LHS) of eq. (4.150) as expansion in terms of some basis tensors $\left\{\mathbf{B}_j^{(m)}\right\}$. The LHS must therefore have the form

$$\sum_i w_i c_{i\alpha_1} c_{i\alpha_2} \dots c_{i\alpha_m} = \sum_{j=1}^{D_{\mathrm{T}}(m)} \lambda_j^{(m)} B_{j\alpha_1\alpha_2\dots\alpha_m}^{(m)}. \tag{4.157}$$

The LHS is invariant under the exchange of indexes and under transformations in the cubic group. The right-hand side (RHS) of eq. (4.150) however is invariant under exchange of indexes and arbitrary rotations. The dimensionality of the tensor space $D_{\mathrm{T}}(m)$ can be determined by counting the minimum number of tensors $\mathbf{B}_i^{(m)}$ necessary to express the velocity moments on the LHS of eq. (4.150) as a linear combination. However, even fewer basis tensors may be needed because some coefficients may vanish due to the rotation invariance of the RHS. For example, this is the case for the coefficient $\lambda_1^{(4)}$ as it is defined below.

For symmetry reasons, the weights for the velocities in an equivalence shell must be the same. Because the flip of the sign is part of the cubic group, there is a velocity with opposite sign $\mathbf{c}_j = -\mathbf{c}_i$ for each velocity $\mathbf{c}_i$ in the same shell. Therefore, the velocity moments only differ from zero if the tensor indexes can be divided into even groups of equal indexes, which is of course only possible for even rank m:

$$\begin{aligned}
&\sum_i w_i c_{i\alpha} = 0 && \Rightarrow \left\{\mathbf{B}_j^{(1)}\right\} = \emptyset && \Rightarrow D_{\mathrm{T}}(1) = 0,\\
&\sum_i w_i c_{i\alpha} c_{i\beta} = \lambda_1^{(2)} \delta_{\alpha\beta} && \Rightarrow \left\{\mathbf{B}_j^{(2)}\right\} = \left\{\boldsymbol{\delta}^{(2)}\right\} && \Rightarrow D_{\mathrm{T}}(2) = 1,\\
&\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} = 0 && \Rightarrow \left\{\mathbf{B}_j^{(3)}\right\} = \emptyset && \Rightarrow D_{\mathrm{T}}(3) = 0,\\
&\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} = \lambda_1^{(4)} \delta_{\alpha\beta\gamma\delta} && && \\
&\quad + \lambda_2^{(4)} (\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}) && \Rightarrow \left\{\mathbf{B}_j^{(4)}\right\} = \left\{\boldsymbol{\delta}^{(4)}, \boldsymbol{\delta}^{(2,2)}\right\} && \Rightarrow D_{\mathrm{T}}(4) = 2.
\end{aligned}$$

Here we have introduced the tensorial generalization of the Kronecker delta

$$\left(\boldsymbol{\delta}^{(m)}\right)_{\alpha_1\dots\alpha_m} = \delta_{\alpha_1\dots\alpha_m} = \begin{cases} 1: & \alpha_1 = \dots = \alpha_m \\ 0: & \text{else} \end{cases}, \tag{4.158}$$

as well as their outer products which are explicitly symmetrized wrt. to index permutation

$$\boldsymbol{\delta}^{(m_1,\dots,m_n)} = \mathrm{Sym}\left(\boldsymbol{\delta}^{(m_1)} \otimes \dots \otimes \boldsymbol{\delta}^{(m_n)}\right). \tag{4.159}$$

These are objects of rank $\sum_{i=1}^{n} m_i$. For higher ranks we find the basis sets

$$\left\{\mathbf{B}_j^{(6)}\right\} = \left\{\boldsymbol{\delta}^{(6)}, \boldsymbol{\delta}^{(4,2)}, \boldsymbol{\delta}^{(2,2,2)}\right\} \qquad \Rightarrow D_{\mathrm{T}}(6) = 3, \tag{4.160}$$

$$\left\{\mathbf{B}_j^{(8)}\right\} = \left\{\boldsymbol{\delta}^{(8)}, \boldsymbol{\delta}^{(6,2)}, \boldsymbol{\delta}^{(4,4)}, \boldsymbol{\delta}^{(4,2,2)}, \boldsymbol{\delta}^{(2,2,2,2)}\right\} \qquad \Rightarrow D_{\mathrm{T}}(8) = 5. \tag{4.161}$$

By now one can quite clearly see that $D_{\mathrm{T}}(m)$ is just the number of ways in which the number $m/2$ can be written as a sum of positive integers. In the language of number theory, $D_{\mathrm{T}}(m)$ is called the partition function of $m/2$. There is no analytical form for the partition function, however it can easily be calculated numerically, for example via recursion.

4.2.3 Contraction to scalar equations

In the next step, for each m, randomly generated unit vectors $\hat{\mathbf{n}}_i^{(m)}$ are used to construct a set of $D_{\mathrm{T}}(m)$ random tensors $\left\{\mathbf{N}_j^{(m)}\right\}$ with

$$N^{(m)}_{j\alpha_1\alpha_2\ldots\alpha_m} = \hat{n}^{(m)}_{j\alpha_1}\hat{n}^{(m)}_{j\alpha_1}\ldots\hat{n}^{(m)}_{j\alpha_m}, \qquad j = 1, \ldots, D_{\mathrm{T}}(m). \tag{4.162}$$

From a numerical standpoint, assuming that each component of the random unit vectors is stored with a precision of about 14 digits, the thus resulting set of random tensors is almost certainly linearly independent. By contracting eq. (4.150) for each m with these tensors, the tensor equations can be transformed to an equivalent set of linear equations, which is much easier to solve with standard numerical methods.

Contraction of the LHS of eq. (4.150) by the random tensors (we imply sum convention for repeated Greek indexes) results in $D_{\mathrm{T}}(m)$ scalar terms per tensor rank m:

$$\begin{aligned}\sum_{i=1}^{D_{\mathrm{T}}(m)} \lambda_i^{(m)} B^{(m)}_{i\alpha_1\alpha_2\ldots\alpha_m} N^{(m)}_{j\alpha_1\alpha_2\ldots\alpha_m} &= \sum_i w_i c_{i\alpha_1} c_{i\alpha_2}\ldots c_{i\alpha_m}\hat{n}^{(m)}_{j\alpha_1}\hat{n}^{(m)}_{j\alpha_2}\ldots\hat{n}^{(m)}_{j\alpha_m} \\ &= \sum_i w_i\left(\mathbf{c}_i\cdot\hat{\mathbf{n}}_j^{(m)}\right)^m, j = 1, \ldots, D_{\mathrm{T}}(m).\end{aligned} \tag{4.163}$$

Therefore there will be

$$R = \sum_{m=2,4,\ldots}^{M} D_{\mathrm{T}}(m) \tag{4.164}$$

linear equations in total. These equations are enumerated by $r = 1, 2, \ldots, R$ where the r-th equation is derived from the tensor equation of rank m_r. Similarly, the random vectors

are simply enumerated by $\hat{\mathbf{n}}_r,\ r = 1, \ldots, R$. The contraction of the RHS is then

$$\begin{aligned} &\left(2\pi c_\mathrm{s}^2\right)^{-d/2} \int \mathrm{d}^d\mathbf{v} \exp\left(-\frac{\mathbf{v}^2}{2c_\mathrm{s}^2}\right)(\mathbf{v}\cdot\hat{\mathbf{n}}_r)^{m_r} \\ &\quad = \left(2\pi c_\mathrm{s}^2\right)^{-1/2} \int \mathrm{d}v_x\, v_x^{m_r} \exp\left(-\frac{v_x^2}{2c_\mathrm{s}^2}\right) = (m_r-1)!!c_\mathrm{s}^{m_r}, \end{aligned} \tag{4.165}$$

where standard relations for Gaussian integration have been applied. The double factorial for odd numbers is defined by $(m_r - 1)!! = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (m_r - 1)$.

By enumerating the non-trivial (not including zero) equivalence shells by $s = 1, \ldots, N_\mathrm{s}$ and exploiting the fact that the weights must be equal within one shell, the set of linear equations can be written as

$$\sum_{s=1}^{N_\mathrm{s}} w_s \sum_{i\in s} (\mathbf{c}_i \cdot \hat{\mathbf{n}}_r)^{m_r} = (m_r-1)!!c_\mathrm{s}^{m_r}, \tag{4.166}$$

where $i \in s$ are indexes of the velocities in shell s. The weight w_0 of the zero shell can be calculated from the normalization condition eq. (4.151) once the other weights are determined. Equation (4.166) can be rearranged to

$$\sum_{s=1}^{N_\mathrm{s}} w_s \frac{\sum_{i\in s}(\mathbf{c}_i\cdot\hat{\mathbf{n}}_r)^{m_r}}{(m_r-1)!!} =: \sum_{s=1}^{N_\mathrm{s}} w_s A_{rs} = c_\mathrm{s}^{m_r}, \tag{4.167}$$

where the $R \times N_\mathrm{s}$ matrix $\mathbf{A}$ was introduced. With $b_r = c_\mathrm{s}^{m_r}$ being the components of the vector $\mathbf{b} \in \mathbb{R}^R$ and the vector of weights $\mathbf{w} \in \mathbb{R}^{N_\mathrm{s}}$, this can be written as

$$\mathbf{A}\cdot\mathbf{w} = \mathbf{b}, \qquad A_{rs} = \frac{\sum_{i\in s}(\mathbf{c}_i\cdot\hat{\mathbf{n}}_r)^{m_r}}{(m_r-1)!!}, \qquad \mathbf{A} \in \mathbb{R}^{R\times N_\mathrm{s}}. \tag{4.168}$$

With the vector of c_s^2 powers, $\mathbf{c} = (c_\mathrm{s}^2, c_\mathrm{s}^4, \ldots, c_\mathrm{s}^M) \in \mathbb{R}^{M/2}$, we can now express $\mathbf{b}$, which itself consists of c_s^2 powers, by multiplying $\mathbf{c}$ from the left with a suitable matrix $\mathbf{D}$:

$$\mathbf{b} = \mathbf{D}\cdot\mathbf{c}, \qquad D_{r\mu} = \delta_{m_r\mu}, \qquad \mathbf{D} \in \mathbb{R}^{R\times M/2}. \tag{4.169}$$

As the matrix $\mathbf{A}$ does not depend on c_s^2, it is clear from eq. (4.168) that the components of $\mathbf{w}$ must be polynomials in c_s^2 and $\mathbf{w}$ can be factorized in a similar way:

$$\mathbf{w} = \mathbf{Q}\cdot\mathbf{c}, \qquad \mathbf{Q} \in \mathbb{R}^{N_\mathrm{s}\times M/2}. \tag{4.170}$$

Putting eqs. (4.169) and (4.170) into eq. (4.168) and canceling the vector $\mathbf{c}$ leads to the new equation

$$\mathbf{A}\mathbf{Q} = \mathbf{D}. \tag{4.171}$$

Due to the cancellation, this equation is now completely independent of the speed of sound. Hence one can look for a solution $\mathbf{Q}$ first and, if such a solution exists, write the weights as polynomials in c_s^2. It then needs to be checked whether or not there are any values of c_s^2 that result in all weights being non-negative. The existence of a solution $\mathbf{Q}$, however, depends on the exact properties of the matrix $\mathbf{A}$.

4.2.4 Differentiating the cases

Whether the system eq. (4.171) has a unique solution, infinitely many solutions, or no solution at all, depends on the details of the matrix $\mathbf{A}$, in particular its shape and rank. Here we make use of the singular-value decomposition [90], which is a generalization of the eigendecomposition that also works for rectangular matrices. The singular-value decomposition of $\mathbf{A}$ has the form

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}, \qquad \mathbf{U} \in \mathbb{R}^{R\times R}, \qquad \mathbf{S} \in \mathbb{R}^{R\times N_\mathrm{s}}, \qquad \mathbf{V} \in \mathbb{R}^{N_\mathrm{s}\times N_\mathrm{s}}, \tag{4.172}$$

where $\mathbf{U}$ and $\mathbf{V}$ are square orthogonal matrices ($\mathbf{U}^{-1} = \mathbf{U}^\mathsf{T}$, $\mathbf{V}^{-1} = \mathbf{V}^\mathsf{T}$) and the matrix $\mathbf{S}$ has the same shape as $\mathbf{A}$. It has the singular values σ_i of $\mathbf{A}$ on the diagonal and is zero otherwise:

$$S_{ij} = \delta_{ij}\sigma_j. \tag{4.173}$$

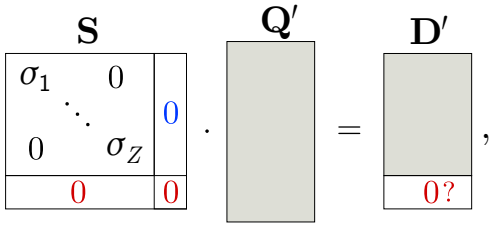
The singular-value decomposition eq. (4.172) is generally not unique. Here we adopt the convention that the singular values must be ordered by size: $\sigma_1 \geq \cdots \geq \sigma_Z > 0$. This determines $\mathbf{S}$, but not necessarily $\mathbf{U}$ and $\mathbf{V}$, uniquely. Implementation in the script is done by applying the routine `numpy.linalg.svd` from the Python `numpy` package [91]. The number of nonzero singular values $Z \leq \min(R, N_\mathrm{s})$ is the rank of $\mathbf{S}$ and at the same time the rank of $\mathbf{A}$. Equation (4.171) can now be rewritten as

$$\mathbf{S}\mathbf{V}^\mathsf{T}\mathbf{Q} = \mathbf{U}^\mathsf{T}\mathbf{D}, \tag{4.174}$$

or, with $\mathbf{Q}' = \mathbf{V}^\mathsf{T}\mathbf{Q}$ and $\mathbf{D}' = \mathbf{U}^\mathsf{T}\mathbf{D}$:

$$\mathbf{S}\mathbf{Q}' = \mathbf{D}', \qquad \mathbf{S} \in \mathbb{R}^{R\times N_\mathrm{s}}, \qquad \mathbf{Q}' \in \mathbb{R}^{N_\mathrm{s}\times M/2}, \qquad \mathbf{D}' \in \mathbb{R}^{R\times M/2}. \tag{4.175}$$

Now there are several different scenarios. If the number of equations R exceeds the rank, $R > Z$, the system of equations is overdetermined. Recalling that $\mathbf{S}$ is a $R \times N_\mathrm{s}$ matrix, the last $R - Z$ rows of $\mathbf{S}$ are then just filled with zeros and eq. (4.175) can be imagined like:



where the blue columns on the right of $\mathbf{S}$ only exist if $Z < N_\mathrm{s}$. It then needs to be checked whether the last $R - Z$ rows of $\mathbf{D}'$ on the RHS are zero as well. If this is not the case, the system is inconsistent and has no solution. If the last $R - Z$ rows on the RHS are zero however, these rows can be pruned from $\mathbf{S}$ and $\mathbf{D}'$ without loss of information. We denote this pruning by $\mathbf{S} \to \mathbf{S}'$ and $\mathbf{D}' \to \mathbf{D}''$ respectively. The new equation then reads

$$\mathbf{S}'\mathbf{Q}' = \mathbf{D}'', \qquad \mathbf{S}' \in \mathbb{R}^{Z \times N_\mathrm{s}}, \qquad \mathbf{Q}' \in \mathbb{R}^{N_\mathrm{s} \times M/2}, \qquad \mathbf{D}'' \in \mathbb{R}^{Z \times M/2}. \tag{4.176}$$

At this point, $Z \leq R$. If now $Z < N_\mathrm{s}$, $\mathbf{S}'$ has some zero columns on the right (indicated in blue in the above graphic), and the system is underdetermined. In this case, there are infinitely many solutions, which is treated in section 4.2.6. If $Z = R$, however, there are no such zero columns, $\mathbf{S}'$ is a square diagonal matrix, and a unique solution exists. This case is treated in the following section.

4.2.5 Unique solution possible

When $Z = R$, a unique solution exists, $\mathbf{S}'$ is easily invertible and the matrix $\mathbf{Q}$ can be recovered by

$$\mathbf{Q} = \mathbf{V}\mathbf{Q}' = \mathbf{V}(\mathbf{S}')^{-1}\mathbf{D}''. \tag{4.177}$$

According to eq. (4.170), the weights are the polynomials in c_s^2 where the coefficients are the entries of $\mathbf{Q}$:

$$w_i(c_\mathrm{s}^2) = \sum_{j=1}^{M/2} Q_{ij} c_\mathrm{s}^{2j}, \; i = 1, \ldots, N_\mathrm{s}. \tag{4.178}$$

A useful LB model can only be constructed with positive weights $w_i \geq 0 \, \forall \, i$. Therefore, we need to ask whether or not there exist values of c_s^2 such that all weights are indeed positive. A procedure that answers this question and at the same time has a straight-forward numerical implementation is as follows:

1. Find all positive and real roots of *all* polynomials $w_i(c_\mathrm{s}^2)$, e.g. using `numpy.roots` [92].
2. Create ordered list of roots $0 < r_0 \leq r_1 \leq \ldots$
3. Check for intervals $[r_j, r_k]$ in which all w_i are positive. Valid choices for c_s^2 lie in these intervals.

If such intervals are found, a LB model can be constructed. The weights of the non-trivial shells are expressed by polynomials inside these intervals, and the weight of the zero shell can be calculated from the normalization condition:

$$w_0 = 1 - \sum_{i=1}^{N_\mathrm{s}} w_i. \tag{4.179}$$

At the borders of these intervals, $c_\mathrm{s}^2 = r_j$, at least one of the weights is necessarily zero, and the model is reduced by the corresponding velocity shell(s).

For example, the input parameters spacial dimension $d = 3$ and a maximum tensor rank of $M = 4$ result in $R = D_\mathrm{T}(2) + D_\mathrm{T}(4) = 3$ linear equations, revealing that at most 3 velocity shells are needed. Supplying the values $c_i^2 \in \{1, 2, 4\}$ leads to a system of equations with a unique solution and one valid interval $1/3 \leq c_\mathrm{s}^2 \leq 4/9$ (table 4.1). Evaluation at the

shell number	shell size	typical vector	weight polynomial $c_\mathrm{s}^2 \in [1/3, 4/9]$	weight at $c_\mathrm{s}^2 = 1/3$	weight at $c_\mathrm{s}^2 = 4/9$
0	1	$(0,0,0)$	$1 - (15/4)c_\mathrm{s}^2 + (21/4)c_\mathrm{s}^4$	$1/3$	$10/27$
1	6	$(0,0,1)$	$(2/3)c_\mathrm{s}^2 - (3/2)c_\mathrm{s}^4$	$1/18$	0
2	12	$(0,1,1)$	$(1/4)c_\mathrm{s}^4$	$1/36$	$4/81$
3	6	$(0,0,2)$	$-(1/24)c_\mathrm{s}^2 + (1/8)c_\mathrm{s}^4$	0	$1/162$

Table 4.1: Properties of a 25-speed model in three dimensions that is isotropic up to tensor rank 4.

interval borders yields two 19-speed models, the one at $c_\mathrm{s}^2 = 1/3$ being the well-known D3Q19 model (c.f. section 3.2.2). Hence, to find the D3Q19 model with the script, one additional shell, for example $c_i^2 = 4$, must be provided, which can be eliminated again in the end due to the freedom of choice in c_s^2.

Using the script, it was even possible to discover a novel model isotropic up to tensor rank $M = 10$ with 221 velocities (see table 4.2). This has one valid interval and reduces to two 197-velocity models at the interval borders.

4.2.6 Infinitely many solutions

If, after the pruning of the matrices, the rank is less than the number of shells, $Z < N_\mathrm{s}$, the system is underdetermined and has infinitely many solutions. However it is possible to force a solution by introducing constraints such as the requirement that one or more of the weights are to be minimized. To treat such systems, the matrices $\mathbf{S}'$, $\mathbf{V}^\mathsf{T}$ and $\mathbf{D}''$ are written to the disk by `LBWeights.py` to be processed further by the secondary script `Continue.py`. This script then solves the linear optimization problem based on the equation eq. (4.176)

$$\mathbf{S}'\mathbf{V}^\mathsf{T} \cdot \mathbf{w} = \mathbf{D}'' \cdot \mathbf{c}, \tag{4.180}$$

which is subjected to the constraints

$$w_s \stackrel{!}{=} \min \forall\, s \in I, \qquad w_s \geq 0 \,\forall\, s, \qquad \sum_s w_s \leq 1, \tag{4.181}$$

shell number	shell size	typical vector	weight at $c_s^2 = 1.033691$	weight at $c_s^2 = 1.206545$
0	1	(0, 0, 0)	1.125792×10^{-1}	5.101845×10^{-2}
1	6	(0, 0, 1)	1.444892×10^{-2}	3.953745×10^{-2}
2	12	(0, 1, 1)	2.781069×10^{-2}	4.937669×10^{-3}
3	8	(1, 1, 1)	1.970138×10^{-2}	3.536908×10^{-2}
4	6	(0, 0, 2)	2.251462×10^{-2}	2.485832×10^{-2}
5	24	(1, 1, 2)	3.624508×10^{-3}	3.216647×10^{-3}
6	12	(0, 2, 2)	4.387148×10^{-3}	7.022298×10^{-3}
7	6	(0, 0, 3)	6.910281×10^{-4}	1.578096×10^{-3}
8	24	(1, 1, 3)	1.038248×10^{-3}	1.597874×10^{-3}
9	8	(2, 2, 2)	4.381319×10^{-4}	5.451840×10^{-4}
10	24	(0, 1, 4)	3.513518×10^{-5}	0
11	24	(2, 2, 3)	4.350915×10^{-5}	1.453046×10^{-4}
12	24	(1, 1, 4)	0	9.956211×10^{-5}
13	12	(0, 3, 3)	1.885761×10^{-6}	3.047305×10^{-5}
14	6	(0, 0, 5)	2.394034×10^{-6}	1.300108×10^{-5}
15	24	(0, 3, 4)	7.194413×10^{-6}	1.815117×10^{-5}

Table 4.2: Properties of a 221-velocity model in three dimensions that is isotropic up to tensor rank 10.

where I is a set containing the indexes of the weights that should be minimized. This is done using the convex optimization library `CVXPY` [93–95].

From eq. (4.180) on sees that, apart of an index set I, the script needs to be provided with a value for c_s^2. The user has the option to either supply a single value or even a whole range of c_s^2 values, which allows to scan for solutions.

4.2.7 Test mode

From the main script, it is also possible to run a testing mode that allows the user to supply a numerical solution which is then tested for consistency. The user needs to supply the spacial dimension, maximum tensor rank and the set of modulus shells. The solution is provided in form of a value for c_s^2 as well as a vector of weights $\mathbf{w}$. The weights may be given in the very general form of a linear combination

$$\mathbf{w} = \mathbf{w}_0 + \sum_{i>0} \lambda_i \mathbf{w}_i, \tag{4.182}$$

where the parameters λ_i can vary independently. The equation

$$\mathbf{A} \cdot \mathbf{w} = \mathbf{b} \tag{4.183}$$

is satisfied, if the individual components of the vector residuals

$$\boldsymbol{\Delta}_0 = \mathbf{A} \cdot \mathbf{w}_0 - \mathbf{b}, \tag{4.184}$$

$$\boldsymbol{\Delta}_i = \mathbf{A} \cdot \mathbf{w}_i, \ i > 0, \tag{4.185}$$

all vanish. Since the solution values are typically only given with finite precision, a rounding error is introduced, and the residuals will not be exactly zero even for a (approximately) correct solution. For a residual of the form

$$\Delta_i = \sum_j A_{ij} w_j - b_i \tag{4.186}$$

we can however derive an upper bound for the residuals that is adjusted to the magnitude of the rounding errors in the provided solution. If the solution is given with n relevant digits, we can estimate the relative accuracy by $\varepsilon = 10^{-n+1}$. The maximum deviation of a weight w_j from the real value is thus $\delta w_j = \varepsilon w_j$. Similarly, the supplied value c_s^2 might deviate from the real value by as much as $\varepsilon c_\mathrm{s}^2$. Recalling that $b_i = (c_\mathrm{s}^2)^{m_i/2}$, the error in b_i is then

$$\delta b = \varepsilon \frac{m_i}{2} \left(c_\mathrm{s}^2\right)^{m_i/2} = \varepsilon \frac{m_i}{2} b_i. \tag{4.187}$$

From there, the total error on the residual can be estimated by Gaussian error propagation

$$\delta\Delta_i = \sqrt{\sum_j \left(A_{ij}\,\delta w_j\right)^2 + \left(\delta b_i\right)^2} = \varepsilon\sqrt{\sum_j \left(A_{ij} w_j\right)^2 + \left(\frac{m_i}{2} b_i\right)^2}. \tag{4.188}$$

Of course, this also applies to the residuals of the form of 4.185; in this case we can just set $\mathbf{b} = 0$. If for all residuals $|\Delta_i| < \delta\Delta_i$, the provided solution is considered to be correct within the numerical bounds. Using the test mode, several LB models from the literature could be verified, among others the models derived by X. Shan via Gaussian quadratures in ref. [96].

The functionality of the `LBWeights.py` script is summarized in the flow diagram fig. 4.3.

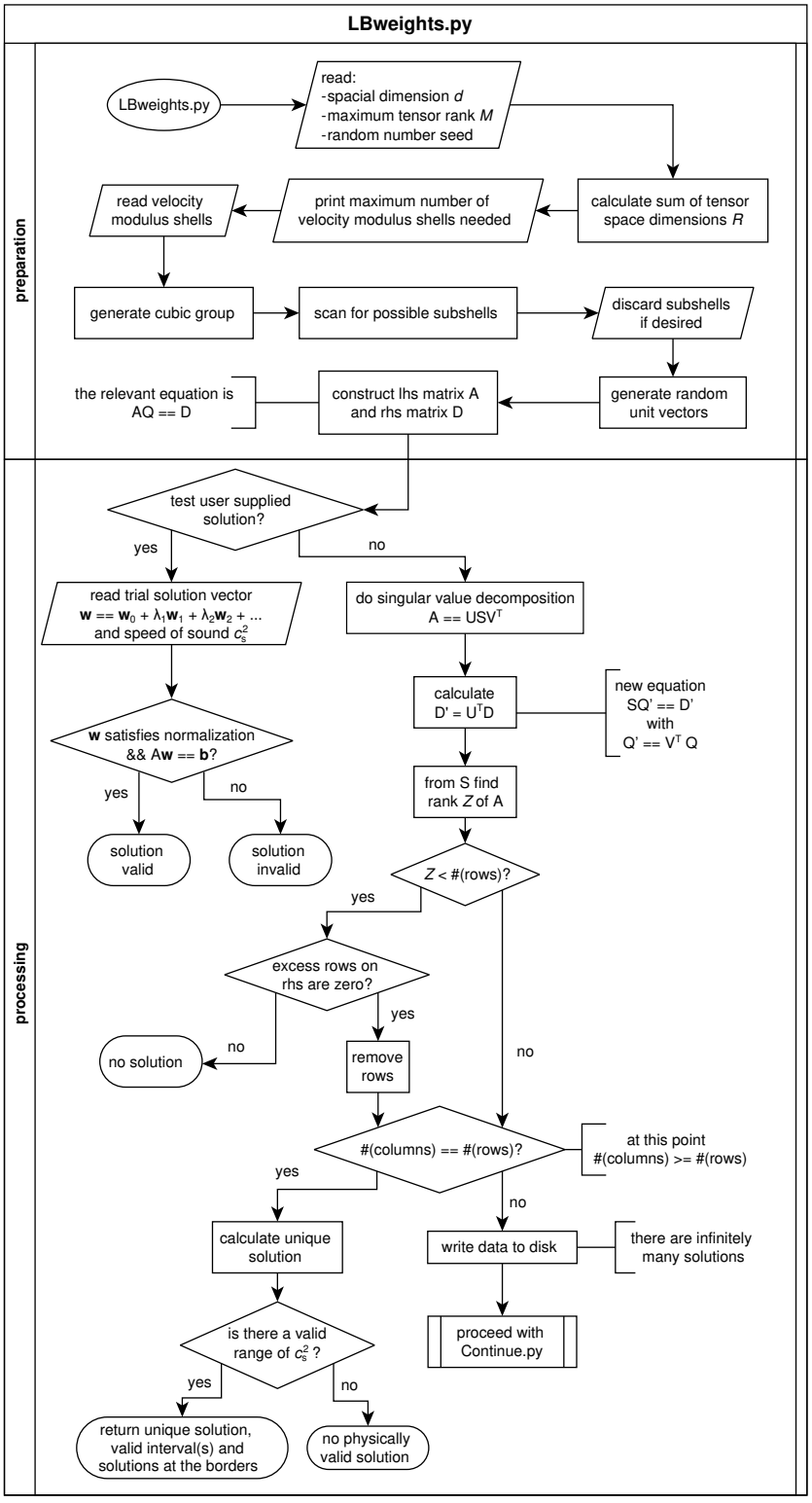


Figure 4.3: Flow diagram of the Python script `LBWeights.py`.

Chapter 5

Numerical Results

This chapter compiles some of the results obtained from numerical simulations of pure MD systems as well as results from the coupled LB/MD scheme described in section 3.2.4.

In section 5.1, benchmarks obtained by simulating the coupled model for a typical particle density on a varying number of processors are shown. The results can be used to estimate a reasonable number of processors for the simulation of a given size system.

Since some of the numerical quench experiments are performed in two-dimensional systems, one should have an idea about the critical attraction strength of the collapse transition. This is estimated in section 5.2 by simulating different chain lengths at varying interaction strength and exploiting the universal scaling in the theta solvent.

As has been described in section 2.3, a possible model for the non-bonded interactions in a polymer system, that also allows for the description of phase separation, can be derived from Van der Waals theory. Several systems at various densities are simulated in order to obtain an equation of state. The results are then compared to the theoretical Van der Waals model and first estimates for its parameters are made in section 5.3.

In section 5.4, the dynamic structure factor is evaluated for quenches of two- and three-dimensional systems. By following the structure factor peak over time, dynamic scaling plots for the characteristic coarsening length scales are produced. Comparisons are made with numerical results from a two-dimensional continuum model that has been simulated by a collaborating group. By suitable scaling and shifting of the peaks and comparison to simple fluids, it can be confirmed that both models indeed produce non-standard phase separation dynamics.

How Minkowski functionals can be used to determine geometrical properties of the polymer morphologies during phase separation is shown in section 5.5. To do so, the configurations are interpolated to a binary lattice from which quantities like volume, area, curvature, and

Euler characteristic are computed. The initial dynamic behavior is especially sensitive to the choice of lattice constant and the threshold value for the density at which a lattice site is considered to be occupied. In the limit of long times, the behavior becomes universal, allowing for the estimation of dynamic exponents.

5.1 Benchmarks

In order to estimate the performance and parallel scalability of the LB/MD coupling scheme as it is implemented in the simulation package ESPResSo++ [72], some benchmarks were performed on the high-performance cluster 'DRACO', which is part of the Max Planck Computing & Data Facility (MPCDF) in Garching.

The architecture of DRACO is as follows:

- there are ~ 880 nodes equipped with Intel 'Haswell' Xeon E5 processors
- one node has 32 cores with 2.3 GHz
- hyperthreading is possible running two threads on one core
- the maximum memory per node is 128 GB

Furthermore, 106 of the nodes are equipped with graphics accelerator cards (two PNY GTX980 GPUs each), and there are 64 Intel 'Broadwell' nodes with 40 cores and 256 GB of memory; however, they were not used for any simulations presented here.

5.1.1 Parallel scalability, strong scaling

The strong scaling efficiency measures the parallel performance with fixed problem size, i.e. fixed number of particles, and fixed number of integrations. It can be calculated by

$$e_{\mathrm{ss}} = \frac{1}{N}\frac{T_1}{T_N}, \tag{5.1}$$

where T_1 is the time needed to complete the task on one processing unit and T_N is the time needed to complete the same task on N processing units. Equation (5.1) holds for a fixed number of integrations. Here, however, we measure the number of integrations n_N and the time needed to perform these integrations on N processing units, which is $\tilde{T}_N$. Therefore $\tilde{T}_N$ are calculated for a variable problem size. This then defines the rates $R_N := n_N/\tilde{T}_N$. The time needed by N processing units to perform a fixed number of n_0 integrations is then given by $T_N = n_0/R_N$. This allows to express the strong scaling efficiency by the rates:

$$e_{\mathrm{ss}} = \frac{1}{N}\frac{R_N}{R_1}. \tag{5.2}$$

The benchmarks are performed for a three-dimensional system of linear size $L = 128$ containing $N = 4784$ chains with $M = 64$ beads each. This corresponds to a particle density of $c \approx 0.146$. These numbers were chosen to match the volume fractions of macroscopic

simulations that where available for comparison at the time. The particles are coupled to a LB fluid with a grid with lattice constant one. Hence, there are $N_{\text{part}} \approx 3.1 \times 10^5$ particles in total and $128^3 \approx 2.1 \times 10^6$ Lattice Boltzmann sites. The system was run for a certain time, counting the number of MD integrations performed within that time, and varying the number of nodes used. Note that this is not a rigorous approach to measure the efficiency since only one run per size is performed, and the smallest computation unit is one node with 32 processors rather than one single processor.

As unit system $\sigma_{\text{LJ}} = \varepsilon_{\text{LJ}} = 1$ was chosen. With $r_{\text{c}} = 1.5$, this fixes the LJC parameters $\alpha = 3.1730728678$ and $\beta = -0.85622864544$. For the FENE potential (section 3.1.4), a strength of $K_{\text{F}} = 30$ and a maximum extension of $r_{\text{max}} = 1.5$ was used. A Langevin thermostat (section 3.1.3) fixes the temperature at $T = 1$. The skin thickness is $r_{\text{s}} = 0.3$ and the timestep is $\delta t_{\text{MD}} = 0.005$. Configurations are written to the disk every 2000 integrations. For the LB fluid, both shear and bulk viscosity are set to 3; it is coupled to the MD particles with a friction constant of 20. A costly LB step is performed only every 10 MD steps, which still gives good accuracy while increasing efficiency [83]. All simulations are performed with an attraction strength of $\phi = 0$ i.e. at good solvent conditions. At this point we should note that in actual quench simulations the system coarsens over time, which leads to worse load balancing and a gradual decrease in performance.

A significant amount of the time goes into reading the initial configuration from disk in the beginning. However, we find that this time roughly decreases with a power law with

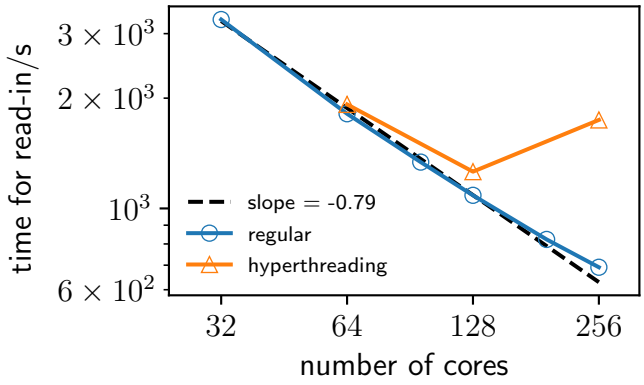


Figure 5.1: Double logarithmic plot of the time needed to read in the initial configuration for varying number of processing units. Three runs have been performed with hyperthreading running two threads per core.

exponent -0.79 with an increasing number of cores (see fig. 5.1). Hyperthreading is not useful in this case and can even increase the time needed to read the initial configuration.

The number of integrations per second is found to increase approximately linearly with

the number of cores with a slope of about 0.056 (see fig. 5.2). This linear increase is not sustained when using hyperthreading; then the rate even decreases again for a large number of processors. We conclude that hyperthreading is not useful and even disadvantageous

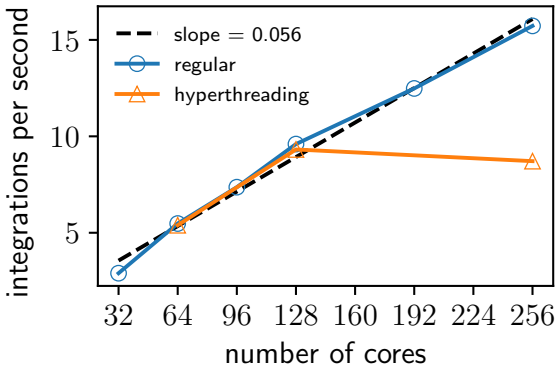


Figure 5.2: Number of integrations per second performed by a varying number of cores in single and hyperthreading mode.

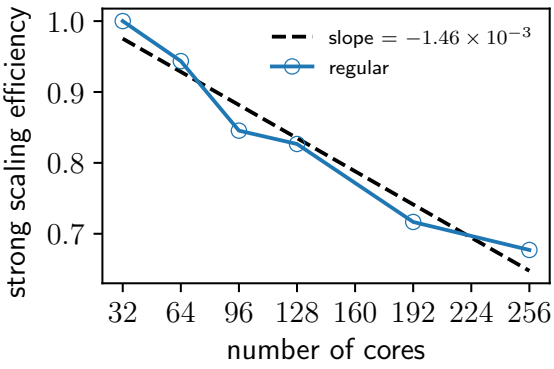


Figure 5.3: Strong scaling efficiency as defined in eq. (5.3) for varying number of cores.

for the present problem.

Recall, that the smallest processing unit used is one node with 32 cores, hence we use a modified formula for the strong scaling efficiency,

$$e_{\mathrm{ss}} = \frac{32}{N}\frac{R_N}{R_{32}}, \tag{5.3}$$

which is at 100% for 32 cores. Equation (5.3) for single threaded runs is plotted in fig. 5.3. The efficiency decreases approximately linearly with the number of cores with a slope of about -1.5×10^{-3}. At 256 cores there is roughly one processor per 1200 particles or 8200 LB sites. Hence, the problem scales well even at a large number of processors.

The simulations presented in the following sections have been run with no less than 5000 particles or roughly 33000 LB nodes per processors, which corresponds to 64 or fewer processors, for the problem size used in these benchmarks. Therefore efficiency can be expected to be within a reasonable range.

5.2 Theta-collapse transition in two dimensions

As introduced in section 3.1.5, the Lennard-Jones Cosine (LJC) potential is used to model the pairwise non-bonded interaction between beads. It has the form

$$U_{\mathrm{LJC}}(r) = \begin{cases} 4\varepsilon_{\mathrm{LJ}}\left[\left(\frac{\sigma_{\mathrm{LJ}}}{r}\right)^{12} - \left(\frac{\sigma_{\mathrm{LJ}}}{r}\right)^{6} + \frac{1}{4}\right] - \varepsilon_{\mathrm{LJ}}\phi & r \leq 2^{1/6}\sigma_{\mathrm{LJ}} \\ \frac{1}{2}\phi\varepsilon_{\mathrm{LJ}}\left(\cos\left[\alpha\left(\frac{r}{\sigma_{\mathrm{LJ}}}\right)^{2} + \beta\right] - 1\right) & 2^{1/6}\sigma_{\mathrm{LJ}} < r \leq r_{\mathrm{c}} \\ 0 & \text{else} \end{cases}, \tag{5.4}$$

where the parameter ϕ controls the strength of attraction, and thereby implicitly controls the quality of the solvent. In this section, the attraction strength corresponding to the theta solvent in two dimensions is estimated by numerical simulations.

We follow a similar approach as Steinhauser in ref. [61] where the theta point of three-dimensional polymer solutions has been determined to be $\phi_\Theta = 0.65 \pm 0.02$. Here it is exploited that for chains with a large number of monomers M, the chain size scales like a random walk with $M - 1$ steps at the theta temperature:

$$\left\langle R_{\mathrm{g}}^2 \right\rangle \propto \left\langle R_{\mathrm{e}}^2 \right\rangle \propto M - 1; \tag{5.5}$$

also see eq. (2.2) in section 2.1 about polymer solutions. Therefore, the squared chain size divided by the number of bonds $(M - 1)$ takes a fixed value at the theta point apart from corrections due to the finite size of the chains.

To estimate the theta point, multiple simulations for different chain lengths and different interaction strengths have been performed. The number of chains N in a system was chosen such that the total number of particles is fixed at 32768 for all systems (c.f. table 5.1). For the potential confining the particles in the $z = 0$ plane (c.f. section 3.1.6),

M	N
256	128
512	64
1024	32
2048	16

Table 5.1: Chain length M versus the number of chains in a system N.

a strength of $U_0 = 512$ was chosen. As only static properties are of interest here, there is no hydrodynamic interaction with the solvent needed and thus no coupling to LB is applied. Interchain interaction was turned of as well so that each chain can be viewed as single, independent molecule. The friction constant of the Langevin thermostat is set to $\zeta_{\mathrm{L}} = 1$. Apart from that, the same basic parameters as in section 5.1 were used.

The simulations were performed at values of ϕ ranging from $\phi = 1.0$ to $\phi = 2.0$ in steps of 0.2 resp. 0.1. For each chain length and attraction strength, an independent initial configuration was set up and equilibrated. The average values of the squared gyration radius $\left\langle R_g^2 \right\rangle$ divided by $M - 1$ are plotted in fig. 5.4; the error bars give the standard deviation.

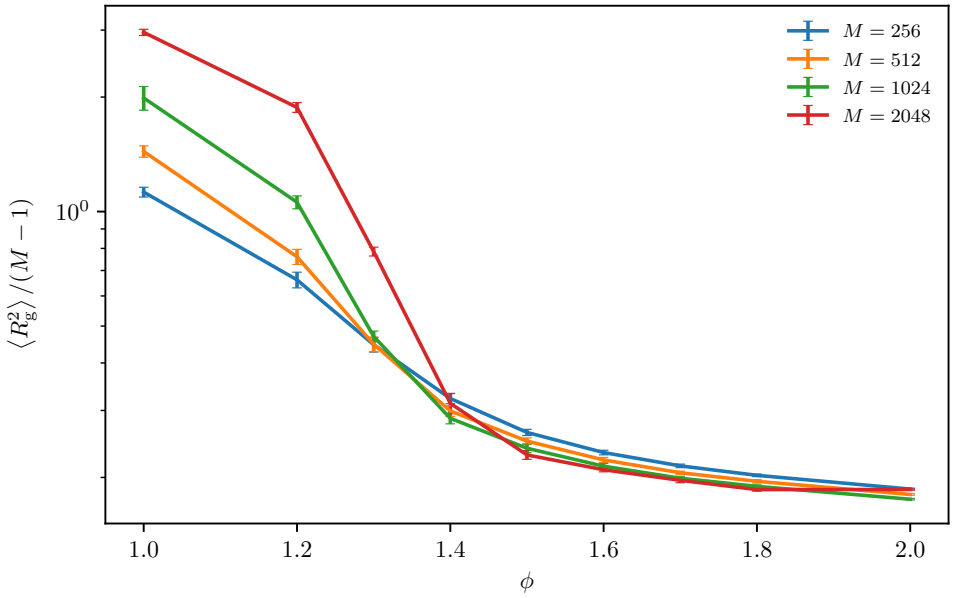


Figure 5.4: Logarithmic plot of $R_g^2/(M-1)$ over the attraction strength ϕ for different chain lengths. The intersection points give an estimate for the theta transition.

The first intersections of the curves of lower chain length occur at about $\phi \approx 1.3$, while the curve for $M = 2048$ intersects the curve for $M = 1024$ at about $\phi \approx 1.45$. As the finite-size effects decrease with increasing chain lengths, the data from larger chains provides a more accurate estimate of the theta point, which we expect at around $\phi \approx 1.5$. Of course, this is only a very rough estimate for the theta point, which is defined in the limit $M \to \infty$. For more exact results, ideally, longer chains and a finer resolution in ϕ should be simulated. Then the intersection points can be plotted and extrapolated to infinite chain length. Nevertheless, the estimated value determined here seems appropriate to justify the choice of $\phi = 2$ when quenching the system from good to poor solvent quality in the experiments presented in section 5.4, which are of course likewise performed at finite chain lengths. Note that in three dimensions the chain conformation has a fractal dimension of two at the theta point while it takes a value of three in the compact phase. In two dimensions the fractal dimension is two in both cases making ideal and compact scaling behavior identical.

5.3 Van der Waals equation of state

In the future it will be interesting to compare numerical simulations of the continuum model for viscoelastic phase separation derived in section 4.1 with simulations preformed with the LB/MD coupling scheme (mesoscopic model, c.f. section 3.2.4). To ensure best comparability, the parameters of both models should be calibrated to each other. A good candidate for the equation of state is the Van der Waals pressure which was derived in section 2.3:

$$P = \frac{Nk_{\mathrm{B}}T}{MN}\frac{n}{1-Bn} - An^2, \tag{5.6}$$

hence the interaction parameter A as well as the volume parameter B are among the parameters that one would like to estimate for the mesoscopic model.

In this section, an estimate for these parameters is obtained by fitting the Van der Waals equation of state to data obtained from numerical simulations with the mesoscopic model. Note that only the partial pressure of the polymer phase is of interest here; hence the simulations can be performed without coupling to the LB-ideal gas. The simulated systems contain $N = 16384$ chains of length $M = 8$ at temperature $T = 1$. As the systems are three dimensional, no confinement potential is applied. Otherwise, the parameters are identical to the ones used in section 5.2. Simulations have been performed at different box volumes keeping the number of particles fixed. For each run, the average pressure and its standard deviation were calculated.

In fig. 5.5, the Van der Waals equation of state is plotted for these parameters at $B = 1$ for different values of the interaction parameter A. For low values of A, there is only one phase, while for high values of A, there is a dense phase at low volume (high density) and a gaseous phase at high volume (low density). However, Van der Waals theory fails in the region in between these phases where phase transition occurs. There, one would expect the pressure curve to be constant and not an oscillation as it occurs in fig. 5.5 at $A = 0.45$. This can be mended by the Maxwell construction, where a horizontal line is spliced into the curve in such a way that the two resulting areas above and below the line are of opposite sign but equal magnitude. Because fluctuations increase massively in the phase transition region, it takes significantly more data points to obtain a reasonable statistical accuracy. Therefore, the benefits of the Maxwell construction are slim with the present quality of data, which is why the points in the phase transition region are simply omitted when fitting the equation of state.

In the left side of fig. 5.6, the results for a purely repulsive non-bonded interaction, i.e. a Lennard-Jones Cosine attraction strength of $\phi = 0$, are plotted. The system is in the one-phase region and a least-squares fit of the equation of state results in the parameters $A = -2.0$ and $B = 0.94$.

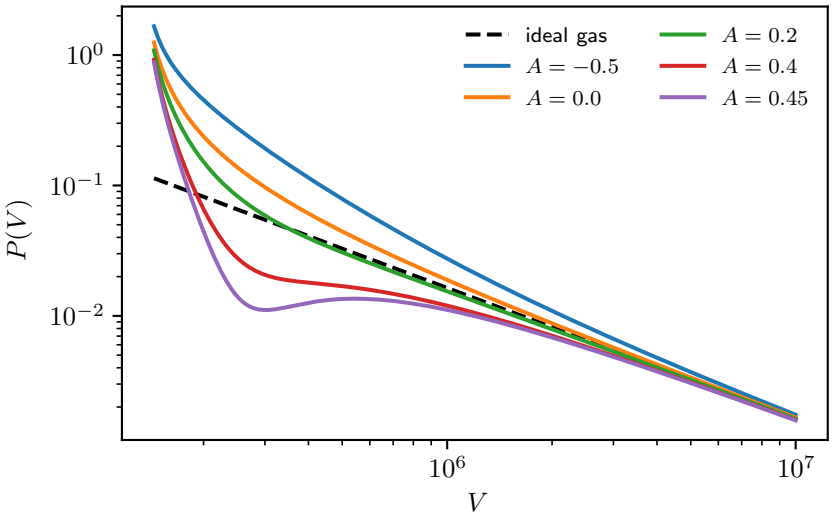


Figure 5.5: Van der Waals pressure equation of state eq. (5.6) with $B = 1$ for a system of $N = 16384$ chains of length $M = 8$ and with temperature $T = 1$ plotted at different interaction parameters A. The ideal gas is given as a reference.

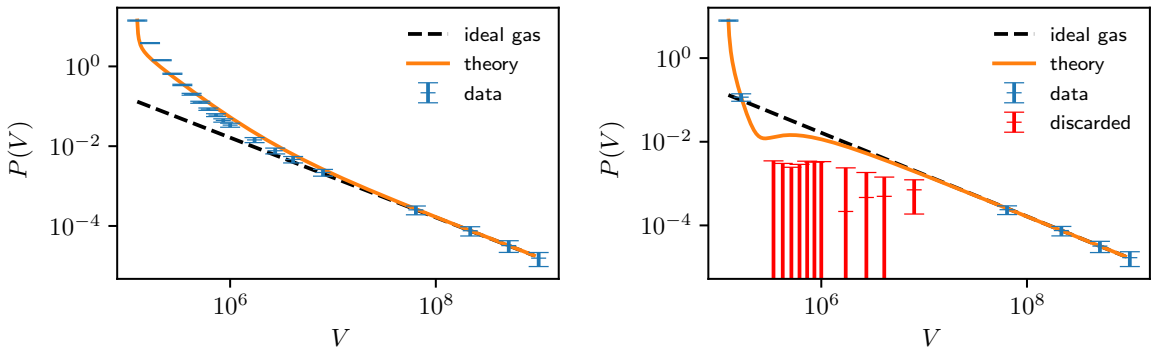


Figure 5.6: Pressure curve at $\phi = 0$ (left) and $\phi = 1$ (right) as well as the fitted Van der Waals equation of state. The equation of state of the ideal gas is given as reference. Points marked red have been discarded.

The right side of fig. 5.6 shows the results for $\phi = 1$, data points in the transition region are marked red and are not taken into account in the fitting procedure. In the intermediate region, the error bars are large, because the fluctuations massively increase at the phase transition. The fit results in the parameters $A = 0.42$ and $B = 0.94$.

5.4 Dynamic structure factor

In order to study the coarsening behavior after a quench to a poor solvent quality, the dynamic structure factor (see section 2.4 eq. (2.93)) was calculated for two and three-dimensional systems. In the case of the two-dimensional system, the results are compared to data obtained from a continuum model, which was simulated by a collaborating group.

To calculate the dynamic (time-dependent) structure factor defined by

$$S(q,t) = \frac{1}{N}\left\langle |\tilde{\rho}(\mathbf{q},t)|^2 \right\rangle, \qquad \tilde{\rho}(\mathbf{q},t) = \int \mathrm{d}^d\mathbf{r}\, \rho(\mathbf{r},t)\exp(-\mathbf{i}\mathbf{q}\cdot\mathbf{r}), \tag{5.7}$$

the polymer density is first mapped onto a discrete lattice with lattice constant a. The

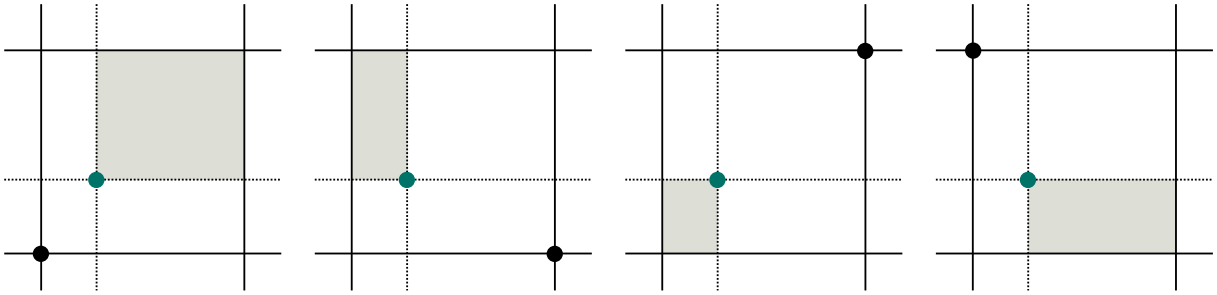


Figure 5.7: The density at a lattice site (black dot) is calculated by distributing mass contributions of enclosed particles (green dot). The magnitude of the contribution is proportional to the gray shaded area and depends on how close the particle is to the respective lattice site.

way in which this mapping is carried out is illustrated for a two-dimensional setting in fig. 5.7. The particles in the system are iterated, and for each particle (green dot in the figure), the lattice indexes of the lattice site to the bottom left are determined. Then, the fraction of the particle mass that corresponds to the gray shaded area opposite of the current lattice site is added to the mass at the current lattice site. For example, in two dimensions, if the lattice site is at the position (x, y) and a particle with mass m is at the position $(x + \delta x, y + \delta y)$, the density contribution of the particle the lattice site is given by

$$\delta\rho(x,y) = \frac{m}{a^4}(a-\delta x)(a-\delta y), \tag{5.8}$$

where a is the lattice constant. The same is done for all other remaining lattice sites surrounding the particle. In higher dimensions, the mapping works analogously.

From the resulting lattice density, the structure factor is calculated via Fast Fourier Transform using the C++ library `FFTW3` [97]. The lattice constant a has to be chosen small enough to provide sufficient resolution. It is found that for a lattice constant smaller than

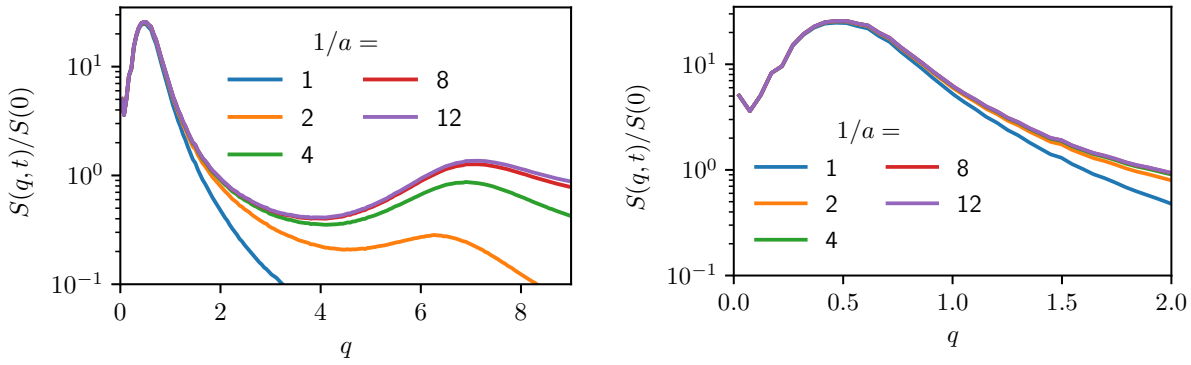


Figure 5.8: Structure factor peak at different lattice constants a of the underlying density lattice. The system contains $N = 4784$ chains of length $M = 64$ in a three-dimensional box with linear size $L = 128$.

$a = 1/8$ the structure factor curves are reasonably converged, see fig. 5.8. This is especially true for the left peak, which is of primary interest here. For the results presented below a lattice constant of $a = 1/8$ or lower was used.

5.4.1 In two dimensions

In this section, the structure factor and relating results from two-dimensional mesoscopic simulations are presented and compared to results from a continuum model that was simulated by Aaron Brunk at the Johannes Gutenberg University in Mainz. The results presented here are partly published in reference [98].

The mesoscopic system consists of 1024 polymer chains with 128 beads of mass $m = 1$ each, in a box of size $512 \times 512 \times 4$. For the confinement of the particles in the xy-plane, a potential strength of $U_0 = 512$ (see section 3.1.6) was applied. Otherwise, the same basic parameters as in section 5.1 were used. Four independent systems were equilibrated with Molecular Dynamics (MD) exclusively at first. Then, the systems were coupled to a Lattice Boltzmann (LB) fluid and equilibrated further. Once the coupled systems are equilibrated, phase separation is induced by introducing an attractive well with a depth of $\phi = 2$ in the Lennard-Jones Cosine potential, quenching the system deep into the two-phase regime.

The time evolution of the configuration one of the systems after quenching is given in fig. 5.9. After the attractive interaction is introduced, small holes begin to form in the polymer matrix. As time progresses, these solvent holes grow in size leading to a network-like structure in the polymer phase. The size of the solvent holes is a characteristic length scale in the system, and, as has been motivated in section 2.4, the dynamic structure factor is an excellent tool for estimating it.

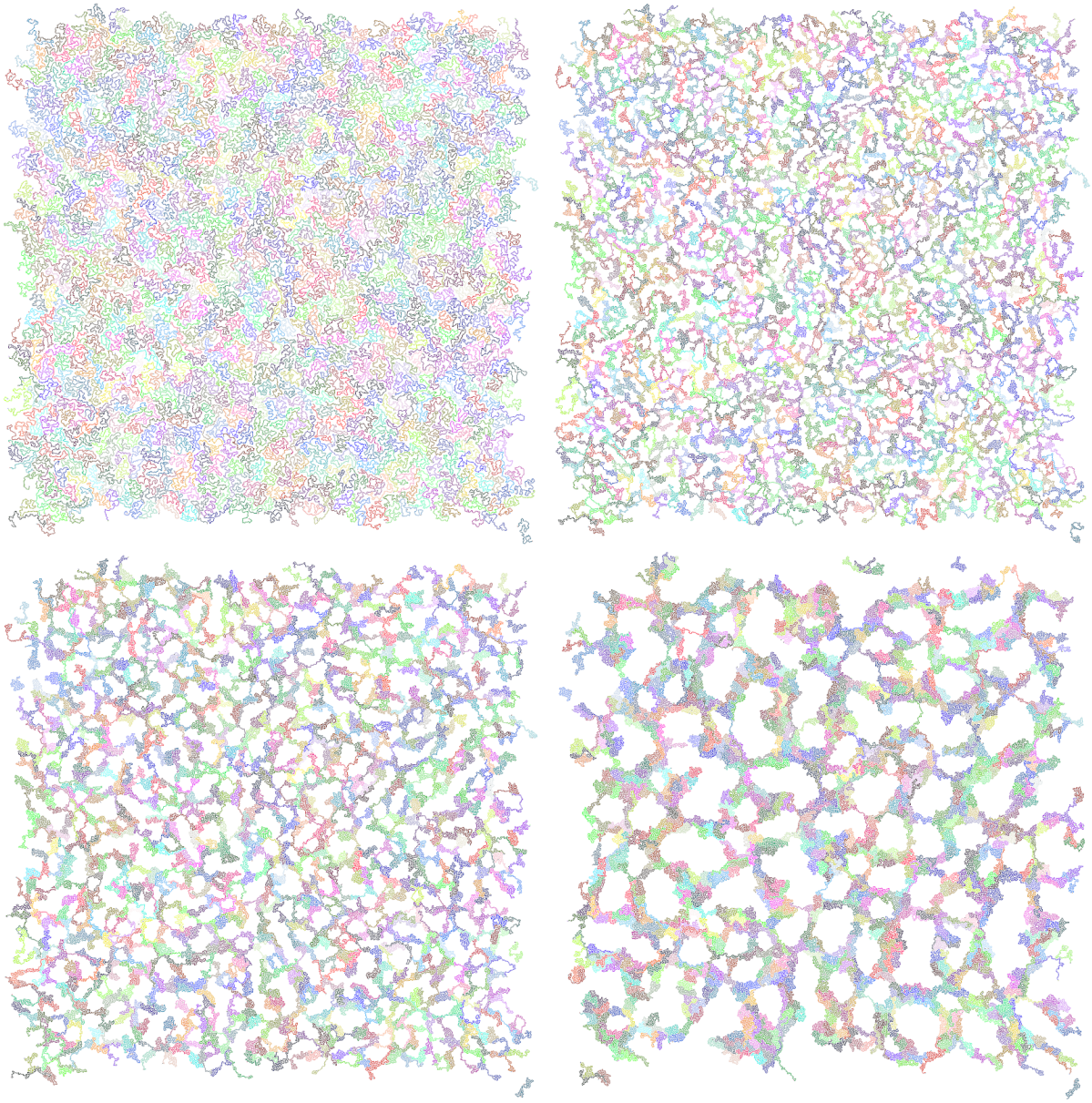


Figure 5.9: Time evolution after quenching a two-dimensional system. Different colors are used to better distinguish individual chains. The configurations are taken at $t = 0$, 1500, 7900 and 87500 (left to right, top to bottom).

The time-dependent structure factor is calculated for all four systems, and the curves are averaged. A plot of the average structure factor curves at different points in time after quenching is given in fig. 5.10. At $t = 0$, the system is close to homogeneous, and there

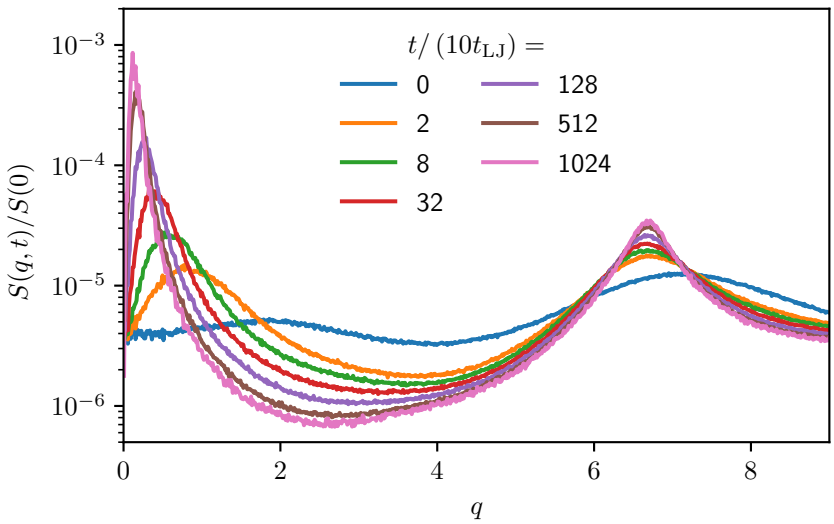


Figure 5.10: Structure factor for different points in time after the quench. The curves are normalized by the time-constant value $S(0,t) =: S(0)$, which is proportional to the total density.

is little structure. In the structure factor, there is a peak at wavenumber $q \approx 7$, which corresponds to the typical interparticle distance at equilibrium. As the system coarsens, this peak moves towards smaller wavenumbers, and the maximum scattering intensity $S(q_{\max})$ increases, a behavior which is explained in section 2.4.

The results from the mesoscopic model are compared to results from the continuum model based on the equations

$$\dot{\phi} = \boldsymbol{\nabla} \cdot \Big(n^2(\phi) \boldsymbol{\nabla} \mu - n(\phi) \boldsymbol{\nabla} (A(\phi)\pi) \Big), \tag{5.9}$$

$$\dot{\pi} = -h_1(\phi)\pi + A(\phi) \boldsymbol{\nabla} \cdot (\boldsymbol{\nabla}(A(\phi)\pi) - n(\phi)\boldsymbol{\nabla}\mu) + \varepsilon_1 \nabla^2 \pi \tag{5.10}$$

$$\dot{\mathbf{v}} = \boldsymbol{\nabla} \cdot (\eta(\phi)\mathbf{D}) - \boldsymbol{\nabla} P + \boldsymbol{\nabla} \cdot \mathbf{T} + \mu \boldsymbol{\nabla} \phi, \tag{5.11}$$

$$\overset{\triangledown}{\mathbf{C}} = -h_2(\phi) B (\operatorname{tr} \mathbf{C})(\operatorname{tr}(\mathbf{C})\mathbf{C} - \mathbb{1}) + \varepsilon_2 \nabla^2 \mathbf{C}, \tag{5.12}$$

$$\mu = -c_0 \nabla^2 \phi + \frac{\partial f(\phi)}{\partial \phi}, \tag{5.13}$$

$$\boldsymbol{\nabla} \cdot \mathbf{v} = 0, \quad \mathbf{T} = \operatorname{tr}(\mathbf{C})\mathbf{C}. \tag{5.14}$$

Here, ϕ is the polymer volume fraction, π is a putative bulk stress, $\mathbf{v}$ is the mass average velocity, $\mathbf{D}$ is the deformation rate tensor defined in eq. (2.158), $\mathbf{C}$ is the conformation

tensor and f is the free energy density. The model is simulated using finite element methods (FEMs). For more information on the model as well as on the parameters used, the reader is referred to the references [98, 99].

Structure factors obtained both by the mesoscopic and macroscopic model are given in fig. 5.11. In the first row, the unnormalized peaks are plotted at different times; both approaches clearly show the coarsening and the growth in scattering intensities. The second row features the same data, with the difference that q is normalized by the wavenumber of maximum scattering intensity $q_{\max}$ and $S(q/q_{\max}, t)/S(q_{\max}, t)$ is plotted, fixing the position of the peak at the point $(1, 1)$ in the plot. For simple fluids, the curves should collapse to a master curve (see section 2.4.2). However, both approaches show significant variation at successive time steps. The same plots for the non-viscoelastic version of the models are given in the third row. In case of the mesoscopic model, the connectivity interaction (FENE potential) was omitted. This was compensated by setting a deeper quench depth of $\phi = 3$. In case of the macroscopic model, elastic stresses, i.e. π and $\mathbf{C}$ and their respective coupling terms, were removed. In this case, the collapse works much better, indicating that both models indeed feature non-standard phase separation dynamics. This particular comparison is motivated by ref. [100] where a related viscoelastic continuum model was studied.

In the next step, the dynamic scaling behavior is examined by tracking the wavenumber of maximum scattering intensity $q_{\max}$ with time. A double logarithmic plot of $q_{\max}(t)$ is given for both models in fig. 5.12. In the initial regime, both models are close to the Lifshitz-Slyozov growth law $q_{\max} \propto t^{-1/3}$ (c.f. section 2.4.3). For $t > 10^3 t/t_{\mathrm{FE}}$ the macroscopic model transitions to a different regime with slower growth rate. This change of regimes is also seen in the plot of the maximum scattering intensity in fig. 5.13. Here it also becomes apparent that the scale of the scattering intensity differs for both approaches. The value $S(q = 0)$ is proportional to the total density, it is off scale and therefore not included in the plots in figs. 5.10 and 5.11. The asymptotic value $\lim_{q\to 0} S(q)$ however is related to the compressibility. As the equation of state is not matched for both models, their compressibility differs, which results in the mismatch of scales in the scattering intensity.

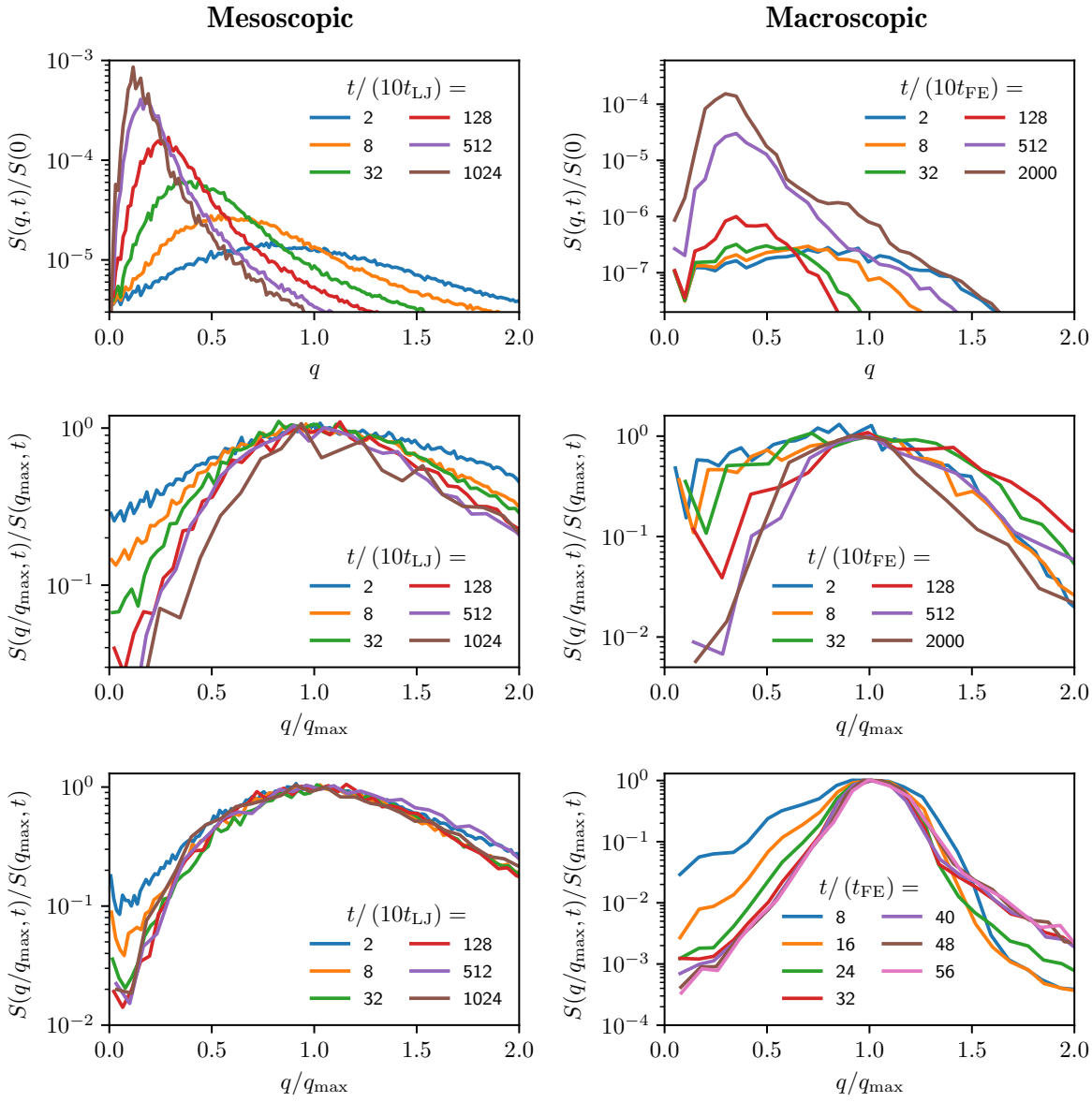


Figure 5.11: Time evolution of the structure factor (left panel: mesoscopic simulations, right panel: macroscopic simulations). The first row shows the normalized scattering intensity $S(q,t)/S(q=0)$ vs. the wavenumber q. In the second row, the scattering intensity has been normalized by its peak value $S(q_{\max})$ and the wavenumber by $q_{\max}$. For comparison, we show another normalized plot for simple fluids without any elastic effects in the third row.

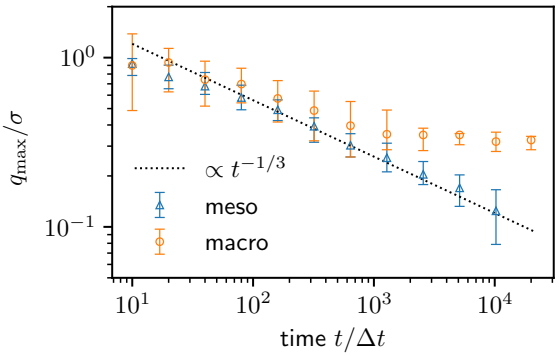


Figure 5.12: Time evolution of the peak wavenumber $q_{\max}$. Triangles indicate results from the mesoscopic model ($\Delta t = t_{\mathrm{LJ}}, \sigma = \sigma_{\mathrm{LJ}}$), while circles represent the macroscopic model ($\Delta t = t_{FE}$). The Lifshitz-Slyozov growth law $q_{\max} \propto t^{-1/3}$ is given for reference.

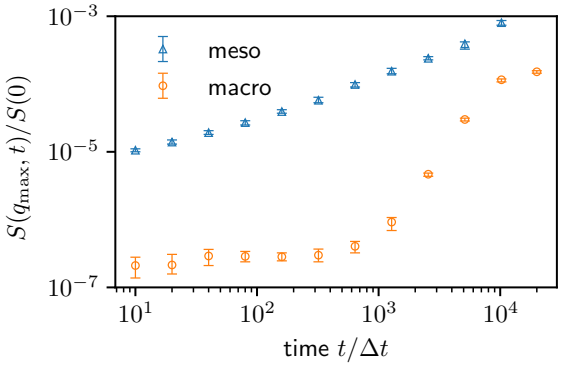


Figure 5.13: Normalized peak-scattering intensity $S(q_{\max}, t)/S(0)$ for mesoscopic ($\Delta t = t_{\mathrm{LJ}}$) and macroscopic simulations ($\Delta t = t_{\mathrm{FE}}$) respectively. The difference in scale is due to discrepancies in the equation of state of both approaches.

5.4.2 In three dimensions

As hydrodynamics and polymer physics in two dimensions both differ significantly from the corresponding physics in three dimensions, the focus should be put on the numerical simulation of three-dimensional systems in the long run.

Several three-dimensional systems at different particle concentrations were simulated. Of particular interest are the results obtained from a model with 4784 chains of length 64 in a three-dimensional box of linear size $L = 128$. Aside from a quench depth of $\phi = 1$, the same parameters as for the two-dimensional system above were used. Eight independent systems have been equilibrated analogously, and average structure factor curves were computed.

Visualization is not as straightforward in three dimensions as it is in two dimensions. One possibility is taking slices of thickness b, $\{\mathbf{r} = (x, y, z) : |z - L/2| \leq b/2\}$, from the system and plot their projection onto a plane. This is shown at different times after the quench in fig. 5.14. Analogously to the second and third row in fig. 5.11, the structure factor peaks are shifted by the peak wavenumber and normalized by the maximum scattering intensity, see fig. 5.15. Especially in early times, the curves deviate significantly, which is a strong indication for non-standard phase separation. The resolution of the data is not as good as for the two-dimensional system; this is because the smallest reasonable bin size for the q values is limited by the linear system size by $2\pi/L$, and L is much smaller for the three-dimensional data.

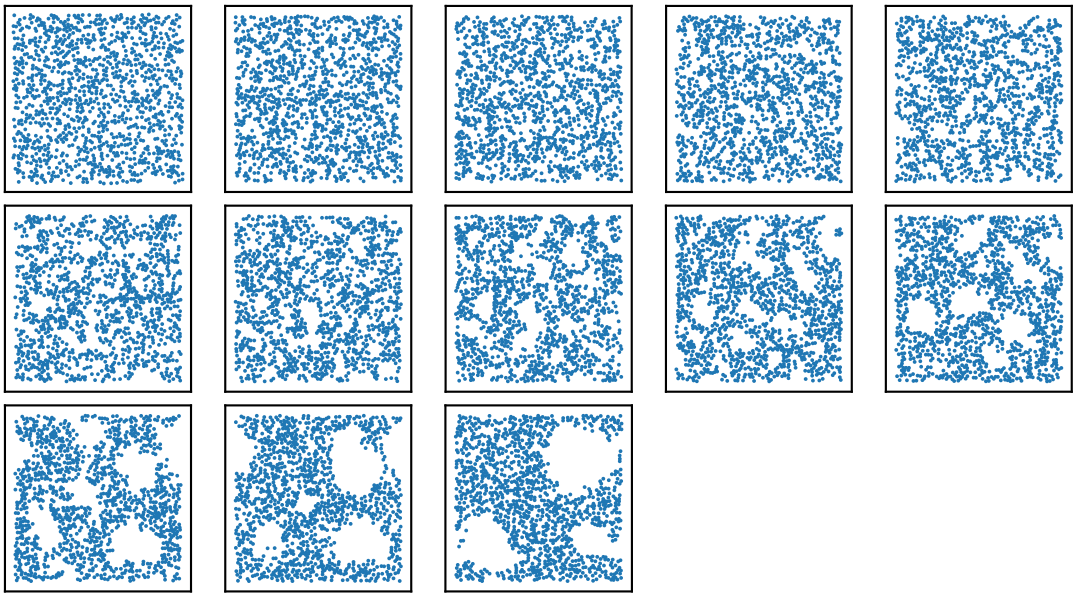


Figure 5.14: Slices of thickness 2 of the configurations plotted at times $t = 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ in units of $10t_{\mathrm{LJ}}$.

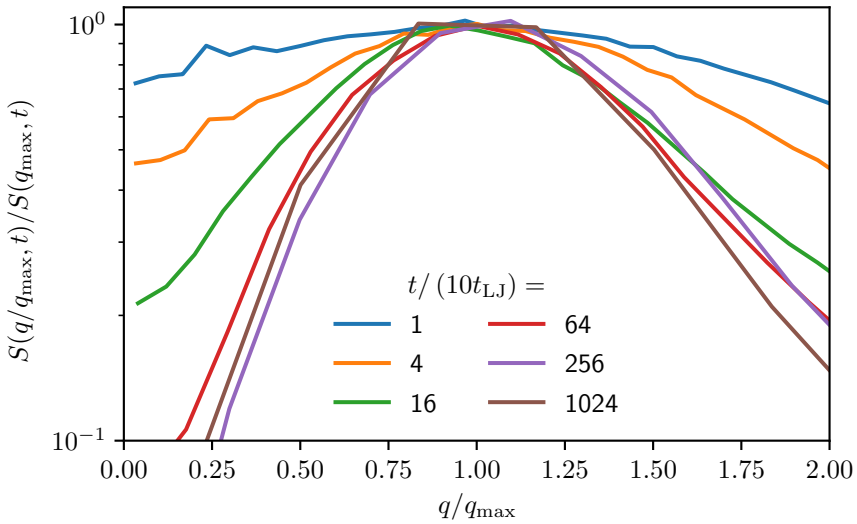


Figure 5.15: Structure factor peak shifted by $q_{\max}$ and normalized by $S(q_{\max})$.

The time evolution of the peak wavenumber $q_{\max}(t)$ is plotted in fig. 5.16. Interestingly,

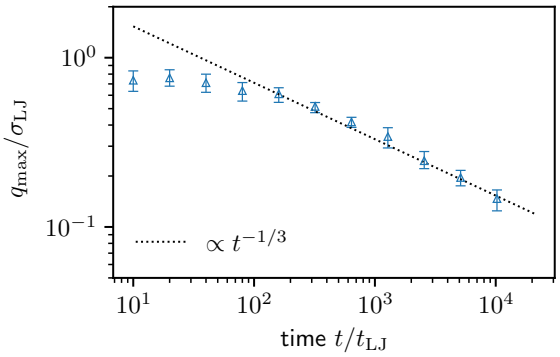


Figure 5.16: Time evolution of peak wavenumber $q_{\max}$. The Lifshitz-Slyozov growth law $q_{\max} \propto t^{-1/3}$ is given for reference.

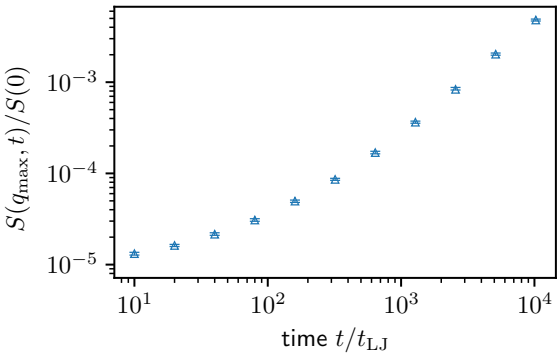


Figure 5.17: Normalized peak-scattering intensity $S(q_{\max}, t)/S(0)$.

the coarsening is very slow in the initial phase and only at later times the Lifshitz-Slyozov growth law is approached.

Similar observations have been made experimentally by Tanaka [11]; he calls this initial regime in which the solvent slowly starts to push in between the polymer molecules the 'frozen regime'. In ref. [100] by Tanaka and Araki, the regime following the frozen regime is called the 'elastic regime', which is dominated by elastic interactions in the developing polymer network structure. For entangled polymer solutions, it is said to have a non-universal dynamic scaling exponent that depends on parameters such as the quench depth and the molecular weight of the viscoelastic component. This is not seen here, as the polymers are not entangled; instead, scaling follows the standard Lifshitz-Slyozov law. Also, they observe a 'volume shrinking' of the viscoelastic phase in the late stage, which results in a decrease of the scattering intensity. Such a decrease is not seen here (c.f. fig. 5.17); however, the relevant time scale might not have been reached yet. Once the network structure breaks down, the system is said to enter the final stage, the 'hydrodynamic regime' with a growth exponent smaller than the one of the elastic regime.

5.5 Minkowski Functionals

Minkowski functionals provide a way to extract geometrical information from complex morphologies and spacial patterns [101–104]. In d dimensions, there are $d+1$ Minkowski functionals $\mathscr{M}_i$. In three dimensions, they can be interpreted as volume, area, mean curvature, and Euler characteristic.

The Euler characteristic in this case is given by $\chi_{\mathrm{E}} = 2N_{\mathrm{c}}(1 - N_{\mathrm{h}})$, where N_{c} is the number of connected domains in the topology, and N_{h} is the number of holes. Hence, for a full sphere, the Euler characteristic is 2, and it is 0 for a torus. If the morphology is a single connected network with pores that are not all interconnected, the Euler characteristic is a negative number. If, on the other hand, the system consists of many separated droplets, the Euler characteristic is positive. It therefore seems to be a promising tool for the investigation of network formation and breakdown in viscoelastic phase separation.

The Minkowski functionals $\mathscr{M}_i, i = 0, \ldots, d$ are defined on convex rings R and have the following properties: They are additive, i.e. $\mathscr{M}_i(A \cup B) = \mathscr{M}_i(A) + \mathscr{M}_i(B) - \mathscr{M}_i(A \cap B)$ for A, $B \in R$, continuous, as well as translation and rotation invariant. The completeness theorem states that all functionals satisfying these conditions must be a linear combination of the $d+1$ Minkowski functionals [105].

In order to calculate the Minkowski functionals from the particle configurations obtained from the simulations, a suitable representation of the morphology needs to be created first. This is done following the approach described in ref. [101]. First, the mass density of the system is mapped onto a discrete cubic lattice with lattice constant a using the same methods described in section 5.4, see fig. 5.7. The lattice constant should be chosen small enough for the lattice to capture the features of interest, i.e. the size of the solvent holes, but not so small that it is sensitive to the particle size. This lattice density is then transformed into a 'black and white' representation, where a lattice site is assigned the value one, if its associated density exceeds a given threshold density ρ_{t}, and zero otherwise. Because of their additivity, the elementary Minkowski functionals can be calculated for each lattice cell individually and added up to give the full result.

In three dimensions, each cell has $2^3 = 8$ lattices sites at its edges. Since each site either has a value of one or zero, there are $2^8 = 256$ possible cell configurations. Because of rotation symmetry, this can be reduced to a number of 22 cell configurations which are unique wrt. the value of the Minkowski functionals. These cell configurations and their associated values can be found in ref. [101]. For mean curvature and Euler characteristics, two alternative values, depending on whether or not two diagonally adjacent cells should be considered connected, ($8n$ resp. $26n$) or not ($4n$ resp. $26n$), are given. A derivation of the Minkowski functional values for the elementary cells can be found in ref. [106].

In the present implementation, all lattice cells are iterated, and it is determined which

of the 256 cell configurations is that of the current cell. This is done by calculating the number

$$C = \sum_{i=0}^{7} 2^i \sigma_i, \tag{5.15}$$

where the sum is over all surrounding lattice sites, and σ_i is either one or zero, depending on whether the threshold density is exceeded or not. The number C is unique to each possible cell configuration. Hardcoded in the program is an array of size 256 containing pointers pointing to the elements of a second array of size 22, which lists the values of the elementary Minkowski functionals. These values are then obtained by following the pointer at index C in the large array.

This approach to calculating geometrical quantities is surprisingly close to what is done in some experiments. For example, Tanaka et al.use threshold values on grayscale images taken from the phase separation of thin films [11, 15, 100] to estimate the area occupied by the viscoelastic component. They observe a decrease of area over time, which they term 'volume shrinking'. Furthermore, they count the number of solvent holes forming, which is found to rise in the initial regime and decrease again in later stages.

Mecke and Sofonea have used Minkowski functionals to study the dynamic morphology of spinodal decomposition with two-dimensional Lattice Boltzmann fluids in ref. [102]. They find a sharp rise in the Minkowski area in the early stage, followed by a slow increase. The boundary length is found to rise steeply in the beginning as well but decreases again quickly after reaching its maximum value. That is to be expected as the boundary length in two dimensions can be associated to the area in three dimensions, the minimization of area of course being a central driving force in the dynamics of phase separation. Similarly, they find a sharp increase of the Euler characteristics in the beginning corresponding to the formation of many disconnected domains of the minority phase. As these domains grow and coagulate, the Euler characteristic decreases again.

Presented here are results obtained from the same system as shown in section 5.4.2, which contains 4784 chains of length 64 in a three-dimensional box of linear size $L = 128$ and was quenched with a depth of $\phi = 1$. All curves are again averages obtained from eight independent systems. The remaining parameters can be found in section 5.4.2.

Plots of the time-dependent Minkowski volume normalized by L^3 for different lattice constants a of the discretization lattice are shown in fig. 5.18. The threshold value is fixed at the average density $\rho_\mathrm{t} = \bar{\rho}$. At this threshold value, the decrease in volume is observed at all lattice constants. This is in line with the observations made by Tanaka et al., and differs from the results obtained by Mecke et al. for standard spinodal decomposition. The interpretation is that as the system coarsens, the solvent is displaced from the polymer-dense regions, which decreases their overall volume fraction.

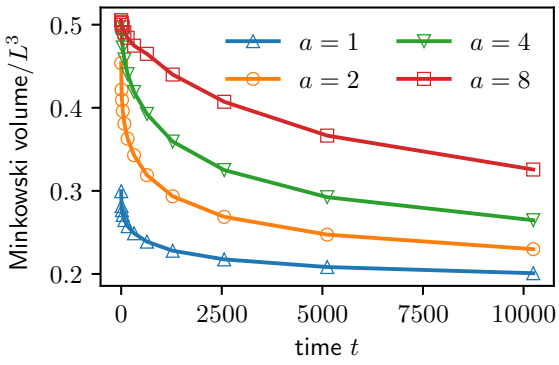


Figure 5.18: Minkowski volume where the threshold density is fixed at the average density $\rho_t = \bar{\rho}$ and the lattice constant a of the discretization lattice is varied.

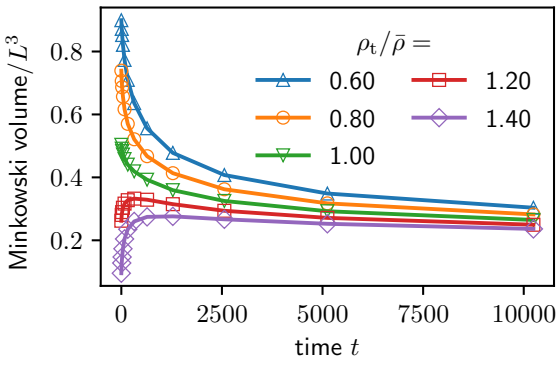


Figure 5.19: Minkowski volume where the lattice constant a of the discretization lattice is fixed at 4 and the threshold density ρ_t is varied.

The initial value of the normalized volume is at about 0.5 for $a = 4$ and larger. This is expected, since, in a homogeneous system, half of the cells should be above the average density, and half of them below. If the lattice constant is too small, there are too many empty cells, and the volume is underestimated. Hence, a lattice constant of 4 is chosen for the following plots. This is at 1/32 of the linear system size, and ideally larger systems should be considered in the future.

The Minkowski volume at different threshold values and fixed lattice constant $a = 4$ is shown in fig. 5.19. The initial behavior is very much sensitive to the choice of the threshold. At a threshold of $1.2\bar{\rho}$ and higher, a rise in volume is seen at the beginning, which is not the case at lower values. The longtime behavior however seems to be universal for all thresholds, although the scaling regime might change again at later times, for example, when network structures begin to break down.

The remaining Minkowski functionals for $a = 4$ and $\rho_t = \bar{\rho}$ are shown in fig. 5.20. The area shows a slight increase in the beginning before it reaches a peak and decreases again in the long run (blue curve in the right image of the top row). However, it has to be noted that this again strongly depends on the choice of the threshold value. Both for lower and higher threshold $\rho_t = 0.6$ and $\rho_t = 1.4$, the initial rise is much more pronounced (yellow and green curves). An interpretation is that the formation of solvent holes in the beginning leads to an increase of the total surface of the polymer-rich component, which is then again decreased by the coarsening of the morphology.

Similar behavior is seen in the curvature plots (left image second row). Again, many small holes could explain a higher mean curvature in the beginning. As the holes grow, the mean curvature decreases again.

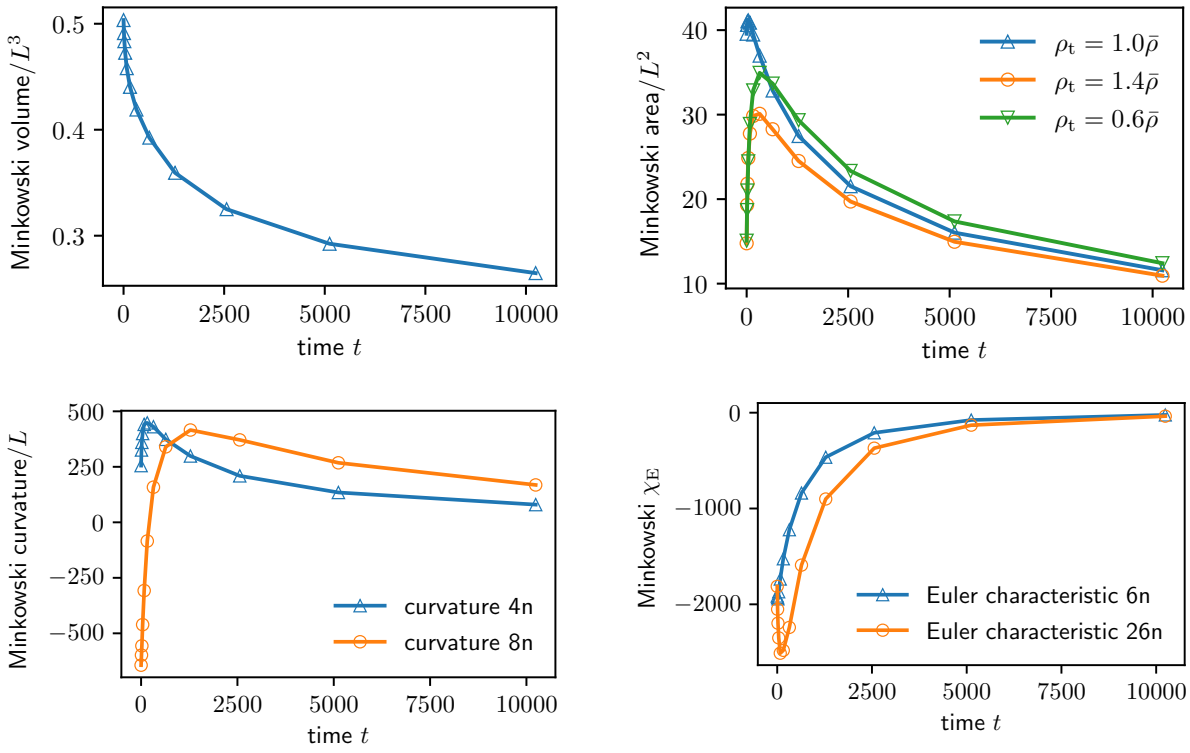


Figure 5.20: Minkowski functionals at a threshold density of $\rho_t = \bar{\rho}$ (if not stated otherwise), and lattice constant $a = 4$. The 8-neighbor mean curvature was scaled by a factor of 0.07 for better visibility.

The Euler characteristic (right image second row) behaves differently depending on whether the highly connected ($26n$) or less connected ($6n$) variant is used for the calculation of the elementary functionals. For the lower connected version, the Euler characteristic starts at a large negative value and increases with time, meaning that the number of holes is reduced. The highly connected variant however shows a dip in the beginning, indicating that the number of holes first increases before it decreases again, which is more in line with the intuitive picture. This is then again highly dependent on the threshold value, as it is seen in fig. 5.21. While for a lower threshold, both variants show the dip in the

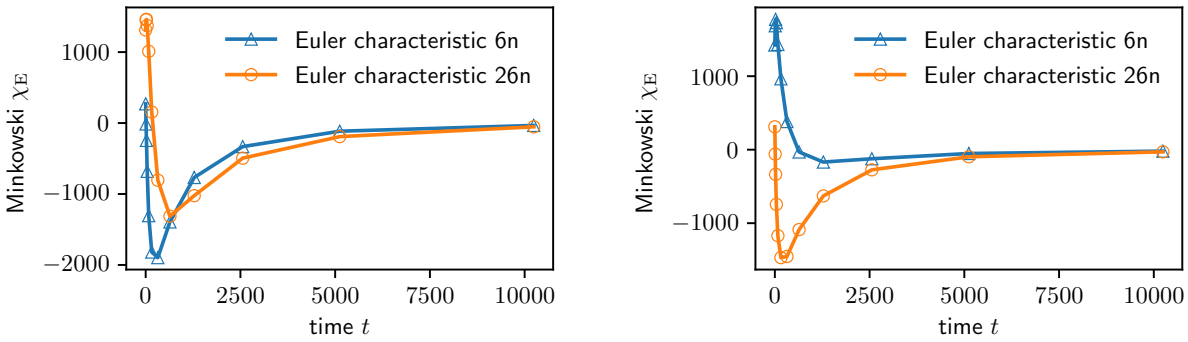


Figure 5.21: Minkowski Euler characteristic at lattice constant $a = 4$ with a threshold density of $\rho_t = 0.6\bar{\rho}$ (left) and $\rho_t = 1.6\bar{\rho}$ (right).

beginning; this is at the higher threshold only the case for the highly connected variant. The less connected variant detects a large number of separate domains in the beginning, which briefly rises initially. This number then decreases again as the domains coagulate, which is a behavior one would expect from regular spinodal decomposition.

While the Minkowski functionals can give some qualitative idea of the dynamic behavior of the morphology, they are not very robust with respect to variations of the lattice constant and threshold density. Therefore, it is hard to justify their use for the quantitative comparison of different phase separation models as it was the original intent here. Only at long times, the behavior seems to become universal. This can be exploited to calculate dynamic scaling laws, which was also done in ref. [102]. This is exemplified in the double logarithmic plots of the Minkowski volume and Euler characteristic in fig. 5.22. The Minkowski volume seems to approach a scaling law $V \propto t^{-1}$, while the Euler characteristic is closer to $\chi_E \propto t^{-2}$. However, it is also apparent that the universal regime is just barely reached, and longer times need to be simulated.

Nevertheless, the Minkowski functionals can confirm the existence of distinct regimes in the dynamics, as was also seen in the analysis of the dynamic structure factor in section 5.4.2.

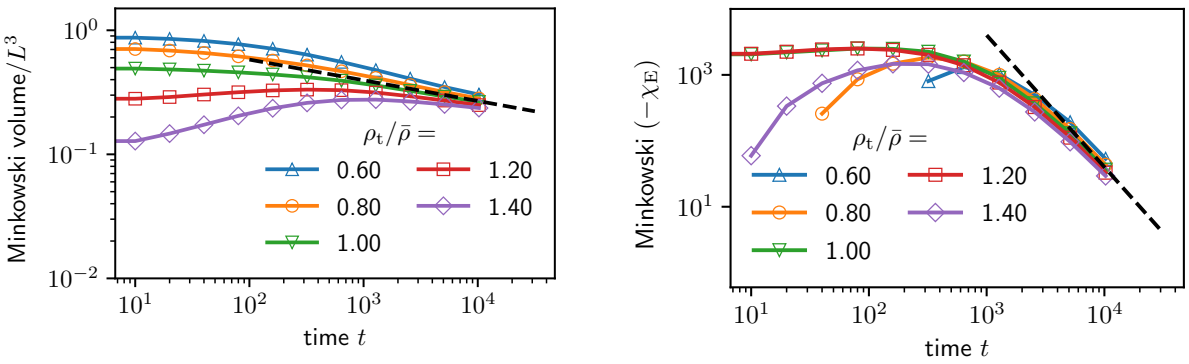


Figure 5.22: Double logarithmic plot of the Minkowski volume (left), and the negative of the Euler characteristic (right), where the lattice constant a of the discretization lattice is fixed at 4, and the threshold density ρ_t is varied. The dashed line in the left figure has a slope of -1, the line in the right figure has a slope of -2.

Chapter 6

Conclusion

In this thesis, the spinodal decomposition of polymer-solvent systems has been studied both numerically and theoretically. It was shown that the coupled Lattice Boltzmann/Molecular Dynamics (LB/MD) simulation scheme is indeed able to model non-standard phase separation behavior. Based on this simulation scheme, a set of continuum equations that aims at modeling the dynamics of viscoelastic phase separation was derived.

To facilitate the simulation of two-dimensional systems, an external potential confining the particles to the immediate vicinity of a plane was introduced. For systems that are confined in this way, the critical attraction strength of the non-bonded potential corresponding to the theta-solvent was determined by exploiting fundamental scaling relations of polymer chains.

Three-dimensional systems were simulated at different densities. The resulting pressure curve was compared to the equation of state obtained from Van der Waals theory, and first estimates for its parameters were determined.

The dynamics of geometrical properties of the polymer morphologies occurring during phase separation were examined using Minkowski functionals. Their short-time behavior was found to be very sensitive to the method parameters used when interpolating the polymer structure to a binary lattice. Hence, the Minkowski functionals at short times can be used for a qualitative analysis at best. However, the long-time behavior turned out to be more robust and can be used to determine dynamic scaling exponents.

The structure factor and related dynamic scaling curves were calculated from both two- and three-dimensional simulations at successive time steps after a quench of the solvent quality. By comparison with the structure factors obtained from the phase separation of a simple Lennard-Jones fluid, it was shown that the demixing dynamics is indeed non-standard. First comparisons to structure factor data obtained from a two-dimensional

macroscopic model were made. While some parallels could be drawn, it was found that the macroscopic models available at the time were not well-suited for an in-depth parameter matching with the mesoscopic LB/MD method.

Because of that, a novel set of dynamic continuum equations for the description of viscoelastic phase separation has been derived. In doing so, a new strategy has been taken combining Hamiltonian Poisson bracket methods with the insights from the GENERIC formalism. This approach avoids some of the conceptual problems inherent in classic kinetic approaches such as the Oldroyd-B model and at the same time allows to maintain a close connection to microscopic physics.

The starting point is a mesoscopic picture where polymer molecules are approximated by simple Hookean dumbbells that do not interact directly but only through a background-free energy. By defining appropriate fields, which can be expressed in the mesoscopic variables, the Poisson bracket formalism was used to construct the Hamiltonian part of the dynamics. Then, the GENERIC formalism was applied to determine the dissipative contribution from the continuum dissipation rate. Up to this point, no approximations or assumptions were necessary other than the postulation of suitable continuum limits of the Hamiltonian and the dissipation rate. From this basis, the equations were then simplified by adiabatically eliminating fast variables and introducing phenomenological terms to retain thermodynamic consistency. While sharing some similarities, the new model differs from the existing ones in some key aspects: The difficulties of the Oldroyd-B model that arise when combining Fokker-Planck and continuum picture are avoided by treating all variables on the same footing. Instead of separate bulk and shear stresses with unclear interpretation, the conformation tensor is the only variable describing the elastic effects. As one would expect from entropic elasticity, the conformation tensor enters the free energy linearly and not quadraticly. Because fluctuations are not yet included, the conformation tensor relaxes towards zero rather than some finite value.

By construction, the thus obtained model is tightly connected to both the underlying microscopic physics and the mesoscopic simulation model. Only this makes a future parameter matching feasible and removes a significant obstacle in creating a computationally efficient simulation method for viscoelastic phase separation, which is nevertheless sound in terms of fundamental physics.

Furthermore, a Python script for the calculation of Lattice Boltzmann weights has been developed. It provides a tool that dramatically simplifies the task of constructing LB models with a large number of velocities and a high degree of isotropy. The novelty of the approach lies in using random numbers to transform the tensorial Maxwell-Boltzmann constraints into a set of linear equations, which can be solved using standard linear algebra routines. The script can also be used to validate existing models that are found in the literature. Methods from linear programming were applied to facilitate the determination

of useful models, even when the resulting system of equations has infinitely many solutions.

6.1 Outlook

The two-fluid model as presented here does not yet include thermal fluctuations. Because of this, the conformation tensor relaxes towards zero, which constitutes one of the key differences to the existing models based on Oldroyd-B-like equations. In the future, thermal fluctuations can be incorporated by adding suitable random terms to the complete set of equations while ensuring that the resulting Langevin equations satisfy the fluctuation-dissipation theorem.

In terms of Rouse theory, the center of mass and relative vector used here to describe a dumbbell molecule correspond to the first two Rouse modes of a polymer chain. Given that the resulting set of Poisson brackets is closed, higher Rouse modes could be incorporated, resulting in a more detailed model. Furthermore, alternative spring potentials, for example the FENE potential, may be considered.

A formulation in terms of variables that are invariant with respect to a flip of the dumbbell orientation would be desirable. It was motivated how this could be done by using suitable tensors describing the dumbbell orientation and the orientation dynamics. This, however, artificially introduces new degrees of freedom. To retain a consistent description, these would most likely need to be compensated by introducing suitable constraints into the description. Whether or not this is possible and feasible has not been exhaustively resolved yet.

One of the next steps should be the numerical simulation of the model and comparison to the mesoscopic LB/MD model. Ideally, this should be done by simulating actual dumbbells in the mesoscopic picture first, and then investigate discrepancies occurring when moving to higher chain lengths.

The way in which the present equations have been derived is unconventional and avoids some of the problems existing with common rheological approaches. By combining Poisson brackets with the insights of the GENERIC formalism, it is possible to construct a continuum theory from a molecular picture without the need for many assumptions or approximations. This philosophy ensures consistency with microscopic physics in every step and could potentially be applied to a variety of different problems in the future.

References

[1] J. W. Cahn and J. E. Hilliard, "Free energy of a nonuniform system. i. interfacial free energy," *The Journal of Chemical Physics*, vol. 28, pp. 258–267, Feb. 1958.

[2] J. W. Cahn, "On spinodal decomposition," *Acta Metallurgica*, vol. 9, pp. 795–801, Sept. 1961.

[3] P. C. Hohenberg and B. I. Halperin, "Theory of dynamic critical phenomena," *Reviews of Modern Physics*, vol. 49, pp. 435–479, July 1977.

[4] K. Binder, "Theory of first-order phase transitions," *Reports on Progress in Physics*, vol. 50, pp. 783–859, July 1987.

[5] A. J. Bray, "Theory of phase-ordering kinetics," *Advances in Physics*, vol. 43, pp. 357–459, June 1994.

[6] A. Onuki, *Phase Transition Dynamics*. Cambridge University Press, 2007.

[7] K. Binder, H. L. Frisch, and J. Jäckle, "Kinetics of phase separation in the presence of slowly relaxing structural variables," *The Journal of Chemical Physics*, vol. 85, pp. 1505–1512, Aug. 1986.

[8] H. Tanaka and T. Nishi, "Anomalous phase separation behavior in a binary mixture of poly(vinyl methyl ether) and water under deep quench conditions," *Japanese Journal of Applied Physics*, vol. 27, p. L1787, Oct. 1988.

[9] J. H. Aubert, "Structural coarsening of demixed polymer solutions," *Macromolecules*, vol. 23, pp. 1446–1452, Mar. 1990.

[10] S.-W. Song and J. M. Torkelson, "Coarsening effects on microstructure formation in isopycnic polymer solutions and membranes produced via thermally induced phase separation," *Macromolecules*, vol. 27, pp. 6389–6397, Oct. 1994.

[11] H. Tanaka, "Unusual phase separation in a polymer solution caused by asymmetric molecular dynamics," *Physical Review Letters*, vol. 71, pp. 3158–3161, Nov. 1993.

[12] H. Tanaka and T. Araki, "Phase inversion during viscoelastic phase separation: Roles of bulk and shear relaxation moduli," *Physical Review Letters*, vol. 78, pp. 4966–4969, June 1997.

[13] M. Doi and A. Onuki, "Dynamic coupling between stress and composition in polymer solutions and blends," *Journal de Physique II*, vol. 2, pp. 1631–1656, Aug. 1992.

[14] S. T. Milner, "Dynamical theory of concentration fluctuations in polymer solutions under shear," *Physical Review E*, vol. 48, pp. 3674–3691, Nov. 1993.

[15] H. Tanaka, "Universality of viscoelastic phase separation in dynamically asymmetric fluid mixtures," *Physical Review Letters*, vol. 76, pp. 787–790, Jan. 1996.

[16] H. Tanaka, Y. Nishikawa, and T. Koyama, "Network-forming phase separation of colloidal suspensions," *Journal of Physics: Condensed Matter*, vol. 17, pp. L143–L153, Apr. 2005.

[17] M. Tateno and H. Tanaka, "Power-law coarsening in network-forming phase separation governed by mechanical relaxation," *Nature Communications*, vol. 12, p. 912, Feb. 2021.

[18] H. Tanaka, "Viscoelastic phase separation," *Journal of Physics: Condensed Matter*, vol. 12, pp. R207–R264, Mar. 2000.

[19] H. Pleiner and J. L. Harden, "General nonlinear 2-fluid hydrodynamics of complex fluids and soft matter," *AIP Conference Proceedings*, vol. 708, pp. 46–51, Apr. 2004.

[20] T. Taniguchi and A. Onuki, "Network domain structure in viscoelastic phase separation," *Physical Review Letters*, vol. 77, pp. 4910–4913, Dec. 1996.

[21] H. Nakazawa, S. Fujinami, M. Motoyama, T. Ohta, T. Araki, H. Tanaka, T. Fujisawa, H. Nakada, M. Hayashi, and M. Aizawa, "Phase separation and gelation of polymer-dispersed liquid crystals," *Computational and Theoretical Polymer Science*, vol. 11, no. 6, pp. 445–458, 2001.

[22] D. Zhou, P. Zhang, and W. E, "Modified models of polymer phase separation," *Physical Review E*, vol. 73, no. 6, p. 061801, 2006.

[23] M. Doi, "Onsager's variational principle in soft matter," *Journal of Physics: Condensed Matter*, vol. 23, no. 28, p. 284118, 2011.

[24] M. Doi, *Soft Matter Physics*. Oxford ; New York: Oxford University Press, June 2013.

[25] J. Zhou and M. Doi, "Dynamics of viscoelastic filaments based on onsager principle," *Physical Review Fluids*, vol. 3, Aug. 2018.

[26] H. Stark and T. C. Lubensky, "Poisson-bracket approach to the dynamics of nematic liquid crystals," *Physical Review E*, vol. 67, p. 061709, June 2003.

[27] M. Grmela and H. C. Öttinger, "Dynamics and thermodynamics of complex fluids. i. development of a general formalism," *Physical Review E*, vol. 56, pp. 6620–6632, Dec. 1997.

[28] H. C. Öttinger and M. Grmela, "Dynamics and thermodynamics of complex fluids. ii. illustrations of a general formalism," *Physical Review E*, vol. 56, pp. 6633–6655, Dec. 1997.

[29] M. Rubinstein and R. H. Colby, *Polymer Physics*. Oxford, New York: Oxford University Press, June 2003.

[30] P.-G. de Gennes, *Scaling Concepts in Polymer Physics*. Cornell University Press, 1979.

[31] B. Dünweg, "Introduction to theoretical polymer physics, lecture notes." `https://users.monash.edu.au/~rprakash/PolymPhysCourse/index.html`, 2011.

[32] B. Dünweg, "Polymer solutions," in *Handbook of Materials Modeling: Methods: Theory and Modeling* (W. Andreoni and S. Yip, eds.), pp. 1361–1379, Cham: Springer International Publishing, 2020.

[33] D. Mukherji, C. M. Marques, T. Stuehn, and K. Kremer, "Depleted depletion drives polymer swelling in poor solvent mixtures," *Nature Communications*, vol. 8, p. 1374, Nov. 2017.

[34] P. E. Rouse, "A theory of the linear viscoelastic properties of dilute solutions of coiling polymers," *The Journal of Chemical Physics*, vol. 21, pp. 1272–1280, July 1953.

[35] B. H. Zimm, "Dynamics of polymer molecules in dilute solution: Viscoelasticity, flow birefringence and dielectric loss," *The Journal of Chemical Physics*, vol. 24, pp. 269–278, Feb. 1956.

[36] P. Ahlrichs, R. Everaers, and B. Dünweg, "Screening of hydrodynamic interactions in semidilute polymer solutions: A computer simulation study," *Physical Review E*, vol. 64, p. 040501, Sept. 2001.

[37] P. J. Flory, "Thermodynamics of high polymer solutions," *The Journal of Chemical Physics*, vol. 10, pp. 51–61, Jan. 1942.

[38] M. L. Huggins, "Some properties of solutions of long-chain compounds.," *The Journal of Physical Chemistry*, vol. 46, pp. 151–158, Jan. 1942.

[39] P. J. Flory, *Principles of Polymer Chemistry*. Cornell University Press, 1953.

[40] R. A. L. Jones, *Soft Condensed Matter*. Oxford ; New York: Oxford University Press, June 2002.

[41] M. G. Bawendi and K. F. Freed, "Systematic corrections to flory–huggins theory: Polymer–solvent–void systems and binary blend–void systems," *The Journal of Chemical Physics*, vol. 88, pp. 2741–2756, Feb. 1988.

[42] J. Dudowicz and K. F. Freed, "Effect of monomer structure and compressibility on the properties of multicomponent polymer blends and solutions: 1. lattice cluster theory of compressible systems," *Macromolecules*, vol. 24, pp. 5076–5095, Sept. 1991.

[43] J. Dudowicz, M. S. Freed, and K. F. Freed, "Effect of monomer structure and compressibility on the properties of multicomponent polymer blends and solutions. 2. application to binary blends," *Macromolecules*, vol. 24, pp. 5096–5111, Sept. 1991.

[44] P. M. Chaikin and T. C. Lubensky, *Principles of Condensed Matter Physics*. Cambridge University Press, 2003.

[45] I. M. Lifshitz and V. V. Slyozov, "The kinetics of precipitation from supersaturated solid solutions," *Journal of Physics and Chemistry of Solids*, vol. 19, pp. 35–50, Apr. 1961.

[46] C. Wagner, "Theorie der alterung von niederschlägen durch umlösen (ostwald-reifung)," *Zeitschrift für Elektrochemie, Berichte der Bunsengesellschaft für physikalische Chemie*, vol. 65, no. 7-8, pp. 581–591, 1961.

[47] N. S. Landkof, *Foundations of Modern Potential Theory*. Grundlehren Der Mathematischen Wissenschaften, Berlin Heidelberg: Springer-Verlag, 1972.

[48] E. D. Siggia, "Late stages of spinodal decomposition in binary mixtures," *Physical Review A*, vol. 20, pp. 595–605, Aug. 1979.

[49] H. Furukawa, "Effect of inertia on droplet growth in a fluid," *Physical Review A*, vol. 31, pp. 1103–1108, Feb. 1985.

[50] J. G. Oldroyd and A. H. Wilson, "On the formulation of rheological equations of state," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 200, pp. 523–541, Feb. 1950.

[51] R. B. Bird, C. F. Curtiss, R. C. Armstrong, and O. Hassager, *Dynamics of Polymeric Liquids, Volume 2: Kinetic Theory, 2nd Edition*. Wiley, 1987.

[52] C. Le Bris and T. Lelièvre, "Micro-macro models for viscoelastic fluids: Modelling, mathematics and numerics," *Science China Mathematics*, vol. 55, pp. 353–384, Feb. 2012.

[53] P. Degond, A. Lozinski, and R. G. Owens, "Kinetic models for dilute solutions of dumbbells in non-homogeneous flows revisited," *Journal of Non-Newtonian Fluid Mechanics*, vol. 165, pp. 509–518, May 2010.

[54] H. Risken, *The Fokker-Planck Equation: Methods of Solution and Applications*. Springer Series in Synergetics, Berlin Heidelberg: Springer-Verlag, 1984.

[55] R. G. Larson, *Constitutive Equations for Polymer Melts and Solutions*. Boston: AT&T Bell Laboratories, 1988.

[56] D. Spiller, A. Brunk, O. Habrich, H. Egger, M. Lukáčová-Medviová, and B. Dünweg, "Systematic derivation of hydrodynamic equations for viscoelastic phase separation," *Journal of Physics: Condensed Matter*, 2021.

[57] L. Onsager, "Reciprocal relations in irreversible processes. i.," *Physical Review*, vol. 37, pp. 405–426, Feb. 1931.

[58] L. Onsager, "Reciprocal relations in irreversible processes. ii.," *Physical Review*, vol. 38, pp. 2265–2279, Dec. 1931.

[59] K. Kremer and G. S. Grest, "Dynamics of entangled linear polymer melts: a molecular-dynamics simulation," *The Journal of Chemical Physics*, vol. 92, pp. 5057–5086, Apr. 1990.

[60] T. Soddemann, B. Dünweg, and K. Kremer, "A generic computer model for amphiphilic systems," *The European Physical Journal E*, vol. 6, pp. 409–419, Dec. 2001.

[61] M. O. Steinhauser, "A molecular dynamics study on universal properties of polymer chains in different solvent qualities. part i. a review of linear chain properties," *The Journal of Chemical Physics*, vol. 122, p. 094901, Feb. 2005.

[62] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*. San Diego: Academic Press, 2nd revised edition. ed., 2002.

[63] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*. Oxford University Press, Aug. 2017.

[64] L. Verlet, "Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules," *Physical Review*, vol. 159, pp. 98–103, July 1967.

[65] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters," *The Journal of Chemical Physics*, vol. 76, pp. 637–649, Jan. 1982.

[66] M. Tuckerman, B. J. Berne, and G. J. Martyna, "Reversible multiple time scale molecular dynamics," *The Journal of Chemical Physics*, vol. 97, pp. 1990–2001, Aug. 1992.

[67] A. A. Chialvo and P. G. Debenedetti, "On the use of the verlet neighbor list in molecular dynamics," *Computer Physics Communications*, vol. 60, pp. 215–224, Sept. 1990.

[68] B. Dünweg and W. Paul, "Brownian dynamics simulations without gaussian random numbers," *International Journal of Modern Physics C*, vol. 02, pp. 817–827, Sept. 1991.

[69] B. Leimkuhler and C. Matthews, "Rational construction of stochastic numerical methods for molecular sampling," *Applied Mathematics Research eXpress*, vol. 2013, pp. 34–56, Jan. 2013.

[70] B. Leimkuhler and C. Matthews, "Robust and efficient configurational molecular sampling via langevin dynamics," *The Journal of Chemical Physics*, vol. 138, p. 174102, May 2013.

[71] G. N. Milstein and M. V. Tretyakov, "Quasi-symplectic methods for langevin-type equations," *IMA Journal of Numerical Analysis*, vol. 23, pp. 593–626, Oct. 2003.

[72] H. V. Guzman, N. Tretyakov, H. Kobayashi, A. C. Fogarty, K. Kreis, J. Krajniak, C. Junghans, K. Kremer, and T. Stuehn, "Espresso++ 2.0: Advanced methods for multiscale molecular simulation," *Computer Physics Communications*, vol. 238, pp. 66–76, May 2019.

[73] B. Dünweg and A. J. C. Ladd, "Lattice boltzmann simulations of soft matter systems," in *Advanced Computer Simulation Approaches for Soft Matter Sciences III* (C. Holm and K. Kremer, eds.), no. 221 in Advances in Polymer Science, pp. 89–166, Berlin Heidelberg: Springer, 2009.

[74] 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.

[75] S. Succi, *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*. Numerical Mathematics and Scientific Computation, Oxford, New York: Oxford University Press, May 2013.

[76] B. Dünweg, U. D. Schiller, and A. J. C. Ladd, "Statistical mechanics of the fluctuating lattice boltzmann equation," *Physical Review E*, vol. 76, p. 036704, Sept. 2007.

[77] D. d'Humieres, P. Lallemand, and Y. H. Quian, "Lattice bgk models for navier-stokes equations," *Europhysics Letters*, vol. 17, no. 6, pp. 479–484, 1992.

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

[79] D. Spiller and B. Dünweg, "Semiautomatic construction of lattice boltzmann models," *Physical Review E*, vol. 101, p. 043310, Apr. 2020. `https://doi.org/10.1103/PhysRevE.101.043310`.

[80] H. Chen, I. Goldhirsch, and S. A. Orszag, "Discrete rotational symmetry, moment isotropy, and higher order lattice boltzmann models," *Journal of Scientific Computing*, vol. 34, pp. 87–112, Jan. 2008.

[81] L. D. Landau and E. Lifshitz, *Fluid Mechanics*, vol. 6. Pergamon Press, 1959.

[82] J. D. Halverson, T. Brandes, O. Lenz, A. Arnold, S. Bevc, V. Starchenko, K. Kremer, T. Stuehn, and D. Reith, "Espresso++: A modern multiscale simulation package for soft matter systems," *Computer Physics Communications*, vol. 184, pp. 1129–1149, Apr. 2013.

[83] N. Tretyakov and B. Dünweg, "An improved dissipative coupling scheme for a system of molecular dynamics particles interacting with a lattice boltzmann fluid," *Computer Physics Communications*, vol. 216, pp. 102–108, July 2017.

[84] B. Dünweg and D. Spiller, "Lattice-boltzmann-weights." `https://github.com/BDuenweg/Lattice-Boltzmann-weights`, May 2020.

[85] J. Langer, "An introduction to the kinetics of first-order phase transition," in *Solids Far from Equilibrium. Edited by C. Godrèche*, p. 297, Cambridge University Press, 1992.

[86] M. Plapp, "Phase-field models," in *Multiphase Microfluidics: The Diffuse Interface Model* (R. Mauri, ed.), CISM Courses and Lectures, pp. 129–175, Vienna: Springer, 2012.

[87] Y. M. Ivanchenko and A. A. Lisyansky, "The ginzburg-landau functional," in *Physics of Critical Fluctuations* (Y. M. Ivanchenko and A. A. Lisyansky, eds.), Graduate Texts in Contemporary Physics, pp. 29–63, New York, NY: Springer, 1995.

[88] S. S. Chikatamarla and I. V. Karlin, "Lattices for the lattice boltzmann method," *Physical Review E*, vol. 79, no. 4, p. 046701, 2009.

[89] "Itertools — functions creating iterators for efficient looping — python 3.8.2." `https://docs.python.org/3.8/library/itertools.html`.

[90] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge, UK ; New York: Cambridge University Press, third ed., Sept. 2007.

[91] "Numpy.linalg.svd — numpy v1.18." `https://numpy.org/doc/1.18/reference/generated/numpy.linalg.svd.html`.

[92] "Numpy.roots — numpy v1.18." `https://numpy.org/doc/1.18/reference/generated/numpy.roots.html`.

[93] "Cvxpy 1.0.26." `https://www.cvxpy.org/api_reference/cvxpy.html`.

[94] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[95] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[96] X. Shan, "The mathematical structure of the lattices of the lattice boltzmann method," *Journal of Computational Science*, vol. 17, pp. 475–481, Nov. 2016.

[97] "Fftw home page." `http://fftw.org/`.

[98] A. Brunk, B. Dünweg, H. Egger, O. Habrich, M. Lukáčová-Medviová, and D. Spiller, "Analysis of a viscoelastic phase separation model," *Journal of Physics: Condensed Matter*, 2021.
`https://doi.org/10.1088/1361-648X/abeb13`.

[99] A. Brunk and M. Lukáčová-Medviová, "Global existence of weak solutions to viscoelastic phase separation: Part ii degenerate case." `https://arxiv.org/abs/2004.14790v1`, Apr. 2020.

[100] H. Tanaka and T. Araki, "Viscoelastic phase separation in soft matter: Numerical-simulation study on its physical mechanism," *Chemical Engineering Science*, vol. 61, pp. 2108–2141, Apr. 2006.

[101] C. H. Arns, M. A. Knackstedt, W. V. Pinczewski, and K. R. Mecke, "Euler-poincaré characteristics of classes of disordered media," *Physical Review E*, vol. 63, p. 031112, Feb. 2001.

[102] K. R. Mecke and V. Sofonea, "Morphology of spinodal decomposition," *Physical Review E*, vol. 56, pp. R3761–R3764, Oct. 1997.

[103] K. R. Mecke, "Additivity, convexity, and beyond: Applications of minkowski functionals in statistical physics," in *Statistical Physics and Spatial Statistics* (K. R.

Mecke and D. Stoyan, eds.), Lecture Notes in Physics, pp. 111–184, Springer Berlin Heidelberg, 2000.

[104] K. R. Mecke and D. Stoyan, eds., *Statistical Physics and Spatial Statistics: The Art of Analyzing and Modeling Spatial Structures and Pattern Formation*. Lecture Notes in Physics, Berlin Heidelberg: Springer-Verlag, 2000.

[105] H. Hadwiger, *Vorlesungen Über Inhalt, Oberfläche und Isoperimetrie*. Grundlehren der mathematischen Wissenschaften, Berlin Heidelberg: Springer-Verlag, 1975.

[106] S. T. Hyde, I. S. Barnes, and B. W. Ninham, "Curvature energy of surfactant interfaces confined to the plaquettes of a cubic lattice," *Langmuir*, vol. 6, pp. 1055–1062, June 1990.

[107] W. Greiner and J. Reinhardt, *Field Quantization*. Berlin Heidelberg: Springer-Verlag, 1996.

[108] R. Courant and D. Hilbert, *Methods of Mathematical Physics Volume 1*. Interscience Publishers Inc., 1953.

[109] P. A. M. Dirac, "Generalized hamiltonian dynamics," *Canadian Journal of Mathematics*, vol. 2, pp. 129–148, 1950/ed.

[110] P. A. M. Dirac, *Lectures on Quantum Mechanics*. Mineola, NY: Dover Publications, 2001.

[111] K. F. Freed and J. Dudowicz, "Role of monomer structure and compressibility on the properties of multicomponent polymer blends and solutions," *Theoretica chimica acta*, vol. 82, pp. 357–382, Sept. 1992.

[112] "Sphinx documentation." `https://www.sphinx-doc.org`.

[113] "Doxygen." `https://www.doxygen.nl`.

Index

Appendix A

Supplementary material

This part of the appendix compiles some relevant supplementary material that would otherwise hinder the flow of the main text.

Functional derivatives occur in numerous places throughout this thesis and are introduced in appendix A.1. The particular form of the chain- and product rule for functional derivatives are found there.

Appendix A.2 features an Eulerian two-fluid model that results as a special case from the viscoelastic model derived in section 4.1. The transformation to respective two-fluid variables is shown explicitly. The resulting dynamic equations depend on total pressure and a relative chemical potential that are obtained from a general two-fluid Helmholtz free energy.

In appendix A.3, some supplementary calculations to the viscoelastic model in section 4.1 are collected. Furthermore, tensorial variables for the description of dumbbell conformation and conformation dynamic, which are invariant with respect to a flip of the connecting vector, are introduced.

Appendix A.4 presents a derivation of an alternative equation of state for a compressible mixture based on Flory-Huggins theory. This is done by introducing empty lattice sites of variable number, so-called voids, which do not interact energetically but admit a change of total system volume.

In appendix A.5, the effects of an external confining on the pressure tensor are examined. The virial expansion for that particular case is shown explicitly. From there, it can be seen that for the particular potential introduced in section 3.1.6 which confines the particles in the z-plane, the pressure in the xy-plane behaves exactly as in a real two-dimensional system.

A.1 Some notes on functional derivatives

Functional derivatives are used in numerous places in this thesis. This section provides a brief overview as well as some helpful calculation rules. A more in-depth treatment of functional derivatives can be found e.g. in the references [107, 108].

Consider a general functional $\mathscr{F}$, which takes some function f and maps it onto a scalar value. We define the variation of $\mathscr{F}$ when $f(\mathbf{r})$ is varied by some arbitrary test function $\eta(\mathbf{r})$ with strength $\varepsilon \to 0$, $\delta f(\mathbf{r}) = \varepsilon\eta(\mathbf{r})$, by

$$\delta\mathscr{F}[f] = \mathscr{F}[f + \varepsilon\eta] - \mathscr{F}[f] = \int \mathrm{d}^d\mathbf{r}\, \frac{\delta\mathscr{F}[f]}{\delta f(\mathbf{r})}\delta f(\mathbf{r}), \tag{A.1}$$

where $\delta F[f]/\delta f(\mathbf{r})$ is the functional derivative of F at position $\mathbf{r}$. This can be understood as the functional generalization of the total differential

$$\mathrm{d}g(\mathbf{x}) = \sum_i \frac{\partial g}{\partial x_i}\,\mathrm{d}x_i\,. \tag{A.2}$$

By setting $\delta f = \varepsilon\delta(\mathbf{r} - \mathbf{r}')$, the integral in eq. (A.1) can be evaluated and we may write

$$\frac{\delta\mathscr{F}[f]}{\delta f(\mathbf{r}')} = \lim_{\varepsilon\to 0} \frac{\mathscr{F}[f + \varepsilon\delta(\mathbf{r} - \mathbf{r}')] - \mathscr{F}[f]}{\varepsilon} = \left.\frac{\partial\mathscr{F}[f + \varepsilon\delta(\mathbf{r} - \mathbf{r}')]}{\partial\varepsilon}\right|_{\varepsilon=0}. \tag{A.3}$$

While the argument $\mathbf{r}$ still appears on the right-hand side, the functional derivative does not depend on it; it is considered a 'silent' variable. Equation (A.3) provides one way of calculating functional derivatives.

Consider now a functional given by the integral

$$\mathscr{F}[\mathbf{x}(\mathbf{r}, t)](t) = \int \mathrm{d}^d\mathbf{r}\, F(\mathbf{x}, \boldsymbol{\nabla}\mathbf{x}), \tag{A.4}$$

which is defined as the integral over a function that has $\mathbf{x}$ and $\boldsymbol{\nabla}\mathbf{x}$ as independent variables. The variation of the functional $\mathscr{F}$ can be expressed by

$$\begin{aligned}\delta\mathscr{F}[\{x_\alpha\}] &= \int \mathrm{d}^d\mathbf{r} \left[\frac{\partial F}{\partial x_\gamma}\delta x_\gamma + \frac{\partial F}{\partial(\partial_\delta x_\gamma)}\partial_\delta\delta x_\gamma\right] \\ &= \int \mathrm{d}^d\mathbf{r} \left[\frac{\partial F}{\partial x_\gamma} - \partial_\delta\frac{\partial F}{\partial(\partial_\delta x_\gamma)}\right]\delta x_\gamma.\end{aligned} \tag{A.5}$$

Comparing to the definition eq. (A.1), the functional derivative, which is itself a function of $\mathbf{r}$, is found to be

$$\frac{\delta\mathscr{F}}{\delta x_\alpha} = \frac{\partial F}{\partial x_\gamma} - \partial_\delta\frac{\partial F}{\partial(\partial_\delta x_\gamma)}. \tag{A.6}$$

It can be shown that for two functionals $\mathscr{F}_1, \mathscr{F}_2$ there is a product rule

$$\frac{\delta(\mathscr{F}_1\mathscr{F}_2)}{\delta f(\mathbf{r})} = \frac{\delta\mathscr{F}_1}{\delta f(\mathbf{r})}\mathscr{F}_2 + \mathscr{F}_1\frac{\delta\mathscr{F}_2}{\delta f(\mathbf{r})}. \tag{A.7}$$

Furthermore, for a functional $\mathscr{F}[\mathscr{G}]$ that depends on a functional $\mathscr{G}[f](\mathbf{r}')$, which itself is a functional on f as well as a function on $\mathbf{r}'$, $\mathscr{F}[f]$ can be considered to be a functional on f directly and there is the chain rule

$$\frac{\delta\mathscr{F}[f]}{\delta f(\mathbf{r})} = \int \mathrm{d}^d\mathbf{r}' \, \frac{\delta\mathscr{F}[\mathscr{G}]}{\delta\mathscr{G}(\mathbf{r}')}\frac{\delta\mathscr{G}[f]}{\delta f(\mathbf{r})}. \tag{A.8}$$

A useful observation is that one can always express a function as a functional on itself by using a delta function:

$$f(\mathbf{r}) = \int \mathrm{d}^d\mathbf{r}' \, \delta(\mathbf{r}-\mathbf{r}')f(\mathbf{r}'), \qquad \frac{\delta f(\mathbf{r})}{\delta f(\mathbf{r}')} = \delta(\mathbf{r}-\mathbf{r}'). \tag{A.9}$$

Because of this, the chain rule with a regular function g instead of a functional $\mathscr{G}$ as inner function becomes

$$\frac{\delta\mathscr{F}[g(f)]}{\delta f(\mathbf{r})} = \frac{\delta\mathscr{F}[g]}{\delta g(f(\mathbf{r}))}\frac{\partial g(f)}{\partial f(\mathbf{r})}. \tag{A.10}$$

Furthermore, the derivative with respect to a variable t of f, which is not eliminated by taking the functional, can be expressed by

$$\frac{\partial\mathscr{F}[f(\mathbf{r},t)]}{\partial t} = \frac{\partial\mathscr{F}[f](t)}{\partial t} = \int \mathrm{d}^d\mathbf{r} \, \frac{\delta\mathscr{F}[f]}{\delta f(\mathbf{r})}\frac{\partial f(\mathbf{r},t)}{\partial t}. \tag{A.11}$$

A.2 Two-fluid Euler equations

The viscoelastic two-fluid model derived in section 4.1 can be reduced to a simple Eulerian two-fluid model where the coupling between components is only due to their thermodynamic equation of state. This is done by eliminating the elastic degrees of freedom of the dumbbell component, the effects of surface tension, as well as coupling and viscous dissipation in the solvent component. Starting point are the full equations eqs. (4.100) to (4.105). By setting $\mathbf{q} = \mathbf{v}^{(\mathrm{r})} = \boldsymbol{\eta} = \kappa = 0$ and taking the limit $\tau \to \infty$ these equations reduce to

$$\mathrm{D}_t^{(1)}\rho^{(1)} = -\rho^{(1)}\boldsymbol{\nabla}\cdot\mathbf{v}^{(1)}, \tag{A.12}$$

$$\mathrm{D}_t^{(1)}\mathbf{v}^{(1)} = -\boldsymbol{\nabla}\frac{\partial f}{\partial \rho^{(1)}}, \tag{A.13}$$

$$\mathrm{D}_t^{(2)}\rho^{(2)} = -\rho^{(2)}\boldsymbol{\nabla}\cdot\mathbf{v}^{(2)}, \tag{A.14}$$

$$\mathrm{D}_t^{(2)}\mathbf{v}^{(2)} = -\boldsymbol{\nabla}\frac{\partial f}{\partial \rho^{(2)}}, \tag{A.15}$$

where f is the thermodynamic free energy density. An expression for f in terms of total pressure and relative chemical potential is derived in the next section.

A.2.1 Helmholtz free energy

As we are interested in isothermal hydrodynamics, a Helmholtz free energy is used. In order to derive it, we imagine the system to be decomposed into subsystems that we shall call fluid elements. By the assumption of local equilibrium, every fluid element can be treated as an individual thermodynamic system in equilibrium with respect to the fast variables. A fluid element consists of a certain number of molecules of species one, $N^{(1)}$, and a certain number of molecules of species two, $N^{(2)}$. It is of constant mass M_{tot} but can be distorted and change its volume and composition. The differential Helmholtz free energy is then given by

$$\mathrm{d}F = -S\mathrm{d}T - P\mathrm{d}V + \mu^{(1)}\mathrm{d}N^{(1)} + \mu^{(2)}\mathrm{d}N^{(2)} \tag{A.16}$$

with entropy S, pressure P, volume of the fluid element V and chemical potentials of the components $\mu^{(i)}$. Temperature is set to be constant, hence $\mathrm{d}T = 0$. Equation (A.16) can be expressed with the mass densities instead of particle numbers by applying the relations

$$N^{(1)} = \frac{M_{\mathrm{tot}}}{m^{(1)}}\frac{\rho^{(1)}}{\rho}, \qquad N^{(2)} = \frac{M_{\mathrm{tot}}}{m^{(2)}}\frac{\rho^{(2)}}{\rho}. \tag{A.17}$$

Here the total mass density is denoted by $\rho = \rho^{(1)} + \rho^{(2)}$. Normalizing with the total mass $M_{\rm tot}$, the free energy becomes

$$\begin{aligned}
\mathrm{d}\tilde{f} = \frac{\mathrm{d}F}{M_{\rm tot}} &= -P\,\mathrm{d}\left(\frac{1}{\rho}\right) + \frac{\mu^{(1)}}{m^{(1)}}\,\mathrm{d}\left(\frac{\rho^{(1)}}{\rho}\right) + \frac{\mu^{(2)}}{m^{(2)}}\,\mathrm{d}\left(\frac{\rho^{(2)}}{\rho}\right) \\
&= \frac{P}{\rho^2}\mathrm{d}\rho + \frac{\tilde{\mu}^{(1)}}{\rho^2}\Big(\rho\,\mathrm{d}\rho^{(1)} - \rho^{(1)}\mathrm{d}\rho\Big) + \frac{\tilde{\mu}^{(2)}}{\rho^2}\Big(\rho\,\mathrm{d}\rho^{(2)} - \rho^{(2)}\mathrm{d}\rho\Big) \\
&= \frac{1}{\rho^2}\Big[\Big(P - \tilde{\mu}^{(1)}\rho^{(1)} - \tilde{\mu}^{(2)}\rho^{(2)}\Big)\mathrm{d}\rho + \Big(\tilde{\mu}^{(1)}\rho\,\mathrm{d}\rho^{(1)} + \tilde{\mu}^{(2)}\rho\,\mathrm{d}\rho^{(2)}\Big)\Big] \\
&= \frac{P}{\rho^2}\mathrm{d}\rho + \frac{\tilde{\mu}}{\rho^2}\Big(\rho^{(2)}\,\mathrm{d}\rho^{(1)} - \rho^{(1)}\,\mathrm{d}\rho^{(2)}\Big),
\end{aligned} \tag{A.18}$$

where

$$\tilde{\mu} = \tilde{\mu}^{(1)} - \tilde{\mu}^{(2)} = \frac{\mu^{(1)}}{m^{(1)}} - \frac{\mu^{(2)}}{m^{(2)}} \tag{A.19}$$

is the relative, mass-normalized chemical potential. With the free energy per volume $\tilde{f} = f/\rho$ we can alternatively write

$$\mathrm{d}\tilde{f} = \frac{1}{\rho^2}(\rho\mathrm{d}f - f\mathrm{d}\rho). \tag{A.20}$$

By using eqs. (A.18) and (A.20) we then find

$$\mathrm{d}f = \rho\,\mathrm{d}\tilde{f} + \frac{f}{\rho}\mathrm{d}\rho = \frac{1}{\rho}(P+f)\mathrm{d}\rho + \frac{\tilde{\mu}}{\rho}\Big(\rho^{(2)}\,\mathrm{d}\rho^{(1)} - \rho^{(1)}\,\mathrm{d}\rho^{(2)}\Big). \tag{A.21}$$

Equation (A.21) can now be used to calculate the derivatives of $F = \int \mathrm{d}^d\mathbf{r}\, f$:

$$\frac{\delta F}{\delta\rho^{(1)}} = \frac{\partial f}{\partial\rho^{(1)}} = \frac{P + f + \tilde{\mu}\rho^{(2)}}{\rho^{(1)} + \rho^{(2)}}, \tag{A.22}$$

$$\frac{\delta F}{\delta\rho^{(2)}} = \frac{\partial f}{\partial\rho^{(2)}} = \frac{P + f - \tilde{\mu}\rho^{(1)}}{\rho^{(1)} + \rho^{(2)}}. \tag{A.23}$$

Later in the course of the derivations we will also encounter the gradients of the derivatives. These can be computed by first looking at the individual differentials

$$\begin{aligned}
\mathrm{d}\left(\frac{P+f}{\rho}\right) = \mathrm{d}\left(\frac{P}{\rho}\right) + \mathrm{d}\left(\frac{f}{\rho}\right) &= \frac{1}{\rho}\mathrm{d}P - \frac{P}{\rho^2}\mathrm{d}\rho + \mathrm{d}\tilde{f} \\
&= \frac{1}{\rho}\mathrm{d}P + \frac{\tilde{\mu}}{\rho^2}\Big(\rho^{(2)}\,\mathrm{d}\rho^{(1)} - \rho^{(1)}\,\mathrm{d}\rho^{(2)}\Big),
\end{aligned} \tag{A.24}$$

and

$$\begin{aligned} \mathrm{d}\left(\frac{\tilde{\mu}\rho^{(2)}}{\rho}\right) &= \frac{\rho^{(2)}}{\rho}\mathrm{d}\tilde{\mu} + \frac{\tilde{\mu}}{\rho^2}\Big(\rho\mathrm{d}\rho^{(2)} - \rho^{(2)}\mathrm{d}\rho\Big) \\ &= \frac{\rho^{(2)}}{\rho}\mathrm{d}\tilde{\mu} + \frac{\tilde{\mu}}{\rho^2}\Big(\rho^{(1)}\mathrm{d}\rho^{(2)} - \rho^{(2)}\mathrm{d}\rho^{(1)}\Big), \end{aligned} \tag{A.25}$$

$$\mathrm{d}\left(\frac{\tilde{\mu}\rho^{(1)}}{\rho}\right) = \frac{\rho^{(1)}}{\rho}\mathrm{d}\tilde{\mu} + \frac{\tilde{\mu}}{\rho^2}\Big(\rho^{(2)}\mathrm{d}\rho^{(1)} - \rho^{(1)}\mathrm{d}\rho^{(2)}\Big). \tag{A.26}$$

By combining the above we find the gradients

$$\partial_\alpha \frac{\partial f}{\partial \rho^{(1)}} = \frac{1}{\rho}\partial_\alpha P + \frac{\rho^{(2)}}{\rho}\partial_\alpha \tilde{\mu}, \tag{A.27}$$

$$\partial_\alpha \frac{\partial f}{\partial \rho^{(2)}} = \frac{1}{\rho}\partial_\alpha P - \frac{\rho^{(1)}}{\rho}\partial_\alpha \tilde{\mu}. \tag{A.28}$$

A.2.2 Transformation to two-fluid variables

The two-fluid equations eqs. (A.12) to (A.15) are transformed into a representation using the same two fluid variables introduced in section 4.1: The individual densities $\rho^{(1)}$ and $\rho^{(2)}$ are replaced by the total density

$$\rho = \rho^{(1)} + \rho^{(2)} \tag{A.29}$$

and the normalized density difference

$$c = \frac{\rho^{(1)} - \rho^{(2)}}{\rho}, \tag{A.30}$$

which changes sign when $\rho^{(1)}$ and $\rho^{(2)}$ are exchanged. Furthermore, it is helpful to define the reduced density

$$\rho^{(\mathrm{red})} = \frac{\rho^{(1)}\rho^{(2)}}{\rho} = \frac{\rho}{4}\Big(1 - c^2\Big). \tag{A.31}$$

Note that this is invariant wrt. exchange of components. Therefore $\rho^{(\mathrm{red})}$ is only used to abbreviate notation, and c is kept as an independent variable.

Instead of the elementary velocities $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$, we introduce the barycentric (mass-average) velocity

$$\mathbf{v} = \frac{\rho^{(1)}\mathbf{v}^{(1)} + \rho^{(2)}\mathbf{v}^{(2)}}{\rho}, \tag{A.32}$$

and the velocity difference

$$\mathbf{w} = \mathbf{v}^{(1)} - \mathbf{v}^{(2)}. \tag{A.33}$$

We also take note of the inverse transformations

$$\rho^{(1)} = \frac{\rho}{2}(1+c), \qquad \rho^{(2)} = \frac{\rho}{2}(1-c), \tag{A.34}$$

$$\mathbf{v}^{(1)} = \mathbf{v} + \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w} = \mathbf{v} - \frac{c-1}{2}\mathbf{w}, \qquad \mathbf{v}^{(2)} = \mathbf{v} - \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}}\mathbf{w} = \mathbf{v} - \frac{c+1}{2}\mathbf{w}. \tag{A.35}$$

The mass-average velocity can then be written as

$$\mathbf{v} = \frac{1+c}{2}\mathbf{v}^{(1)} + \frac{1-c}{2}\mathbf{v}^{(2)}. \tag{A.36}$$

Furthermore, we define a new convective derivative with respect to the barycentric velocity

$$\begin{aligned} \mathrm{D}_t &= \partial_t + \mathbf{v}\cdot\boldsymbol{\nabla} \\ &= \partial_t + \left(\mathbf{v}^{(1)} + \frac{c-1}{2}\mathbf{w}\right)\cdot\boldsymbol{\nabla} = \mathrm{D}_t^{(1)} + \frac{c-1}{2}\mathbf{w}\cdot\boldsymbol{\nabla} \\ &= \partial_t + \left(\mathbf{v}^{(2)} + \frac{c+1}{2}\mathbf{w}\right)\cdot\boldsymbol{\nabla} = \mathrm{D}_t^{(2)} + \frac{c+1}{2}\mathbf{w}\cdot\boldsymbol{\nabla}, \end{aligned} \tag{A.37}$$

respectively

$$\mathrm{D}_t^{(1)} = \mathrm{D}_t + \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\cdot\boldsymbol{\nabla} = \mathrm{D}_t - \frac{c-1}{2}\mathbf{w}\cdot\boldsymbol{\nabla}, \tag{A.38}$$

$$\mathrm{D}_t^{(2)} = \mathrm{D}_t - \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}}\mathbf{w}\cdot\boldsymbol{\nabla} = \mathrm{D}_t - \frac{c+1}{2}\mathbf{w}\cdot\boldsymbol{\nabla}. \tag{A.39}$$

Therefore, the barycentric convective derivative can be written as

$$\mathrm{D}_t = \frac{1}{2}\Big(\mathrm{D}_t^{(1)} + \mathrm{D}_t^{(2)} + c\mathbf{w}\cdot\boldsymbol{\nabla}\Big). \tag{A.40}$$

Total density By applying the transformations introduced above, the mass-average convected derivatives of the individual component densities take the form

$$\begin{aligned} \mathrm{D}_t\rho^{(1)} &= \mathrm{D}_t^{(1)}\rho^{(1)} - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\cdot\boldsymbol{\nabla}\rho^{(1)} \\ &= -\rho^{(1)}\boldsymbol{\nabla}\cdot\mathbf{v}^{(1)} - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\cdot\boldsymbol{\nabla}\rho^{(1)} \\ &= -\rho^{(1)}\boldsymbol{\nabla}\cdot\mathbf{v} - \rho^{(1)}\boldsymbol{\nabla}\cdot\left(\frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\right) - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\cdot\boldsymbol{\nabla}\rho^{(1)} \\ &= -\rho^{(1)}\boldsymbol{\nabla}\cdot\mathbf{v} - \boldsymbol{\nabla}\cdot\Big(\rho^{(\mathrm{red})}\mathbf{w}\Big), \end{aligned} \tag{A.41}$$

and analogously

$$\mathrm{D}_t\rho^{(2)} = -\rho^{(2)}\boldsymbol{\nabla}\cdot\mathbf{v} + \boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{red})}\mathbf{w}\right). \tag{A.42}$$

The conservation equation for the total density is then obtained by taking the sum

$$\mathrm{D}_t\rho = -\rho\boldsymbol{\nabla}\cdot\mathbf{v}. \tag{A.43}$$

Density contrast Using eqs. (A.41) to (A.43), the equation of motion of the density contrast can be written as

$$\begin{aligned}\rho\mathrm{D}_t c &= \mathrm{D}_t\rho^{(1)} - \mathrm{D}_t\rho^{(2)} - c\mathrm{D}_t\rho \\ &= -\left(\rho^{(1)} - \rho^{(2)}\right)\boldsymbol{\nabla}\cdot\mathbf{v} - 2\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{red})}\mathbf{w}\right) + \rho c\boldsymbol{\nabla}\cdot\mathbf{v} \\ &= -2\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{red})}\mathbf{w}\right).\end{aligned} \tag{A.44}$$

Average velocity We proceed by considering the individual velocities

$$\begin{aligned}\mathrm{D}_t\mathbf{v}^{(1)} &= \mathrm{D}_t^{(1)}\mathbf{v}^{(1)} - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v}^{(1)} \\ &= -\boldsymbol{\nabla}\frac{\partial f}{\partial\rho^{(1)}} - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v}^{(1)},\end{aligned} \tag{A.45}$$

$$\mathrm{D}_t\mathbf{v}^{(2)} = -\boldsymbol{\nabla}\frac{\partial f}{\partial\rho^{(2)}} + \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}}\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v}^{(2)}. \tag{A.46}$$

Hence

$$\begin{aligned}\rho\mathrm{D}_t\mathbf{v} &= \rho\mathrm{D}_t\frac{\rho^{(1)}\mathbf{v}^{(1)} + \rho^{(2)}\mathbf{v}^{(2)}}{\rho} \\ &= \rho^{(1)}\mathrm{D}_t\mathbf{v}^{(1)} + \rho^{(2)}\mathrm{D}_t\mathbf{v}^{(2)} + \mathbf{v}^{(1)}\mathrm{D}_t\rho^{(1)} + \mathbf{v}^{(2)}\mathrm{D}_t\rho^{(2)} - \mathbf{v}\mathrm{D}_t\rho\end{aligned} \tag{A.47}$$

becomes

$$\begin{aligned}\rho\mathrm{D}_t\mathbf{v} = {}& \rho^{(1)}\left(-\boldsymbol{\nabla}\frac{\partial f}{\partial\rho^{(1)}} - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}}\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v}^{(1)}\right) \\ &+ \rho^{(2)}\left(-\boldsymbol{\nabla}\frac{\partial f}{\partial\rho^{(2)}} + \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}}\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v}^{(2)}\right) \\ &- \rho^{(1)}\mathbf{v}^{(1)}\boldsymbol{\nabla}\cdot\mathbf{v} - \mathbf{v}^{(1)}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{red})}\mathbf{w}\right) \\ &- \rho^{(2)}\mathbf{v}^{(2)}\boldsymbol{\nabla}\cdot\mathbf{v} + \mathbf{v}^{(2)}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{red})}\mathbf{w}\right) + \rho\mathbf{v}\boldsymbol{\nabla}\cdot\mathbf{v}.\end{aligned} \tag{A.48}$$

Using eqs. (A.27) and (A.28) from the previous section, the terms containing the free energy density can be replaced by the pressure gradient and we find

$$\begin{aligned}\rho \mathrm{D}_t \mathbf{v} &= -\boldsymbol{\nabla} P - \rho^{(\mathrm{red})} \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{w} \\ &\quad - \rho \mathbf{v} \boldsymbol{\nabla} \cdot \mathbf{v} - \mathbf{w} \boldsymbol{\nabla} \cdot \left(\rho^{(\mathrm{red})} \mathbf{w} \right) + \rho \mathbf{v} \boldsymbol{\nabla} \cdot \mathbf{v} \\ &= -\boldsymbol{\nabla} P - \rho^{(\mathrm{red})} \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{w} - \mathbf{w} \boldsymbol{\nabla} \cdot \left(\rho^{(\mathrm{red})} \mathbf{w} \right) \\ &= -\boldsymbol{\nabla} P - \boldsymbol{\nabla} \cdot \left(\rho^{(\mathrm{red})} \mathbf{w} \mathbf{w} \right).\end{aligned} \tag{A.49}$$

Relative velocity Consider

$$\begin{aligned}\mathrm{D}_t \mathbf{w} &= \mathrm{D}_t \mathbf{v}^{(1)} - \mathrm{D}_t \mathbf{v}^{(2)} \\ &= -\left(\boldsymbol{\nabla} \frac{\partial f}{\partial \rho^{(1)}} - \boldsymbol{\nabla} \frac{\partial f}{\partial \rho^{(2)}} \right) - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{v}^{(1)} - \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{v}^{(2)}.\end{aligned} \tag{A.50}$$

Using eqs. (A.27) and (A.28), the free energy terms are expressed via the relative chemical potential $\tilde{\mu}$ and the equation is further modified:

$$\begin{aligned}\mathrm{D}_t \mathbf{w} &= -\boldsymbol{\nabla} \tilde{\mu} - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{v}^{(1)} - \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{v}^{(2)} \\ &= -\boldsymbol{\nabla} \tilde{\mu} - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \cdot \boldsymbol{\nabla} \left(\mathbf{v} + \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \right) - \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \cdot \boldsymbol{\nabla} \left(\mathbf{v} - \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \right) \\ &= -\boldsymbol{\nabla} \tilde{\mu} - \left(\frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} + \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \right) \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{v} \\ &\quad - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \cdot \boldsymbol{\nabla} \left(\frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \right) + \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \cdot \boldsymbol{\nabla} \left(\frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \right) \\ &= -\boldsymbol{\nabla} \tilde{\mu} - \mathbf{w} \cdot \boldsymbol{\nabla} \mathbf{v} \\ &\quad - \frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \cdot \boldsymbol{\nabla} \left(\frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} \mathbf{w} \right) + \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \cdot \boldsymbol{\nabla} \left(\frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} \mathbf{w} \right).\end{aligned} \tag{A.51}$$

Next we make use of

$$\frac{\rho^{(\mathrm{red})}}{\rho^{(1)}} = \frac{1-c}{2}, \qquad \frac{\rho^{(\mathrm{red})}}{\rho^{(2)}} = \frac{1+c}{2}, \tag{A.52}$$

to finalize the transformation of the relative velocity dynamics

$$\begin{aligned}
\mathrm{D}_t\mathbf{w} &= -\boldsymbol{\nabla}\tilde{\mu} - \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v} \\
&\quad - \frac{1-c}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1-c}{2}\mathbf{w}\Big) + \frac{1+c}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1+c}{2}\mathbf{w}\Big) \\
&= -\boldsymbol{\nabla}\tilde{\mu} - \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v} \\
&\quad - \frac{1}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1-c}{2}\mathbf{w}\Big) + \frac{c}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1-c}{2}\mathbf{w}\Big) \\
&\quad + \frac{1}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1+c}{2}\mathbf{w}\Big) + \frac{c}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1+c}{2}\mathbf{w}\Big) \\
&= -\boldsymbol{\nabla}\tilde{\mu} - \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v} \\
&\quad - \frac{1}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1-c}{2}\mathbf{w} - \frac{1+c}{2}\mathbf{w}\Big) \\
&\quad + \frac{c}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\Big(\frac{1-c}{2}\mathbf{w} + \frac{1+c}{2}\mathbf{w}\Big) \\
&= -\boldsymbol{\nabla}\tilde{\mu} - \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v} + \frac{1}{2}\mathbf{w}\cdot\boldsymbol{\nabla}(c\mathbf{w}) + \frac{c}{2}\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{w} \\
&= -\boldsymbol{\nabla}\tilde{\mu} - \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v} + \frac{\mathbf{w}}{2}c\cdot\boldsymbol{\nabla}\mathbf{w} + \frac{\mathbf{w}}{2}\mathbf{w}\cdot\boldsymbol{\nabla}c + \frac{\mathbf{w}}{2}c\cdot\boldsymbol{\nabla}\mathbf{w} \\
&= -\boldsymbol{\nabla}\tilde{\mu} - \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v} + c\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{w} + \frac{1}{2}\mathbf{w}\mathbf{w}\cdot\boldsymbol{\nabla}c.
\end{aligned} \tag{A.53}$$

To summarize, the complete set of transformed two-fluid Euler equations is given by

$$\mathrm{D}_t\rho = -\rho\boldsymbol{\nabla}\cdot\mathbf{v}, \tag{A.54}$$

$$\rho\mathrm{D}_t c = -2\boldsymbol{\nabla}\cdot\Big(\rho^{(\mathrm{red})}\mathbf{w}\Big), \tag{A.55}$$

$$\rho\mathrm{D}_t\mathbf{v} = -\boldsymbol{\nabla}P - \boldsymbol{\nabla}\cdot\Big(\rho^{(\mathrm{red})}\mathbf{w}\mathbf{w}\Big), \tag{A.56}$$

$$\mathrm{D}_t\mathbf{w} = -\boldsymbol{\nabla}\tilde{\mu} - \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{v} + c\mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{w} + \frac{1}{2}\mathbf{w}\mathbf{w}\cdot\boldsymbol{\nabla}c. \tag{A.57}$$

A.3 Viscoelastic Model H

Some of the calculations occurring in the derivation of the viscoelastic phase separation model in section 4.1 are outsourced to this appendix section in order to improve the flow of the text. Furthermore, a formulation in terms of flip-invariant tensors is introduced.

A.3.1 Basic Poisson brackets

In order tor derive the equations of motion from the Hamiltonian eq. (4.14), all individual Poisson brackets must be evaluated. From section 2.7 we know that

$$\{f, g\} = \sum_i \left(\frac{\partial f}{\partial x_i} \frac{\partial g}{\partial p_i} - \frac{\partial f}{\partial p_i} \frac{\partial g}{\partial x_i} \right). \tag{A.58}$$

This means, that if f involves the coordinate x_i, g needs to involve the corresponding momenta p_i in order for $\{f, g\}$ to be nonzero and vice versa. Since all quantities $\rho^{(\mathrm{d})}$, $\mathbf{k}^{(\mathrm{r})}$, $\mathbf{j}^{(\mathrm{d})}$ and $\mathbf{j}^{(\mathrm{r})}$ involve the coordinates $\mathbf{R}_i$, and $\mathbf{j}^{(\mathrm{d})}$ is the only quantity involving the $\mathbf{p}_i^{(\mathrm{d})}$, all brackets containing $\mathbf{j}^{(\mathrm{d})}$ are in general nonzero. The relative density $\mathbf{k}^{(\mathrm{r})}$ is the only quantity involving $\mathbf{q}_i$ and the relative current $\mathbf{j}^{(\mathrm{r})}$ is the only quantity involving $\mathbf{p}_i^{(\mathrm{r})}$, hence also $\left\{ k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{r})}(\mathbf{r}') \right\}$ is in general nonzero (see table A.1).

	$\rho^{(\mathrm{d})}$	$k_\beta^{(\mathrm{r})}$	$j_\beta^{(\mathrm{d})}$	$j_\beta^{(\mathrm{r})}$
$\rho^{(\mathrm{d})}$	0	0	x	0
$k_\alpha^{(\mathrm{r})}$	0	0	x	x
$j_\alpha^{(\mathrm{d})}$	x	x	x	x
$j_\alpha^{(\mathrm{r})}$	0	x	x	0

Table A.1: Nonzero Poisson brackets

We proceed by deriving several auxiliary relations that will aid in the derivation of the Poisson brackets. For ease of notation, we introduce the shorthand $\delta_i = \delta(\mathbf{r} - \mathbf{R}_i)$ resp. $\delta_i' = \delta(\mathbf{r}' - \mathbf{R}_i)$ and examine the elementary bracket (sum convention over repeated Greek

indexes is implied)

$$\begin{aligned}\left\{\delta_i, p^{(\mathrm{d})}_{j\beta}\right\} &= \sum_{k\gamma} \frac{\partial\delta(\mathbf{r}-\mathbf{R}_i)}{\partial R_{k\gamma}} \frac{\partial p^{(\mathrm{d})}_{j\beta}}{\partial p^{(\mathrm{d})}_{k\gamma}} \\ &= \sum_{k\gamma} \frac{\partial\delta(\mathbf{r}-\mathbf{R}_i)}{\partial R_{k\gamma}} \delta_{jk}\delta_{\beta\gamma} \\ &= \frac{\partial\delta(\mathbf{r}-\mathbf{R}_i)}{\partial R_{j\beta}} \\ &= -\delta_{ij}\frac{\partial\delta(\mathbf{r}-\mathbf{R}_i)}{\partial r_\beta} = -\delta_{ij}\partial_\beta\delta_i,\end{aligned} \tag{A.59}$$

as well as the more obvious

$$\left\{\delta_i, \delta'_j\right\} = 0, \tag{A.60}$$

$$\left\{q_{i\alpha}, p^{(\mathrm{r})}_{j\beta}\right\} = \delta_{\alpha\beta}\delta_{ij}. \tag{A.61}$$

Next we note that for any quantity A of the form

$$A(\mathbf{r}) = \sum_i a_i\delta(\mathbf{r}-\mathbf{R}_i), \tag{A.62}$$

the relation

$$\begin{aligned}\sum_i a_i\delta'_i f(\mathbf{r},\mathbf{R}_i) &= \sum_i \int \mathrm{d}^d\mathbf{r}''\, a_i\delta(\mathbf{r}''-\mathbf{R}_i)\delta(\mathbf{r}'-\mathbf{r}'')f(\mathbf{r},\mathbf{r}'') \\ &= \int \mathrm{d}^d\mathbf{r}''\,\delta(\mathbf{r}'-\mathbf{r}'')f(\mathbf{r},\mathbf{r}'')\sum_i a_i\delta(\mathbf{r}''-\mathbf{R}_i) \\ &= \int \mathrm{d}^d\mathbf{r}''\,\delta(\mathbf{r}'-\mathbf{r}'')f(\mathbf{r},\mathbf{r}'')A(\mathbf{r}'') = A(\mathbf{r}')f(\mathbf{r},\mathbf{r}')\end{aligned} \tag{A.63}$$

holds. Additionally, if the coefficient a_i neither depends on $\mathbf{R}_i$ nor on $\mathbf{p}^{(\mathrm{d})}_i$, we have

$$\begin{aligned}\left\{A(\mathbf{r}), j^{(\mathrm{d})}_\beta(\mathbf{r}')\right\} &= \sum_{ij}\left\{a_i\delta_i, p^{(\mathrm{d})}_{j\beta}\delta'_j\right\} \\ &= \sum_{ij}\Big[a_i p^{(\mathrm{d})}_{j\beta}\left\{\delta_i,\delta'_j\right\} + a_i\delta'_j\left\{\delta_i, p^{(\mathrm{d})}_{j\beta}\right\} \\ &\quad + \delta_i p^{(\mathrm{d})}_{j\beta}\left\{a_i,\delta'_j\right\} + \delta_i\delta'_j\left\{a_i, p^{(\mathrm{d})}_{j\beta}\right\}\Big] \\ &= \sum_{ij} a_i\delta'_j\left\{\delta_i, p^{(\mathrm{d})}_{j\beta}\right\} = -\sum_i a_i\delta(\mathbf{r}'-\mathbf{R}_i)\partial_\beta\delta(\mathbf{r}-\mathbf{R}_i) \\ &= -A(\mathbf{r}')\partial_\beta\delta(\mathbf{r}-\mathbf{r}').\end{aligned} \tag{A.64}$$

In particular, this immediately yields the brackets

$$\Big\{\rho^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\Big\} = -\rho^{(\mathrm{d})}(\mathbf{r}')\partial_\beta\delta(\mathbf{r}-\mathbf{r}'), \tag{A.65}$$

$$\Big\{k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\Big\} = -k_\alpha^{(\mathrm{r})}(\mathbf{r}')\partial_\beta\delta(\mathbf{r}-\mathbf{r}'), \tag{A.66}$$

$$\Big\{j_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\Big\} = -j_\alpha^{(\mathrm{r})}(\mathbf{r}')\partial_\beta\delta(\mathbf{r}-\mathbf{r}'). \tag{A.67}$$

For the bracket containing the mixed Cartesian component dumbbell currents we find

$$\begin{aligned}\Big\{j_\alpha^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\Big\} &= \sum_{ij}\Big\{p_{i\alpha}^{(\mathrm{d})}\delta_i, p_{j\beta}^{(\mathrm{d})}\delta_j'\Big\}\\ &= \sum_{ij}\Big[\delta_i p_{j\beta}^{(\mathrm{d})}\Big\{p_{i\alpha}^{(\mathrm{d})}, \delta_j'\Big\} + p_{i\alpha}^{(\mathrm{d})}\delta_j'\Big\{\delta_i, p_{j\beta}^{(\mathrm{d})}\Big\}\Big]\\ &= \sum_{i}\Big[\delta_i p_{i\beta}^{(\mathrm{d})}\partial_\alpha'\delta_i' - p_{i\alpha}^{(\mathrm{d})}\delta_i'\partial_\beta\delta_i\Big]\\ &= j_\beta^{(\mathrm{d})}(\mathbf{r})\partial_\alpha'\delta(\mathbf{r}-\mathbf{r}') - j_\alpha^{(\mathrm{d})}(\mathbf{r}')\partial_\beta\delta(\mathbf{r}-\mathbf{r}').\end{aligned} \tag{A.68}$$

Finally, there is

$$\begin{aligned}\Big\{k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{r})}(\mathbf{r}')\Big\} &= k\sum_{ij}\Big\{q_{i\alpha}\delta_i, p_{j\beta}^{(\mathrm{r})}\delta_j'\Big\} = k\sum_{ij}\delta_i\delta_j'\Big\{q_{i\alpha}, p_{j\beta}^{(\mathrm{r})}\Big\}\\ &= k\sum_{ij}\delta_i\delta_j'\delta_{\alpha\beta}\delta_{ij} = \delta_{\alpha\beta}\frac{k}{m^{(\mathrm{d})}}\sum_i\delta_i m^{(\mathrm{d})}\delta_i'\\ &= \delta_{\alpha\beta}\frac{k}{m^{(\mathrm{d})}}\rho^{(\mathrm{d})}(\mathbf{r}')\delta(\mathbf{r}-\mathbf{r}').\end{aligned} \tag{A.69}$$

We will also find useful the similar relations

$$\begin{aligned}\int \mathrm{d}^d\mathbf{r}'\,\Big\{j_\alpha^{(\mathrm{d})}(\mathbf{r}), j_\beta^{(\mathrm{d})}(\mathbf{r}')\Big\}f(\mathbf{r}') &= \int \mathrm{d}^d\mathbf{r}'\, j_\beta^{(\mathrm{d})}(\mathbf{r})\partial_\alpha'\delta(\mathbf{r}-\mathbf{r}')f(\mathbf{r}')\\ &\quad - \int \mathrm{d}^d\mathbf{r}'\, j_\alpha^{(\mathrm{d})}(\mathbf{r}')\partial_\beta\delta(\mathbf{r}-\mathbf{r}')f(\mathbf{r}')\\ &= -j_\beta^{(\mathrm{d})}(\mathbf{r})\partial_\alpha f(\mathbf{r}) - \partial_\beta\Big(j_\alpha^{(\mathrm{d})}(\mathbf{r})f(\mathbf{r})\Big),\end{aligned} \tag{A.70}$$

$$\begin{aligned}\int \mathrm{d}^d\mathbf{r}'\,\Big\{k_\alpha^{(\mathrm{r})}(\mathbf{r}), j_\beta^{(\mathrm{r})}(\mathbf{r}')\Big\}f(\mathbf{r}') &= \delta_{\alpha\beta}\frac{k}{m^{(\mathrm{d})}}\int \mathrm{d}^d\mathbf{r}'\,\rho^{(\mathrm{d})}(\mathbf{r}')\delta(\mathbf{r}-\mathbf{r}')f(\mathbf{r}')\\ &= \delta_{\alpha\beta}\frac{k}{m^{(\mathrm{d})}}\rho^{(\mathrm{d})}(\mathbf{r})f(\mathbf{r}),\end{aligned} \tag{A.71}$$

$$\begin{aligned}\int \mathrm{d}^d\mathbf{r}'\,\Big\{j_\beta^{(\mathrm{r})}(\mathbf{r}), k_\alpha^{(\mathrm{r})}(\mathbf{r}')\Big\}f(\mathbf{r}') &= -\delta_{\alpha\beta}\frac{k}{m^{(\mathrm{d})}}\rho^{(\mathrm{d})}(\mathbf{r})\int \mathrm{d}^d\mathbf{r}'\,\delta(\mathbf{r}-\mathbf{r}')f(\mathbf{r}')\\ &= -\delta_{\alpha\beta}\frac{k}{m^{(\mathrm{d})}}\rho^{(\mathrm{d})}(\mathbf{r})f(\mathbf{r}).\end{aligned} \tag{A.72}$$

A.3.2 Dissipation rate

The calculation of the dissipation rate in this section is based on the approximated equations of motion which where derived in section 4.1.6 and read

$$\boldsymbol{\nabla}\cdot\mathbf{v}=0, \tag{A.73}$$
$$D_t\rho^{(\mathrm{d})}=-\boldsymbol{\nabla}\cdot\mathbf{j}^{(\mathrm{int})}, \tag{A.74}$$
$$\rho D_t\mathbf{v}=-\boldsymbol{\nabla}P+\eta_{\mathrm{s}}\nabla^2\mathbf{v}+\kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\nabla^2\rho^{(\mathrm{d})} \tag{A.75}$$
$$+\frac{\Gamma}{\tau\tau_q}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{q}\mathbf{q}\right)+\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}, \tag{A.76}$$
$$D_t\mathbf{q}=\mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}-\frac{1}{\tau_q}\mathbf{q}+\mathbf{Q}. \tag{A.77}$$

The terms $\boldsymbol{\nabla}\cdot\mathbf{j}^{(\mathrm{int})}$, $\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}$, and $\mathbf{Q}$ are phenomenological ones introduced to adiabatically eliminate the relative velocity $\mathbf{w}$. For the non-phenomenological terms, dissipative terms can be identified from their sign-changing behavior under time reversal. For example, $\rho\mathrm{D}_t\mathbf{v}$ does not change sign whereas $\eta_{\mathrm{s}}\nabla^2\mathbf{v}$ does, revealing that it is dissipative in nature. To prove that all phenomenological terms are dissipative as well, the reduced set of equations of motion without phenomenological and knowingly dissipative terms is shown to be conservative. The thus reduced set of equations is

$$\boldsymbol{\nabla}\cdot\mathbf{v}=0, \tag{A.78}$$
$$D_t\rho^{(\mathrm{d})}=0, \tag{A.79}$$
$$\rho D_t\mathbf{v}=-\boldsymbol{\nabla}P+\kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\nabla^2\rho^{(\mathrm{d})}+\frac{\Gamma}{\tau\tau_q}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{q}\mathbf{q}\right), \tag{A.80}$$
$$D_t\mathbf{q}=\mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v}. \tag{A.81}$$

The Hamiltonian of this system is given by

$$\mathscr{H}=\int\mathrm{d}^d\mathbf{r}\left[\frac{\rho}{2}\mathbf{v}^2+f+\frac{\kappa}{2}\left(\boldsymbol{\nabla}\rho^{(\mathrm{d})}\right)^2+\frac{1}{2}\frac{\Gamma}{\tau\tau_q}\rho^{(\mathrm{d})}\mathbf{q}^2\right], \tag{A.82}$$

with derivatives

$$\frac{\delta H}{\delta\mathbf{v}}=\rho\mathbf{v}, \tag{A.83}$$
$$\frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}}=\frac{\partial f}{\partial\rho^{(\mathrm{d})}}-\kappa\nabla^2\rho^{(\mathrm{d})}+\frac{1}{2}\frac{\Gamma}{\tau\tau_q}\mathbf{q}^2, \tag{A.84}$$
$$\frac{\delta\mathscr{H}}{\delta\mathbf{q}}=\frac{\Gamma}{\tau\tau_q}\rho^{(\mathrm{d})}\mathbf{q}. \tag{A.85}$$

Taking the time derivative and applying the chain rule results in

$$\frac{\mathrm{d}\mathscr{H}}{\mathrm{d}t} = \int \mathrm{d}^d\mathbf{r} \left[\frac{\delta\mathscr{H}}{\delta\mathbf{v}} \cdot \partial_t\mathbf{v} + \frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}}\partial_t\rho^{(\mathrm{d})} + \frac{\delta\mathscr{H}}{\delta\mathbf{q}} \cdot \partial_t\mathbf{q} \right]. \tag{A.86}$$

Using integration by parts and incompressibility $\boldsymbol{\nabla}\cdot\mathbf{v} = 0$, the integral

$$\int \mathrm{d}^d\mathbf{r}\, \psi\mathbf{v}\cdot\boldsymbol{\nabla}\psi = -\int \mathrm{d}^d\mathbf{r}\, \psi\mathbf{v}\cdot\boldsymbol{\nabla}\psi \tag{A.87}$$

for any field $\psi(\mathbf{r}, t)$ is equal to its own negative and must therefore vanish entirely. This immediately implies that for the convective derivative $\mathrm{D}_t = \partial_t + \mathbf{v}\cdot\boldsymbol{\nabla}$

$$\int \mathrm{d}^d\mathbf{r}\, \psi\mathrm{D}_t\psi = \int \mathrm{d}^d\mathbf{r}\, \psi\partial_t\psi. \tag{A.88}$$

Inserting eqs. (A.80) and (A.83), the first term in the dissipation rate eq. (A.86) becomes

$$\begin{aligned} \int \mathrm{d}^d\mathbf{r}\, \rho\mathbf{v}\cdot\mathrm{D}_t\mathbf{v} &= \int \mathrm{d}^d\mathbf{r}\, \mathbf{v}\cdot\left[-\boldsymbol{\nabla}P + \kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\nabla^2\rho^{(\mathrm{d})} + \frac{\Gamma}{\tau\tau_q}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{q}\mathbf{q}\right)\right] \\ &= \int \mathrm{d}^d\mathbf{r}\, \mathbf{v}\cdot\left[\kappa\rho^{(\mathrm{d})}\boldsymbol{\nabla}\nabla^2\rho^{(\mathrm{d})} + \frac{\Gamma}{\tau\tau_q}\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{q}\mathbf{q}\right)\right], \end{aligned} \tag{A.89}$$

where incompressibility and integration by parts were used to eliminate the pressure term.

The second term in the dissipation rate is

$$\begin{aligned} &\int \mathrm{d}^d\mathbf{r} \left[\frac{\partial f}{\partial\rho^{(\mathrm{d})}} - \kappa\nabla^2\rho^{(\mathrm{d})} + \frac{1}{2}\frac{\Gamma}{\tau\tau_q}\mathbf{q}^2 \right](\mathrm{D}_t - \mathbf{v}\cdot\boldsymbol{\nabla})\rho^{(\mathrm{d})} \\ &= -\int \mathrm{d}^d\mathbf{r} \left[\frac{\partial f}{\partial\rho^{(\mathrm{d})}} - \kappa\nabla^2\rho^{(\mathrm{d})} + \frac{1}{2}\frac{\Gamma}{\tau\tau_q}\mathbf{q}^2 \right]\mathbf{v}\cdot\boldsymbol{\nabla}\rho^{(\mathrm{d})} \\ &= \int \mathrm{d}^d\mathbf{r}\, \rho^{(\mathrm{d})}\mathbf{v}\cdot\boldsymbol{\nabla}\left[\frac{\partial f}{\partial\rho^{(\mathrm{d})}} - \kappa\nabla^2\rho^{(\mathrm{d})} + \frac{1}{2}\frac{\Gamma}{\tau\tau_q}\mathbf{q}^2 \right]. \end{aligned} \tag{A.90}$$

With the dumbbell pressure gradient given by

$$\boldsymbol{\nabla}P^{(\mathrm{d})} = \rho^{(\mathrm{d})}\boldsymbol{\nabla}\frac{\partial f}{\partial\rho^{(\mathrm{d})}}, \tag{A.91}$$

the free energy term can be eliminated again using incompressibility

$$\int \mathrm{d}^d\mathbf{r}\, \rho^{(\mathrm{d})}\mathbf{v}\cdot\frac{\partial f}{\partial\rho^{(\mathrm{d})}} = \int \mathrm{d}^d\mathbf{r}\, \mathbf{v}\cdot\boldsymbol{\nabla}P^{(\mathrm{d})} = 0. \tag{A.92}$$

The second term in the dissipation rate then reduces to

$$\int \mathrm{d}^d\mathbf{r}\,\frac{\delta\mathscr{H}}{\delta\rho^{(\mathrm{d})}}\partial_t\rho^{(\mathrm{d})} = -\int \mathrm{d}^d\mathbf{r}\,\rho^{(\mathrm{d})}\mathbf{v}\cdot\boldsymbol{\nabla}\left[\kappa\nabla^2\rho^{(\mathrm{d})} - \frac{1}{2}\frac{\Gamma}{\tau\tau_q}\mathbf{q}^2\right]. \tag{A.93}$$

For the third term we find

$$\frac{\Gamma}{\tau\tau_q}\int \mathrm{d}^d\mathbf{r}\,\rho^{(\mathrm{d})}\mathbf{q}\cdot(\mathrm{D}_t - \mathbf{v}\cdot\boldsymbol{\nabla})\mathbf{q} = \frac{\Gamma}{\tau\tau_q}\int \mathrm{d}^d\mathbf{r}\,\rho^{(\mathrm{d})}\mathbf{q}\cdot(\mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v} - \mathbf{v}\cdot\boldsymbol{\nabla}\mathbf{q}). \tag{A.94}$$

Adding up all contributions to the dissipation rate, the interfacial terms cancel and it remains:

$$\begin{aligned}\frac{\mathrm{d}\mathscr{H}}{\mathrm{d}t} &= \frac{\Gamma}{(\tau\tau_q)}\int \mathrm{d}^d\mathbf{r}\left[\mathbf{v}\cdot\left(\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{q}\mathbf{q}\right)\right) + \frac{\rho^{(\mathrm{d})}}{2}\mathbf{v}\cdot\boldsymbol{\nabla}\mathbf{q}^2 + \rho^{(\mathrm{d})}\mathbf{q}\cdot(\mathbf{q}\cdot\boldsymbol{\nabla}\mathbf{v} - \mathbf{v}\cdot\boldsymbol{\nabla}\mathbf{q})\right] \\ &= \frac{\Gamma}{(\tau\tau_q)}\int \mathrm{d}^d\mathbf{r}\left[\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{v}\cdot\mathbf{q}\mathbf{q}\right) + \frac{\rho^{(\mathrm{d})}}{2}\mathbf{v}\cdot\boldsymbol{\nabla}\mathbf{q}^2 - \rho^{(\mathrm{d})}\mathbf{q}\cdot(\mathbf{v}\cdot\boldsymbol{\nabla}\mathbf{q})\right] \\ &= \frac{\Gamma}{(\tau\tau_q)}\int \mathrm{d}^d\mathbf{r}\left[\boldsymbol{\nabla}\cdot\left(\rho^{(\mathrm{d})}\mathbf{v}\cdot\mathbf{q}\mathbf{q}\right)\right] = 0,\end{aligned} \tag{A.95}$$

where Gauss' integral theorem was used to eliminate the last remaining term. This proves that the system of equations (A.78) to (A.81) is indeed conservative and the phenomenological terms $\boldsymbol{\nabla}\cdot\mathbf{j}^{(\mathrm{int})}$, $\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}$, and $\mathbf{Q}$ are of dissipative nature.

A.3.3 Formulation with flip-invariant tensors

Instead of the end-to-end vector $\mathbf{q} = \mathbf{r}^{(1)} - \mathbf{r}^{(2)}$ of a dumbbell, which changes sign upon exchange of indexes, one can consider the flip invariant conformation tensor

$$C_{\alpha\beta} = q_\alpha q_\beta \tag{A.96}$$

to describe the orientation. The fact that $\mathbf{P} := \mathbf{C}/\operatorname{Tr}\mathbf{C}$ is symmetric ($\mathbf{P}^{\mathsf{T}} = \mathbf{P}$) and idempotent ($\mathbf{P}^2 = \mathbf{P}$) makes it a projection operator. Therefore, $\mathbf{C}$ needs to satisfy the constraints

$$0 = \mathbf{C} - \mathbf{C}^{\mathsf{T}}, \tag{A.97}$$

$$0 = \mathbf{C}^2 - \mathbf{C}\operatorname{Tr}\mathbf{C}. \tag{A.98}$$

Note that multiplication with an arbitrary vector $\mathbf{n} \neq 0$ results in a vector parallel to $\mathbf{q}$:

$$\mathbf{C}\cdot\mathbf{n} = \mathbf{q}(\mathbf{q}\cdot\mathbf{n}) \propto \mathbf{q}. \tag{A.99}$$

Therefore, the orientation of a dumbbell can be extracted from the conformation tensor via

$$\pm\mathbf{q} = \sqrt{\operatorname{Tr}\mathbf{C}}\frac{\mathbf{C}\cdot\mathbf{n}}{|\mathbf{C}\cdot\mathbf{n}|}. \tag{A.100}$$

$\mathbf{P}$ projects onto a one dimensional space and therefore has only one eigenvalue different from zero, which in this case is 1. The arbitrariness in the choice of $\mathbf{n}$ can be removed by setting $\mathbf{n}$ to the eigenvector $\mathbf{u}_1^{(\mathbf{C})}$ corresponding to the non-trivial eigenvalue. Then $\mathbf{C}\cdot\mathbf{u}_1^{(\mathbf{C})} = \mathbf{u}_1^{(\mathbf{C})}$ and

$$\pm\mathbf{q} = \sqrt{\operatorname{Tr}\mathbf{C}}\frac{\mathbf{u}_1^{(\mathbf{C})}}{\left|\mathbf{u}_1^{(\mathbf{C})}\right|}. \tag{A.101}$$

Hence, the direction of $\mathbf{q}$ is given by the eigenvector $\mathbf{u}_1^{(\mathbf{C})}$ and its magnitude by $\sqrt{\operatorname{Tr}\mathbf{C}}$. A tensor field corresponding to the dumbbell orientation can be defined by

$$\mathbf{K}(\mathbf{r}) = k\sum_i \mathbf{C}_i\delta(\mathbf{r}-\mathbf{R}_i). \tag{A.102}$$

The elastic energy stored in the dumbbell continuum is then simply given by

$$E_{\text{pot}} = \frac{1}{2}\int \mathrm{d}^d\mathbf{r}\,\operatorname{Tr}\mathbf{K}(\mathbf{r}). \tag{A.103}$$

Apart form the conformation, another flip-invariant variable to describe the dynamics is needed. One possible candidate capturing the conformation dynamics is the tensor

$$S_{\alpha\beta} = \dot{q}_\alpha q_\beta \tag{A.104}$$

along with its transpose $\mathbf{S}^\mathsf{T}$. The time derivative of the conformation tensor is then $\dot{\mathbf{C}} = \mathbf{S} + \mathbf{S}^\mathsf{T}$. Normalized by the trace, we receive an idempotent but not symmetric operator

$$\left(\frac{\mathbf{S}}{\operatorname{Tr}\mathbf{S}}\right)^2_{\alpha\beta} = \frac{S_{\alpha\gamma}}{\operatorname{Tr}\mathbf{S}}\frac{S_{\gamma\beta}}{\operatorname{Tr}\mathbf{S}} = \frac{\dot{q}_\alpha q_\gamma \dot{q}_\gamma q_\beta}{(\operatorname{Tr}\mathbf{S})^2} = \frac{\dot{q}_\alpha q_\beta}{\operatorname{Tr}\mathbf{S}} = \frac{S_{\alpha\beta}}{\operatorname{Tr}\mathbf{S}}, \tag{A.105}$$

again projecting onto a one-dimensional space. Hence it needs to satisfy the constraint

$$0 = \mathbf{S}^2 - \mathbf{S}\operatorname{Tr}\mathbf{S}. \tag{A.106}$$

Like $\mathbf{q}$ from $\mathbf{C}$, the expansion velocity $\dot{\mathbf{q}}$ can be extracted from $\mathbf{S}$ via some vector $\mathbf{n} \neq 0$:

$$\pm\dot{\mathbf{q}} = \frac{\mathbf{S}\cdot\mathbf{n}}{\mathbf{q}\cdot\mathbf{n}}. \tag{A.107}$$

With $\mathbf{u}_1^{(\mathbf{S})}$ being the eigenvector of $\mathbf{S}$ corresponding to the non-trivial eigenvalue, this becomes

$$\pm\dot{\mathbf{q}} = \frac{\mathbf{u}_1^{(\mathbf{S})}}{\mathbf{q}\cdot\mathbf{u}_1^{(\mathbf{S})}}, \tag{A.108}$$

where $\mathbf{q}$ can be calculated from $\mathbf{C}$ according to eq. (A.101).

Because $\mathrm{Tr}\big(\mathbf{S}\mathbf{S}^{\mathsf{T}}\big) = \dot{\mathbf{q}}^2\mathbf{q}^2$, the absolute value of $\dot{\mathbf{q}}$ can be calculated by

$$\dot{\mathbf{q}}^2 = \frac{\mathrm{Tr}\big(\mathbf{S}\mathbf{S}^{\mathsf{T}}\big)}{\mathrm{Tr}\,\mathbf{C}}. \tag{A.109}$$

This means that the kinetic energy in the relative motion of one dumbbell is

$$E_{\mathrm{kin}} = \frac{m^{(\mathrm{r})}}{2}\frac{\mathrm{Tr}\big(\mathbf{S}\mathbf{S}^{\mathsf{T}}\big)}{\mathrm{Tr}\,\mathbf{C}}. \tag{A.110}$$

With a corresponding tensor field defined by

$$\mathbf{J}(\mathbf{r}) = m^{(\mathrm{r})}\sum_i \mathbf{S}_i\delta(\mathbf{r}-\mathbf{R}_i), \tag{A.111}$$

the continuum representation of the connectivity related Hamiltonian has the form

$$\mathscr{H}^{(\mathrm{r})} = \frac{1}{2}\,\mathrm{Tr}\int \mathrm{d}^d\mathbf{r}\left[\mathbf{K} + \frac{k}{m^{(\mathrm{r})}}\frac{\mathbf{J}\mathbf{J}^{\mathsf{T}}}{\mathrm{Tr}\,\mathbf{K}}\right]. \tag{A.112}$$

Poisson brackets

In this section, the Poisson brackets involving the macroscopic tensor fields $\mathbf{K}$ and $\mathbf{J}$ are calculated. First, consider the microscopic bracket of $\mathbf{C}$ with $\mathbf{S}$:

$$\begin{aligned}\{C_{\alpha\beta}, S_{\gamma\delta}\} &= q_\alpha\{q_\beta, \dot{q}_\gamma\}q_\delta + q_\beta\{q_\alpha, \dot{q}_\gamma\}q_\delta \\ &= \frac{1}{m^{(\mathrm{r})}}(q_\alpha q_\delta\delta_{\beta\gamma} + q_\beta q_\delta\delta_{\alpha\gamma}) = \frac{1}{m^{(\mathrm{r})}}(C_{\alpha\delta}\delta_{\beta\gamma} + C_{\beta\delta}\delta_{\alpha\gamma}).\end{aligned} \tag{A.113}$$

With the tensor of order $2n$ defined by the outer product

$$T^{\beta_1\beta_2\ldots\beta_n}_{\alpha_1\alpha_2\ldots\alpha_n} = \delta_{\alpha_1\beta_1}\delta_{\alpha_2\beta_2}\ldots\delta_{\alpha_n\beta_n}, \tag{A.114}$$

which substitutes lower for upper indexes upon contraction, this can alternatively be written as

$$\begin{aligned}\{C_{\alpha\beta}, S_{\gamma\delta}\} &= \frac{C_{\mu\nu}}{m^{(\mathrm{r})}}(\delta_{\mu\alpha}\delta_{\nu\delta}\delta_{\beta\gamma} + \delta_{\mu\beta}\delta_{\nu\delta}\delta_{\alpha\gamma}) \\ &= \frac{1}{m^{(\mathrm{r})}}\big(T^{\alpha\delta\gamma}_{\mu\nu\beta} + T^{\beta\delta\gamma}_{\mu\nu\alpha}\big)C_{\mu\nu} =: \frac{1}{m^{(\mathrm{r})}}E^{\alpha\beta\gamma\delta}_{\mu\nu}C_{\mu\nu}.\end{aligned} \tag{A.115}$$

Hence, the Poisson bracket of $\mathbf{C}$ with $\mathbf{S}$ can be written as a linear combination of the components of $\mathbf{C}$.

The second non-zero microscopic Poisson bracket is the one of $\mathbf{S}$ with itself:

$$\begin{aligned}\{S_{\alpha\beta}, S_{\gamma\delta}\} &= \{\dot{q}_\alpha q_\beta, \dot{q}_\gamma q_\delta\} = \dot{q}_\alpha \{q_\beta, \dot{q}_\gamma\} q_\delta + q_\beta \{\dot{q}_\alpha, q_\delta\} \dot{q}_\gamma \\ &= \frac{1}{m^{(\mathrm{r})}} (\dot{q}_\alpha q_\delta \delta_{\beta\gamma} - \dot{q}_\gamma q_\beta \delta_{\alpha\delta}) \\ &= \frac{1}{m^{(\mathrm{r})}} \Big(T^{\alpha\delta\gamma}_{\mu\nu\beta} - T^{\gamma\beta\delta}_{\mu\nu\alpha} \Big) S_{\mu\nu} =: \frac{1}{m^{(\mathrm{r})}} F^{\alpha\beta\gamma\delta}_{\mu\nu} S_{\mu\nu},\end{aligned} \tag{A.116}$$

which can again be written as a linear combination of the components of $\mathbf{S}$.

With the results above, the Poisson brackets for the fields $\mathbf{J}$ and $\mathbf{K}$ can be evaluated. Using eqs. (A.63) and (A.115), we can determine

$$\begin{aligned}\{K_{\alpha\beta}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}')\} &= k m^{(\mathrm{r})} \sum_{ij} \{C_{i\alpha\beta} \delta(\mathbf{r} - \mathbf{R}_i), S_{j\gamma\delta} \delta(\mathbf{r}' - \mathbf{R}_j)\} \\ &= k m^{(\mathrm{r})} \sum_i \{C_{i\alpha\beta} \delta_i, S_{i\gamma\delta} \delta'_i\} = k m^{(\mathrm{r})} \sum_i \delta_i \delta'_i \{C_{i\alpha\beta}, S_{i\gamma\delta}\} \\ &= k \sum_i \delta_i \delta'_i E^{\alpha\beta\gamma\delta}_{\mu\nu} C_{i\mu\nu} = E^{\alpha\beta\gamma\delta}_{\mu\nu} K_{\mu\nu}(\mathbf{r}') \delta(\mathbf{r} - \mathbf{r}').\end{aligned} \tag{A.117}$$

Analogously, the Poisson bracket of $\mathbf{J}$ with itself becomes

$$\begin{aligned}\{J_{\alpha\beta}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}')\} &= \Big(m^{(\mathrm{r})}\Big)^2 \sum_i \delta_i \delta'_i \{S_{i\alpha\beta}, S_{i\gamma\delta}\} = m^{(\mathrm{r})} F^{\alpha\beta\gamma\delta}_{\mu\nu} \sum_i \delta_i \delta'_i S_{\mu\nu} \\ &= F^{\alpha\beta\gamma\delta}_{\mu\nu} J_{\mu\nu}(\mathbf{r}') \delta(\mathbf{r} - \mathbf{r}').\end{aligned} \tag{A.118}$$

Furthermore, there are the brackets of the form $\Big\{\mathbf{K}, \mathbf{j}^{(\mathrm{d})}\Big\}$ and $\Big\{\mathbf{J}, \mathbf{j}^{(\mathrm{d})}\Big\}$, which can be evaluated using the helping relation eq. (A.64). This results in

$$\Big\{K_{\alpha\beta}(\mathbf{r}), j^{(\mathrm{d})}_\gamma(\mathbf{r}')\Big\} = -K_{\alpha\beta}(\mathbf{r}') \partial_\gamma \delta(\mathbf{r} - \mathbf{r}'), \tag{A.119}$$

$$\Big\{J_{\alpha\beta}(\mathbf{r}), j^{(\mathrm{d})}_\gamma(\mathbf{r}')\Big\} = -J_{\alpha\beta}(\mathbf{r}') \partial_\gamma \delta(\mathbf{r} - \mathbf{r}'). \tag{A.120}$$

Thus, the description is complete and the set of Poisson brackets is closed.

Summary The functional derivatives are obtained from the continuum Hamiltonian

$$\mathscr{H}^{(\mathrm{r})} = \frac{1}{2} \operatorname{Tr} \int \mathrm{d}^d \mathbf{r} \left[\mathbf{K} + \frac{k}{m^{(\mathrm{r})}} \frac{\mathbf{J}\mathbf{J}^{\mathsf{T}}}{\operatorname{Tr} \mathbf{K}} \right], \tag{A.121}$$

and read

$$\frac{\delta\mathscr{H}^{(\mathrm{r})}}{\delta\mathbf{J}} = \frac{k}{m^{(\mathrm{r})}}\frac{\mathbf{J}}{\operatorname{Tr}\mathbf{K}} =: \mathbf{G}, \tag{A.122}$$

$$\frac{\delta\mathscr{H}^{(\mathrm{r})}}{\delta\mathbf{K}} = \frac{\mathbb{1}}{2}\left[1 - \frac{k}{m^{(\mathrm{r})}}\frac{\operatorname{Tr}\left(\mathbf{J}\mathbf{J}^{\mathsf{T}}\right)}{\operatorname{Tr}^2\mathbf{K}}\right] = \frac{\mathbb{1}}{2}\left[1 - \frac{m^{(\mathrm{r})}}{k}\operatorname{Tr}\left(\mathbf{G}\mathbf{G}^{\mathsf{T}}\right)\right]. \tag{A.123}$$

Here, a new tensor field $\mathbf{G}(\mathbf{r},t)$ with units of a velocity gradient has been defined. The set of non-zero Poisson brackets is given by

$$\{K_{\alpha\beta}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}')\} = E^{\alpha\beta\gamma\delta}_{\mu\nu}K_{\mu\nu}(\mathbf{r}')\delta(\mathbf{r}-\mathbf{r}'), \tag{A.124}$$

$$\{J_{\alpha\beta}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}')\} = F^{\alpha\beta\gamma\delta}_{\mu\nu}J_{\mu\nu}(\mathbf{r}')\delta(\mathbf{r}-\mathbf{r}'), \tag{A.125}$$

$$\left\{K_{\alpha\beta}(\mathbf{r}), j^{(\mathrm{d})}_{\gamma}(\mathbf{r}')\right\} = -K_{\alpha\beta}(\mathbf{r}')\partial_\gamma\delta(\mathbf{r}-\mathbf{r}'), \tag{A.126}$$

$$\left\{J_{\alpha\beta}(\mathbf{r}), j^{(\mathrm{d})}_{\gamma}(\mathbf{r}')\right\} = -J_{\alpha\beta}(\mathbf{r}')\partial_\gamma\delta(\mathbf{r}-\mathbf{r}'). \tag{A.127}$$

Equations of motion

According to section 2.7 eq. (2.185), the conservative equations of motions can be calculated from the Poisson brackets and the functional derivatives of the Hamiltonian by

$$\left(\frac{\partial\psi}{\partial t}\right)_{\mathrm{cons}} = \sum_j \int \mathrm{d}^d\mathbf{r}' \,\{\psi(\mathbf{r}), \phi_j(\mathbf{r}')\}\frac{\delta\mathscr{H}}{\delta\phi_j(\mathbf{r}')}. \tag{A.128}$$

For the conformation tensor field, this is

$$\begin{aligned}&\left(\frac{\partial K_{\alpha\beta}}{\partial t}\right)_{\mathrm{cons}}\\ &= \int \mathrm{d}^d\mathbf{r}' \left[\{K_{\alpha\beta}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}')\}\frac{\delta\mathscr{H}}{\delta J_{\gamma\delta}(\mathbf{r}')} + \left\{K_{\alpha\beta}(\mathbf{r}), j^{(\mathrm{d})}_{\gamma}(\mathbf{r}')\right\}\frac{\delta\mathscr{H}}{\delta j^{(\mathrm{d})}_{\gamma}(\mathbf{r}')}\right].\end{aligned} \tag{A.129}$$

Using eqs. (A.122), (A.124), and (A.126), as well as $\delta\mathscr{H}\big/\delta\mathbf{j}^{(\mathrm{d})} = \mathbf{v}^{(\mathrm{d})}$, this becomes

$$\begin{aligned}\left(\frac{\partial K_{\alpha\beta}}{\partial t}\right)_{\mathrm{cons}} &= E^{\alpha\beta\gamma\delta}_{\mu\nu}\int \mathrm{d}^d\mathbf{r}'\,\delta(\mathbf{r}-\mathbf{r}')K_{\mu\nu}(\mathbf{r}')G_{\gamma\delta}\\ &\quad - \partial_\gamma\int \mathrm{d}^d\mathbf{r}'\,\delta(\mathbf{r}-\mathbf{r}')K_{\alpha\beta}(\mathbf{r}')v^{(\mathrm{d})}_{\gamma}(\mathbf{r}')\\ &= E^{\alpha\beta\gamma\delta}_{\mu\nu}K_{\mu\nu}(\mathbf{r})G_{\gamma\delta}(\mathbf{r}) - \partial_\gamma\Big(K_{\alpha\beta}(\mathbf{r})v^{(\mathrm{d})}_{\gamma}(\mathbf{r})\Big).\end{aligned} \tag{A.130}$$

By explicitly evaluating the term

$$\begin{aligned} E^{\alpha\beta\gamma\delta}_{\mu\nu} K_{\mu\nu} G_{\gamma\delta} &= \left(T^{\alpha\delta\gamma}_{\mu\nu\beta} + T^{\beta\delta\gamma}_{\mu\nu\alpha}\right) K_{\mu\nu} G_{\gamma\delta} = (K_{\alpha\delta}\delta_{\beta\gamma} + K_{\beta\delta}\delta_{\alpha\gamma}) G_{\gamma\delta} \\ &= K_{\alpha\delta} G_{\beta\delta} + K_{\beta\delta} G_{\alpha\delta} = K_{\alpha\delta} G^{\mathsf{T}}_{\delta\beta} + G_{\alpha\delta} K^{\mathsf{T}}_{\delta\beta}, \end{aligned} \tag{A.131}$$

the equation of motion in tensor notation can be written as

$$\left(\frac{\partial \mathbf{K}}{\partial t}\right)_{\text{cons}} = -\boldsymbol{\nabla} \cdot \left(\mathbf{v}^{(\mathrm{d})} \mathbf{K}\right) + \mathbf{K}\mathbf{G}^{\mathsf{T}} + \mathbf{G}\mathbf{K}, \tag{A.132}$$

where the symmetry $\mathbf{K}^{\mathsf{T}} = \mathbf{K}$ was used. This form is somewhat reminiscent of the upper-convective derivative defined in eq. (2.152). While $\mathbf{K}$ is convected by $\mathbf{v}^{(\mathrm{d})}$, its deformation happens according to $\mathbf{G}$.

For the tensor field $\mathbf{J}$, which describes the conformation dynamics, the equation of motion becomes

$$\begin{aligned} &\left(\frac{\partial J_{\alpha\beta}}{\partial t}\right)_{\text{cons}} \\ &\quad = \int \mathrm{d}^d\mathbf{r}' \left[\{J_{\alpha\beta}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}')\} \frac{\delta \mathscr{H}}{\delta J_{\gamma\delta}(\mathbf{r}')} + \left\{J_{\alpha\beta}(\mathbf{r}), j^{(\mathrm{d})}_{\gamma}(\mathbf{r}')\right\} \frac{\delta \mathscr{H}}{\delta j^{(\mathrm{d})}_{\gamma}(\mathbf{r}')} \right] \\ &\quad = F^{\alpha\beta\gamma\delta}_{\mu\nu} \int \mathrm{d}^d\mathbf{r}\, J_{\mu\nu}(\mathbf{r}') \delta(\mathbf{r} - \mathbf{r}') G_{\gamma\delta}(\mathbf{r}') - \int \mathrm{d}^d\mathbf{r}'\, J_{\alpha\beta}(\mathbf{r}') \partial_\gamma \delta(\mathbf{r} - \mathbf{r}') v^{(\mathrm{d})}_{\gamma}(\mathbf{r}') \\ &\quad = F^{\alpha\beta\gamma\delta}_{\mu\nu} J_{\mu\nu}(\mathbf{r}) G_{\gamma\delta}(\mathbf{r}) - \partial_\gamma \left(J_{\alpha\beta}(\mathbf{r}) v^{(\mathrm{d})}_{\gamma}(\mathbf{r}) \right). \end{aligned} \tag{A.133}$$

Considering

$$\begin{aligned} F^{\alpha\beta\gamma\delta}_{\mu\nu} J_{\mu\nu} G_{\gamma\delta} &= (J_{\alpha\delta}\delta_{\beta\gamma} - J_{\gamma\beta}\delta_{\alpha\delta}) G_{\gamma\delta} = J_{\alpha\delta} G_{\beta\delta} - J_{\gamma\beta} G_{\gamma\alpha} \\ &= J_{\alpha\delta} G^{\mathsf{T}}_{\delta\beta} - G^{\mathsf{T}}_{\alpha\gamma} J_{\gamma\beta}, \end{aligned} \tag{A.134}$$

we find the equation of motion for $\mathbf{J}$:

$$\left(\frac{\partial \mathbf{J}}{\partial t}\right)_{\text{cons}} = -\boldsymbol{\nabla} \cdot \left(\mathbf{v}^{(\mathrm{d})} \mathbf{J}\right) + \mathbf{J}\mathbf{G}^{\mathsf{T}} - \mathbf{G}^{\mathsf{T}}\mathbf{J}. \tag{A.135}$$

From the results in section 4.1, one would expect that $\mathbf{K}$ and $\mathbf{J}$ do not couple to the dynamics of $\mathbf{j}^{(\mathrm{d})}$. Explicit evaluation results in

$$\begin{aligned} \left(\frac{\partial j^{(\mathrm{d})}_{\alpha}(\mathbf{r})}{\partial t}\right)_{\text{cons}} &= \int \mathrm{d}^d\mathbf{r}' \left[\left\{ j^{(\mathrm{d})}_{\alpha}(\mathbf{r}), \rho^{(\mathrm{d})}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}}{\delta \rho^{(\mathrm{d})}(\mathbf{r}')} + \left\{ j^{(\mathrm{d})}_{\alpha}(\mathbf{r}), j^{(\mathrm{d})}_{\beta}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}}{\delta j^{(\mathrm{d})}_{\beta}(\mathbf{r}')} \right. \\ &\quad \left. + \left\{ j^{(\mathrm{d})}_{\alpha}(\mathbf{r}), K_{\gamma\delta}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}}{\delta K_{\gamma\delta}(\mathbf{r}')} + \left\{ j^{(\mathrm{d})}_{\alpha}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}}{\delta J_{\gamma\delta}(\mathbf{r}')} \right]. \end{aligned} \tag{A.136}$$

The last line of the previous equation contains the only terms that could add contributions of $\mathbf{K}$ and $\mathbf{J}$ to the dynamics of $\mathbf{j}^{(\mathrm{d})}$. It can be evaluated using the helping relation eq. (4.32):

$$\begin{aligned}
&\int \mathrm{d}^d\mathbf{r}' \left[\left\{ j_\alpha^{(\mathrm{d})}(\mathbf{r}), K_{\gamma\delta}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}}{\delta K_{\gamma\delta}(\mathbf{r}')} + \left\{ j_\alpha^{(\mathrm{d})}(\mathbf{r}), J_{\gamma\delta}(\mathbf{r}') \right\} \frac{\delta \mathscr{H}}{\delta J_{\gamma\delta}(\mathbf{r}')} \right] \\
&\quad K_{\gamma\delta}(\mathbf{r}) \partial_\alpha \frac{\delta \mathscr{H}}{\delta K_{\gamma\delta}(\mathbf{r})} + J_{\gamma\delta}(\mathbf{r}) \partial_\alpha \frac{\delta \mathscr{H}}{\delta J_{\gamma\delta}(\mathbf{r})} \\
&\quad = \frac{\delta_{\gamma\delta}}{2} K_{\gamma\delta}(\mathbf{r}) \partial_\alpha \left[1 - \frac{m^{(\mathrm{r})}}{k} \operatorname{Tr}\Big(\mathbf{G}\mathbf{G}^{\mathsf{T}} \Big)(\mathbf{r}) \right] + J_{\gamma\delta}(\mathbf{r}) \partial_\alpha G_{\gamma\delta}(\mathbf{r}).
\end{aligned} \tag{A.137}$$

The argument $\mathbf{r}$ is omitted for clarity and the equation further simplified resulting in

$$\begin{aligned}
&- \frac{1}{2} \frac{m^{(\mathrm{r})}}{k} \operatorname{Tr} \mathbf{K} \partial_\alpha \Big[\operatorname{Tr}\Big(\mathbf{G}\mathbf{G}^{\mathsf{T}} \Big) \Big] + J_{\gamma\delta} \partial_\alpha G_{\gamma\delta} \\
&\quad = -\frac{1}{2} \frac{m^{(\mathrm{r})}}{k} \operatorname{Tr} \mathbf{K} \partial_\alpha \Big[\operatorname{Tr}\Big(\mathbf{G}\mathbf{G}^{\mathsf{T}} \Big) \Big] + \frac{m^{(\mathrm{r})}}{k} G_{\gamma\delta} \operatorname{Tr} \mathbf{K} \partial_\alpha G_{\gamma\delta} = 0.
\end{aligned} \tag{A.138}$$

As expected, the tensor fields $\mathbf{K}$ and $\mathbf{J}$ do not couple to the dynamics of $\mathbf{j}^{(\mathrm{d})}$.

In summary, the conservative equations of motion are:

$$\left(\frac{\partial \mathbf{K}}{\partial t} \right)_{\mathrm{cons}} = -\boldsymbol{\nabla} \cdot \Big(\mathbf{v}^{(\mathrm{d})} \mathbf{K} \Big) + \mathbf{K}\mathbf{G}^{\mathsf{T}} + \mathbf{G}\mathbf{K}, \tag{A.139}$$

$$\begin{aligned}
\left(\frac{\partial \mathbf{J}}{\partial t} \right)_{\mathrm{cons}} &= -\boldsymbol{\nabla} \cdot \Big(\mathbf{v}^{(\mathrm{d})} \mathbf{J} \Big) + \mathbf{J}\mathbf{G}^{\mathsf{T}} - \mathbf{G}^{\mathsf{T}}\mathbf{J} \\
&= -\boldsymbol{\nabla} \cdot \Big(\mathbf{v}^{(\mathrm{d})} \mathbf{J} \Big) + \frac{k}{m^{(\mathrm{r})}} \frac{\mathbf{J}\mathbf{J}^{\mathsf{T}} - \mathbf{J}^{\mathsf{T}}\mathbf{J}}{\operatorname{Tr} \mathbf{K}}.
\end{aligned} \tag{A.140}$$

This is, in essence, the flip-invariant formulation of a harmonic oscillator that is convected by the field $\mathbf{v}^{(\mathrm{d})}$. Choosing the tensor formulation has introduced additional degrees of freedom as opposed to using the vector-valued variables $\mathbf{q}$ and $\mathbf{v}^{(\mathrm{r})}$. Whether or not the symmetry and idempotency constraints eqs. (A.97), (A.98), and (A.106) can be used to reduce the total number of equations, or may even be essential for a consistent description, is at this point not fully resolved. This might be possible by using a Hamiltonian formalism with constraints in the spirit of references [109, 110]. Coupling to the solvent velocity field can be facilitated by constructing a suitable continuum dissipation rate similar to what has been done in section 4.1.4; however, further exploration is left for future work.

A.4 Lattice void model

There is an approach of modeling a compressible mixture in the framework of Flory-Huggins theory using so called voids [41–43, 111]. Additional to the particle species $i = 1, 2, \ldots,$ there are the voids $i = 0$ with a molecular weight of $M_0 = 1$ and no interaction of energetic nature, i.e. $u_{0k} = 0 \,\forall\, k$, which serve a special role. To allow for changes in overall volume, the total number of voids N_0 is considered to be variable. A change in volume then corresponds to a change in the number of voids while the number of 'real' particles remains constant for each component individually. As has been shown in the section about Flory-Huggins theory, section 2.2, the free energy per lattice site and $k_\mathrm{B}T$ then has the form

$$\frac{f_\mathrm{FH}(\{N_{k>0}\}, V, T)}{k_\mathrm{B}T} = \sum_i \frac{\phi_i}{M_i} \log \frac{\phi_i}{M_i} + \beta \sum_{ij} \phi_i \phi_j u_{ij}, \tag{A.141}$$

where the volume fraction of voids is given by the fraction of the system that is not occupied by real particles,

$$\phi_0 = 1 - \sum_{i>0} \phi_i. \tag{A.142}$$

From the free energy density the pressure is calculated via

$$P(\{N_{k>0}\}, V, T) = -\left(\frac{\partial F_\mathrm{FH}}{\partial V}\right)_{\{N_{k>0}\},T} = -\frac{1}{a^d}\left[f_\mathrm{FH} + V\left(\frac{\partial f_\mathrm{FH}}{\partial V}\right)_{\{N_{k>0}\},T}\right]. \tag{A.143}$$

Because of the definition of the volume fractions $\phi_i = N_i M_i a^d / V$, and eq. (A.142), the derivatives of the volume fractions are

$$\begin{aligned} V\left(\frac{\partial \phi_i}{\partial V}\right)_{\{N_{k>0}\}} &= -\phi_i \text{ for } i \neq 0, \\ V\left(\frac{\partial \phi_0}{\partial V}\right)_{\{N_{k>0}\}} &= -\sum_{i>0}\left(V\frac{\partial \phi_i}{\partial V}\right)_{\{N_{k>0}\}} = \sum_{i>0} \phi_i = 1 - \phi_0, \end{aligned} \tag{A.144}$$

which can be summarized using the Kronecker delta

$$V\left(\frac{\partial \phi_i}{\partial V}\right)_{\{N_{k>0}\}} = \delta_{i0} - \phi_i. \tag{A.145}$$

Then, with $M_0 = 1$, the derivative of the entropic part of the free energy is found to be

$$V\frac{\partial}{\partial V}\left(\sum_i \frac{\phi_i}{M_i} \log \frac{\phi_i}{M_i}\right)_{\{N_{k>0}\},T} = \log \phi_0 + 1 - \sum_i \frac{\phi_i}{M_i}\left(\log \frac{\phi_i}{M_i} + 1\right). \tag{A.146}$$

Because the voids do not interact energetically, the derivative of the energetic part of the free energy is just

$$V\frac{\partial}{\partial V}\left(\sum_{ij}\phi_i\phi_j u_{ij}\right)_{\{N_{k>0}\},T} = -2\sum_{ij}\phi_i\phi_j u_{ij}. \tag{A.147}$$

Then, the expression for the pressure is determined from eqs. (A.141) and (A.143) by using the derivatives of the free energy density for the entropic and energetic part eqs. (A.146) and (A.147):

$$\frac{a^d}{k_\mathrm{B}T}P(\{N_{k>0}\},V,T) = \sum_i \frac{\phi_i}{M_i} - \log\phi_0 - 1 + \beta\sum_{ij}\phi_i\phi_j u_{ij}. \tag{A.148}$$

In particular, if there is only one non-void component in the system, this becomes

$$\frac{a^d}{k_\mathrm{B}T}P(N_1,V,T) = \phi_0 + \frac{\phi_1}{M_1} - \log\phi_0 - 1 + \beta\phi_1^2 u_{11}. \tag{A.149}$$

In the limit of small volume fraction of the matter-like component, $\phi_1 \to 0$, the void volume fraction approaches one, $\phi_0 \to 1$, resp. $\log\phi_0 \to 0$, and the pressure is approximated by

$$\frac{a^d}{k_\mathrm{B}T}P(N_1,V,T) = \frac{\phi_1}{M_1} = \frac{a^d N_1}{V}, \tag{A.150}$$

which is just the ideal gas law.

For two non-void components the pressure has the form

$$\begin{aligned}\frac{a^d}{k_\mathrm{B}T}P(N_1,N_2,V,T) = \phi_0 + \frac{\phi_1}{M_1} + \frac{\phi_2}{M_2} - \log\phi_0 - 1 \\ + \beta\Big(\phi_1^2 u_{11} + 2\phi_1\phi_2 u_{12} + \phi_2^2 u_{22}\Big),\end{aligned} \tag{A.151}$$

which has a logarithmic singularity for $(\phi_1+\phi_2)\to 1$, accounting for a finite volume of the molecules. This is an improvement upon the equation of state eq. (2.37) obtained with the naive approach. However, the physical interpretation of this model is still somewhat problematic as the voids contribute entropically, and the model is still bound to a lattice interpretation. This is why the equation of state derived from Van der Waals theory in section 2.3 is preferred here.

A.5 Effects of confinement on the pressure tensor

In this section, the effects of the confining potential, which was introduced in section 3.1.6, on the pressure tensor are examined. First, an expression for the pressure tensor in a system with one and two-body interactions is derived, and then the special case where the one-body potential only depends on the z-coordinate of particles is analyzed.

Consider a distortion of the system which transforms the coordinates of the contained particles $\mathbf{r}_i$ like

$$\mathbf{r}_i \to \mathbf{r}_i' = \mathbf{A} \cdot \mathbf{r}_i. \tag{A.152}$$

If this distortion is small enough it can be expressed with the infinitesimal strain tensor $\mathrm{d}\boldsymbol{\epsilon}$ by

$$\mathbf{A} = \mathbb{1} + \mathrm{d}\boldsymbol{\epsilon}. \tag{A.153}$$

We define the Hamiltonian by

$$\mathscr{H}(\{\mathbf{r}_k\}, \{\mathbf{p}_k\}) = \sum_i \frac{\mathbf{p}_i^2}{2m_i} + U(\{\mathbf{r}_k\}), \tag{A.154}$$

where U is for now a very general potential depending on all of the particle's positions. Omitting constant prefactors, the canonical partition function in the undistorted system is

$$\mathscr{Z}_\mathrm{c} = \int \mathrm{d}^{dN}\mathbf{p} \int_V \mathrm{d}^{dN}\mathbf{r} \exp[-\beta\mathscr{H}(\{\mathbf{r}_k\}, \{\mathbf{p}_k\})]. \tag{A.155}$$

Here, $\beta = 1/(k_\mathrm{B}T)$, and the coordinates of the N particles are integrated over the volume of the original box. Accordingly, the partition function in the distorted system (apart from constant prefactors) is given by

$$\mathscr{Z}_\mathrm{c}' = \int \mathrm{d}^{dN}\mathbf{p}' \int_{V'} \mathrm{d}^{dN}\mathbf{r}' \exp[-\beta\mathscr{H}(\{\mathbf{r}_k'\}, \{\mathbf{p}_k'\})], \tag{A.156}$$

where coordinates are integrated over the volume of the distorted box.

A canonical transformation of coordinates and momenta is

$$\mathbf{r}_i' = (\mathbb{1} + \mathrm{d}\epsilon_{\alpha\beta})\mathbf{r}_i, \qquad \mathbf{p}_i' = (\mathbb{1} - \mathrm{d}\epsilon_{\alpha\beta})\mathbf{p}_i. \tag{A.157}$$

Discarding terms that are higher than linear order in the strain, the phase-space volume element of integration then simply transforms like

$$\mathrm{d}^{dN}\mathbf{r}\,\mathrm{d}^{dN}\mathbf{p} = \mathrm{d}^{dN}\mathbf{r}'\,\mathrm{d}^{dN}\mathbf{p}'\,. \tag{A.158}$$

Changing the integration variables and the integration volume accordingly, the partition function in the distorted system is then

$$\mathscr{Z}_{\mathrm{c}}{}' = \int \mathrm{d}^{dN}\mathbf{p} \int_V \mathrm{d}^{dN}\mathbf{r} \exp[-\beta\mathscr{H}(\{\mathbf{r}_k'\},\{\mathbf{p}_k'\})]. \tag{A.159}$$

The potential energy in the distorted system is expressed by

$$U(\{\mathbf{r}_k'\}) = U(\{\mathbf{r}_k\}) + \mathrm{d}U(\{\mathbf{r}_k\}). \tag{A.160}$$

Again neglecting strain of second order, the squared momentum in the distorted system becomes

$$\mathbf{p}_i'^2 = p_{i\alpha}p_{i\alpha} - 2\mathrm{d}\epsilon_{\alpha\beta}p_{i\alpha}p_{i\beta}, \tag{A.161}$$

Combining the above, the transformed Hamiltonian can then be written as

$$\mathscr{H}(\{\mathbf{r}_k'\},\{\mathbf{p}_k'\}) = \mathscr{H}(\{\mathbf{r}_k\},\{\mathbf{p}_k\}) - \mathrm{d}\epsilon_{\alpha\beta}\sum_i \frac{p_{i\alpha}p_{i\beta}}{m_j} + \mathrm{d}U(\{\mathbf{r}_k\}). \tag{A.162}$$

With the phase-space average defined by

$$\langle f(\{\mathbf{r}_k\},\{\mathbf{p}_k\})\rangle = \frac{\int \mathrm{d}^{dN}\mathbf{p}\int_V \mathrm{d}^{dN}\mathbf{r}\exp[-\beta\mathscr{H}(\{\mathbf{r}_k\},\{\mathbf{p}_i\})]f(\{\mathbf{r}_k\},\{\mathbf{p}_k\})}{\int \mathrm{d}^{dN}\mathbf{p}\int_V \mathrm{d}^{dN}\mathbf{r}\exp[-\beta\mathscr{H}(\{\mathbf{r}_k\},\{\mathbf{p}_k\})]}, \tag{A.163}$$

the ratio between the partition functions of the distorted and undistorted system is

$$\frac{\mathscr{Z}_{\mathrm{c}}{}'}{\mathscr{Z}_{\mathrm{c}}} = \left\langle \exp\left(\beta\mathrm{d}\epsilon_{\alpha\beta}\sum_i \frac{p_{i\alpha}p_{i\beta}}{m_i} - \beta\,\mathrm{d}U\right)\right\rangle. \tag{A.164}$$

From this, the change in Helmholtz free energy can then be calculated via

$$\begin{aligned}\mathrm{d}F &= -k_{\mathrm{B}}T\log\frac{\mathscr{Z}_{\mathrm{c}}{}'}{\mathscr{Z}_{\mathrm{c}}} = -\mathrm{d}\epsilon_{\alpha\beta}\sum_i \frac{1}{m_i}\langle p_{i\alpha}p_{i\beta}\rangle + \langle \mathrm{d}U\rangle \\ &= -P_{\alpha\beta}V\mathrm{d}\epsilon_{\alpha\beta},\end{aligned} \tag{A.165}$$

where $P_{\alpha\beta}$ are the components of the pressure tensor. With the conformational contribution to the pressure tensor $\mathbf{P}^{(\mathrm{conf})}$ defined by

$$\langle \mathrm{d}U\rangle = -P_{\alpha\beta}^{(\mathrm{conf})}V\mathrm{d}\epsilon_{\alpha\beta}, \tag{A.166}$$

we get the expression for the pressure tensor

$$P_{\alpha\beta} = \frac{1}{V}\sum_i \frac{\langle p_{i\alpha}p_{i\beta}\rangle}{m_i} + P_{\alpha\beta}^{(\mathrm{conf})}. \tag{A.167}$$

Equation (A.167) holds true for any general potential depending only on the particle positions. Treated below is the case where U can be decomposed into single-particle and pair interaction:

$$U = \sum_i U^{(1)}(\mathbf{r}_i) + \sum_{i>j} U^{(2)}(\mathbf{r}_{ij}). \tag{A.168}$$

To ensure translational invariance in a system with periodic boundary conditions, $\mathbf{r}_{ij}$ is taken to be the minimum-image connecting vector between two particles with $\mathbf{d}\mathbf{r}_{ij} = \mathbf{r}_{ij}\,\mathbf{d}\boldsymbol{\epsilon}$. Then, introducing the n-body forces $\mathbf{F}^{(n)} = -\boldsymbol{\nabla} U^{(n)}$, the change in potential energy becomes

$$\begin{aligned} \mathrm{d}U &= \sum_i \frac{\partial U^{(1)}(\mathbf{r}_i)}{\partial \mathbf{r}_i} \cdot \mathbf{d}\mathbf{r}_i + \sum_{i>j} \frac{\partial U^{(2)}(r_{ij})}{\partial \mathbf{r}_{ij}} \cdot \mathbf{d}\mathbf{r}_{ij} \\ &= -\,\mathrm{d}\epsilon_{\alpha\beta} \sum_i F_\alpha^{(1)}(\mathbf{r}_i) r_{i\beta} + \sum_{i>j} F_\alpha^{(2)}(\mathbf{r}_{ij}) r_{ij\beta}. \end{aligned} \tag{A.169}$$

The configurational pressure tensor can likewise be decomposed into contributions from one- and two-body interactions:

$$P_{\alpha\beta}^{(\mathrm{conf},1)} = \frac{1}{V} \sum_i \left\langle F_\alpha^{(1)}(\mathbf{r}_i) r_{i\beta} \right\rangle, \tag{A.170}$$

$$P_{\alpha\beta}^{(\mathrm{conf},2)} = \frac{1}{V} \sum_{i>j} \left\langle F_\alpha^{(2)}(\mathbf{r}_{ij}) r_{ij\beta} \right\rangle. \tag{A.171}$$

Here, we are particularly interested in the effects of the confining potential introduced in section 3.1.6 (see eq. (3.27)), which results in a force that only depends on the z-coordinates of the particles and acts exclusively in the z-direction,

$$\mathbf{F}^{(1)}(\mathbf{r}_i) = F_z^{(1)}(z_i)\hat{\mathbf{e}}_z. \tag{A.172}$$

Hence, the contribution of the confining potential is given by

$$P_{\alpha\beta}^{(\mathrm{conf},1)} = \frac{1}{V} \sum_i \left\langle F_\alpha^{(1)}(\mathbf{r}_i) r_{i\beta} \right\rangle = \frac{1}{V} \sum_i \left\langle F_z^{(1)}(\mathbf{r}_i) r_{i\beta} \right\rangle \delta_{z\alpha}, \tag{A.173}$$

or in full matrix notation

$$\mathbf{P}^{(\mathrm{conf},1)} = \frac{1}{V} \sum_i \left\langle \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ x_i F_z^{(1)}(z_i) & y_i F_z^{(1)}(z_i) & z_i F_z^{(1)}(z_i) \end{pmatrix} \right\rangle. \tag{A.174}$$

For symmetry reasons, the pressure must be invariant under the arbitrary application of the transformations $x \to -x$ and $y \to -y$. Therefore, the zx and zy components of the pressure tensor need to vanish as well and

$$P_{\alpha\beta}^{(\mathrm{conf},1)} = \frac{1}{V} \sum_i \left\langle F_z^{(1)}(z_i) z_i \right\rangle \delta_{\alpha z} \delta_{\beta z}. \tag{A.175}$$

In particular, the xx, xy, yx and yy components of the pressure tensor are not directly affected by the external potential.

A.5.1 Virial expansion for the pressure tensor

In this section, an expansion of the pressure tensor in the density is derived starting from the grand canonical partition function

$$\mathscr{Z}_{\mathrm{gc}}(\mu, V, T) = \sum_{N=0}^{\infty} \mathscr{Z}_{\mathrm{c}}(N, V, T) \exp(\beta N \mu), \tag{A.176}$$

where μ is the chemical potential. Introducing the shorthand for the N-particle canonical partition function $\mathscr{Z}_N = \mathscr{Z}_{\mathrm{c}}(N, V, T)$, and the fugacity $z = \exp(\beta\mu)$, this becomes

$$\mathscr{Z}_{\mathrm{gc}}(\mu, V, T) = \sum_{N=0}^{\infty} \mathscr{Z}_N z^N. \tag{A.177}$$

The grand potential is then defined by

$$-\beta\Omega = \log(\mathscr{Z}_{\mathrm{c}}) = \log\Big(1 + \mathscr{Z}_1 z + \mathscr{Z}_2 z^2 + \mathscr{O}\left(z^3\right)\Big). \tag{A.178}$$

Assuming small fugacity, z the logarithm can be expanded*, and eq. (A.178) can be approximated by

$$\begin{aligned} -\beta\Omega &= \mathscr{Z}_1 z + \mathscr{Z}_2 z^2 - \frac{1}{2}\Big(\mathscr{Z}_1 z + \mathscr{Z}_2 z^2\Big)^2 + \mathscr{O}\left(z^3\right) \\ &= \mathscr{Z}_1 z + \mathscr{Z}_2 z^2 - \frac{1}{2}\mathscr{Z}_1^2 z^2 + \frac{1}{3}\mathscr{Z}_1^3 z^3 + \mathscr{O}\left(z^3\right) \\ &= \mathscr{Z}_1 z + \left(\mathscr{Z}_2 - \frac{1}{2}\mathscr{Z}_1^2\right) z^2 + \mathscr{O}\left(z^3\right). \end{aligned} \tag{A.179}$$

With the small deformation described by eq. (A.157), the change in the (isothermal, $\mathrm{d}T = 0$) grand potential is

$$\mathrm{d}\Omega = -V P_{\alpha\beta} \mathrm{d}\epsilon_{\alpha\beta} - N\,\mathrm{d}\mu\,. \tag{A.180}$$

*$\log(1+x) = x - x^2/2 + x^3/3 + \mathscr{O}\left(x^4\right)$ for small x

Because of

$$\mathrm{d}z = \beta \exp(\beta\mu)\mathrm{d}\mu = \beta z \mathrm{d}\mu, \tag{A.181}$$

this can also be written like

$$-\beta\,\mathrm{d}\Omega = \beta V P_{\alpha\beta}\mathrm{d}\epsilon_{\alpha\beta} + \frac{N}{z}\mathrm{d}z. \tag{A.182}$$

Taking the differential of the expansion eq. (A.179), on the other hand, results in the expression

$$\begin{aligned} -\beta\mathrm{d}\Omega &= -\beta\left(\frac{\partial\Omega}{\partial\epsilon_{\alpha\beta}}\mathrm{d}\epsilon_{\alpha\beta} + \frac{\partial\Omega}{\partial z}\mathrm{d}z\right) \\ &= \frac{\partial}{\partial\epsilon_{\alpha\beta}}\left[\mathscr{Z}_1 z + \left(\mathscr{Z}_2 - \frac{1}{2}\mathscr{Z}_1^2\right)z^2 + \mathscr{O}\left(z^3\right)\right]\mathrm{d}\epsilon_{\alpha\beta} \\ &\quad + \left[\mathscr{Z}_1 + \left(2\mathscr{Z}_2 - \mathscr{Z}_1^2\right)z\right]\mathrm{d}z + \mathscr{O}\left(z^2\mathrm{d}z\right). \end{aligned} \tag{A.183}$$

Comparing the coefficients of eqs. (A.182) and (A.183) yields an expression for the particle number

$$\frac{N}{z} = \mathscr{Z}_1 + \left(2\mathscr{Z}_2 - \mathscr{Z}_1^2\right)z + \mathscr{O}\left(z^2\right), \tag{A.184}$$

as well as the equation of state

$$\begin{aligned} \beta V P_{\alpha\beta} &= \frac{\partial}{\partial\epsilon_{\alpha\beta}}\left[\mathscr{Z}_1 z + \left(\mathscr{Z}_2 - \frac{1}{2}\mathscr{Z}_1^2\right)z^2\right] + \mathscr{O}\left(z^3\right) \\ &= z\frac{\partial\mathscr{Z}_1}{\partial\epsilon_{\alpha\beta}} + \left(\frac{\partial\mathscr{Z}_2}{\partial\epsilon_{\alpha\beta}} - \mathscr{Z}_1\frac{\partial\mathscr{Z}_1}{\partial\epsilon_{\alpha\beta}}\right)z^2 + \mathscr{O}\left(z^3\right), \end{aligned} \tag{A.185}$$

which however still depends on the fugacity z.

Expression for the fugacity

An expression for the fugacity in terms of the partition functions $\mathscr{Z}_N$, and particle number N is derived below. This is then used to complete the equation of state.

First, an alternative expression for the particle number is obtained from the grand canonical partition function by

$$\begin{aligned} N &= -\frac{\partial\Omega}{\partial\mu} = -\frac{\partial\Omega}{\partial z}\frac{\partial z}{\partial\mu} = -\beta z\frac{\partial\Omega}{\partial z} = z\frac{\partial\log(\mathscr{Z}_{\mathrm{gc}})}{\partial z} = \frac{z}{\mathscr{Z}_{\mathrm{gc}}}\frac{\partial\mathscr{Z}_{\mathrm{gc}}}{\partial z} \\ &= \frac{1}{\mathscr{Z}_{\mathrm{gc}}}\sum_{k=0}^{\infty} k\mathscr{Z}_k z^k, \end{aligned} \tag{A.186}$$

which is equivalent to

$$N\mathscr{Z}_{\mathrm{gc}} = \sum_{k=0}^{\infty} k\mathscr{Z}_k z^k. \tag{A.187}$$

Expanding the fugacity in terms of the particle number with coefficients a_k,

$$z = \sum_{k=1}^{\infty} a_k N^k, \tag{A.188}$$

and inserting in the left hand side of eq. (A.187), results in

$$\begin{aligned} N\mathscr{Z}_{\mathrm{gc}} &= N\Big[1 + \mathscr{Z}_1\Big(a_1 N + a_2 N^2\Big) + \mathscr{Z}_2\Big(a_1 N + a_2 N^2\Big)^2 + \dots\Big] \\ &= N + a_1\mathscr{Z}_1 N^2 + \Big(a_2\mathscr{Z}_1 + a_1^2\mathscr{Z}_2^2\Big)N^3 + \dots \end{aligned} \tag{A.189}$$

Inserting the fugacity expansion eq. (A.188) in the left hand side of eq. (A.187) yields

$$\begin{aligned} \sum_{k=0}^{\infty} k\mathscr{Z}_k z^k &= \mathscr{Z}_1\Big(a_1 N + a_2 N^2\Big) + 2\mathscr{Z}_2\Big(a_1 N + a_2 N^2\Big)^2 + \dots \\ &= a_1\mathscr{Z}_1 N + \Big(a_2\mathscr{Z}_1 + 2a_1^2\mathscr{Z}_2\Big)N^2 + 4a_1 a_2\mathscr{Z}_2 N^3 + \dots \end{aligned} \tag{A.190}$$

Comparing the coefficients corresponding to the various powers of N in both eqs. (A.189) and (A.190) allows to determine the coefficients

$$a_1 = \frac{1}{\mathscr{Z}_1}, \qquad\qquad a_2 = \frac{1}{\mathscr{Z}_1^2}\Big(\mathscr{Z}_1 - 2\frac{\mathscr{Z}_2}{\mathscr{Z}_1}\Big). \tag{A.191}$$

Hence, the fugacity can be expressed by

$$z = \frac{N}{\mathscr{Z}_1} + \frac{N^2}{\mathscr{Z}_1^2}\Big(\mathscr{Z}_1 - 2\frac{\mathscr{Z}_2}{\mathscr{Z}_1}\Big) + \mathscr{O}\Big(\frac{N^3}{\mathscr{Z}_1^3}\Big). \tag{A.192}$$

By inserting in the equation of state eq. (A.185) and ordering by powers of N, we get

$$\begin{aligned} \beta V P_{\alpha\beta} &= \frac{\partial\mathscr{Z}_1}{\partial\epsilon_{\alpha\beta}}\left[\frac{N}{\mathscr{Z}_1} + \frac{N^2}{\mathscr{Z}_1^2}\Big(\mathscr{Z}_1 - 2\frac{\mathscr{Z}_2}{\mathscr{Z}_1}\Big)\right] \\ &\quad + \left(\frac{\partial\mathscr{Z}_2}{\partial\epsilon_{\alpha\beta}} - \mathscr{Z}_1\frac{\partial\mathscr{Z}_1}{\partial\epsilon_{\alpha\beta}}\right)\frac{N^2}{\mathscr{Z}_1^2} + \mathscr{O}\Big(\frac{N^3}{\mathscr{Z}_1^3}\Big) \\ &= \frac{N}{\mathscr{Z}_1}\frac{\partial\mathscr{Z}_1}{\partial\epsilon_{\alpha\beta}} + \frac{N^2}{\mathscr{Z}_1^2}\left(\frac{\partial\mathscr{Z}_2}{\partial\epsilon_{\alpha\beta}} - 2\frac{\mathscr{Z}_2}{\mathscr{Z}_1}\frac{\partial\mathscr{Z}_1}{\partial\epsilon_{\alpha\beta}}\right) + \mathscr{O}\Big(\frac{N^3}{\mathscr{Z}_1^3}\Big). \end{aligned} \tag{A.193}$$

This defines the virial coefficients

$$B^{(1)}_{\alpha\beta} = \frac{1}{\mathscr{Z}_1}\frac{\partial \mathscr{Z}_1}{\partial \epsilon_{\alpha\beta}}, \tag{A.194}$$

$$B^{(2)}_{\alpha\beta} = \frac{1}{\mathscr{Z}_1^2}\left(\frac{\partial \mathscr{Z}_2}{\partial \epsilon_{\alpha\beta}} - 2\frac{\mathscr{Z}_2}{\mathscr{Z}_1}\frac{\partial \mathscr{Z}_1}{\partial \epsilon_{\alpha\beta}}\right) = \frac{\mathscr{Z}_2}{\mathscr{Z}_1^2}\left(\frac{1}{\mathscr{Z}_2}\frac{\partial \mathscr{Z}_2}{\partial \epsilon_{\alpha\beta}} - 2B^{(1)}_{\alpha\beta}\right), \tag{A.195}$$

which depend on the partition functions and therefore on the exact nature of particle interactions. The expansion in N given by eq. (A.193) can be transformed into an expansion in the mass density ρ by the relation

$$N = \frac{\rho V}{m}, \tag{A.196}$$

if all particles have the same mass m. Then, the virial expansion of the pressure has the general form

$$\beta V P_{\alpha\beta} = \sum_k \rho^k \left(\frac{V}{m}\right)^k B^{(k)}_{\alpha\beta}. \tag{A.197}$$

Derivatives of the partition function

To find the derivatives of the canonical partition functions $\mathscr{Z}_N$, note that the derivatives can be expressed by

$$\frac{\partial \mathscr{Z}_N}{\partial \epsilon_{\alpha\beta}} = \frac{\mathscr{Z}'_N}{\mathrm{d}\epsilon_{\alpha\beta}}. \tag{A.198}$$

For the distorted system, it has been shown at the beginning of the section that the partition function is

$$\begin{aligned}
\mathscr{Z}'_N &= \int \mathrm{d}^{dN}\mathbf{p}' \int_{V'} \mathrm{d}^{dN}\mathbf{r}' \exp[-\beta\mathscr{H}'] \\
&= \int \mathrm{d}^{dN}\mathbf{p} \int_{V} \mathrm{d}^{dN}\mathbf{r} \exp\left[-\beta\left(\mathscr{H} - \mathrm{d}\epsilon_{\alpha\beta}\sum_i \frac{p_{i\alpha}p_{i\beta}}{m_i} + \mathrm{d}U(\{\mathbf{r}_k\})\right)\right] \\
&= \beta\mathscr{Z}_N \left\langle \mathrm{d}\epsilon_{\alpha\beta}\sum_i \frac{p_{i\alpha}p_{i\beta}}{m_i} - \mathrm{d}U(\{\mathbf{r}_k\}) \right\rangle,
\end{aligned} \tag{A.199}$$

where the shorthand notations $\mathscr{H}' = \mathscr{H}(\{\mathbf{r}'_k\},\{\mathbf{p}'_k\})$ and $\mathscr{H} = \mathscr{H}(\{\mathbf{r}_k\},\{\mathbf{p}_k\})$ were introduced. With only one and two-body interactions,

$$\mathrm{d}U = -\,\mathrm{d}\epsilon_{\alpha\beta}\sum_i F^{(1)}_\alpha(\mathbf{r}_i) r_{i\beta} + \sum_{i>j} F^{(2)}_\alpha(\mathbf{r}_{ij}) r_{ij\beta}, \tag{A.200}$$

the derivatives become

$$\begin{aligned}\frac{1}{\mathscr{Z}_N}\frac{\partial \mathscr{Z}_N}{\partial \epsilon_{\alpha\beta}} &= \beta\left[\left\langle \sum_j \frac{p_{i\alpha}p_{i\beta}}{m_i}\right\rangle + \left\langle \sum_i F_\alpha^{(1)}(\mathbf{r}_i)r_{i\beta} + \sum_{i>j} F_\alpha^{(2)}(\mathbf{r}_{ij})r_{ij\beta}\right\rangle\right] \\ &= N\delta_{\alpha\beta} + \beta\left\langle \sum_i F_\alpha^{(1)}(\mathbf{r}_i)r_{i\beta} + \sum_{i>j} F_\alpha^{(2)}(\mathbf{r}_{ij})r_{ij\beta}\right\rangle \\ &= N\delta_{\alpha\beta} + N\beta\left\langle F_\alpha^{(1)}(\mathbf{r})r_\beta\right\rangle + \frac{N(N-1)}{2}\beta\left\langle F_\alpha^{(2)}(\mathbf{r}_{12})r_{12\beta}\right\rangle,\end{aligned} \tag{A.201}$$

where the equipartition theorem was used to express the average over the squared momentum.

In particular, the two body interactions do not contribute to the first virial coefficient because only the $N = 1$ partition function enters:

$$B_{\alpha\beta}^{(1)} = \frac{1}{\mathscr{Z}_1}\frac{\partial \mathscr{Z}_1}{\partial \epsilon_{\alpha\beta}} = \delta_{\alpha\beta} + \beta\left\langle F_\alpha^{(1)}(\mathbf{r})r_{i\beta}\right\rangle. \tag{A.202}$$

Hence, the first virial coefficient is proportional to the unit tensor apart from contributions from one-body forces. It is the same contribution that also enters in eq. (A.170) and for the same symmetry arguments it must only affect the zz-element for the confining potential introduced in section 3.1.6.

The full N-particle canonical partition function including prefactors is given by

$$\mathscr{Z}_N = \frac{1}{N!h^{dN}}\int \mathrm{d}^{dN}\mathbf{p}\int_V \mathrm{d}^{dN}\mathbf{r}\exp\left[-\beta\left(\sum_i \frac{\mathbf{p}_i^2}{2m_i} + U(\{\mathbf{r}_k\})\right)\right], \tag{A.203}$$

where h is Planck's constant and the factorial $N!$ accounts for the fact that the particles are indistinguishable. For simplicity, we assume that all particles have the same mass $m_i = m$. The integration with respect to the momenta can then be carried out resulting in

$$\mathscr{Z}_N = \frac{1}{N!\Lambda^{dN}}\int_V \mathrm{d}^{dN}\mathbf{r}\exp[-\beta U(\{\mathbf{r}_k\})] = \frac{\mathscr{C}_N}{N!\Lambda^{dN}}, \tag{A.204}$$

where $\Lambda = \sqrt{\beta h^2/(2\pi m)}$ is the thermal De Broglie wavelength and the N-particle configurational integral was abbreviated by $\mathscr{C}_N$. The one-body interaction then cancels from the second virial coefficient

$$\begin{aligned}B_{\alpha\beta}^{(2)} &= \frac{\mathscr{Z}_2}{\mathscr{Z}_1^2}\left(\frac{1}{\mathscr{Z}_2}\frac{\partial \mathscr{Z}_2}{\partial \epsilon_{\alpha\beta}} - 2B_{\alpha\beta}^{(1)}\right) \\ &= 2\frac{\mathscr{C}_2}{\mathscr{C}_1^2}\left(2\delta_{\alpha\beta} + 2\beta\left\langle F_\alpha^{(1)}(\mathbf{r})r_\beta\right\rangle + \beta\left\langle F_\alpha^{(2)}(\mathbf{r}_{12})r_{12\beta}\right\rangle - 2B_{\alpha\beta}^{(1)}\right) \\ &= 2\beta\frac{\mathscr{C}_2}{\mathscr{C}_1^2}\left\langle F_\alpha^{(2)}(\mathbf{r}_{12})r_{12\beta}\right\rangle.\end{aligned} \tag{A.205}$$

Explicit calculation of virial coefficients

If the average is over a function only depending on positions the integration over momentum space cancels out and eq. (A.163) simplifies to

$$\begin{aligned}\langle f(\{\mathbf{r}_k\})\rangle &= \frac{\int_V \mathrm{d}^{dN}\mathbf{r}\exp[-\beta U(\{\mathbf{r}_k\})]f(\{\mathbf{r}_k\})}{\int_V \mathrm{d}^{dN}\mathbf{r}\exp[-\beta U(\{\mathbf{r}_k\})]} \\ &= \frac{1}{\mathscr{C}_N}\int_V \mathrm{d}^{dN}\mathbf{r}\exp[-\beta U(\{\mathbf{r}_k\})]f(\{\mathbf{r}_k\}).\end{aligned} \tag{A.206}$$

The force term entering the first virial coefficient is then

$$\left\langle F_\alpha^{(1)}(\mathbf{r})r_{i\beta}\right\rangle = \frac{1}{\mathscr{C}_1}\int \mathrm{d}^d\mathbf{r}\exp\left[-\beta U^{(1)}(\mathbf{r})\right]F_\alpha^{(1)}(\mathbf{r})r_\beta. \tag{A.207}$$

The second virial coefficient becomes

$$B_{\alpha\beta}^{(2)} = \frac{2\beta}{\mathscr{C}_1^2}\int_V \mathrm{d}^d\mathbf{r}_1\,\mathrm{d}^d\mathbf{r}_2\exp[-\beta U(\mathbf{r}_1,\mathbf{r}_2)]F^{(2)}(\mathbf{r}_{12})|r_{12\beta}|, \tag{A.208}$$

where

$$U(\mathbf{r}_1,\mathbf{r}_2) = U^{(1)}(\mathbf{r}_1) + U^{(1)}(\mathbf{r}_2) + U^{(2)}(\mathbf{r}_1,\mathbf{r}_2). \tag{A.209}$$

Appendix B

Software documentation

In this chapter, the documentation for some of the software that was developed during this project is presented.

Appendix B.1 contains the documentation for the Python script for the determination of Lattice Boltzmann weights that was introduced in section 4.2. Model parameters are taken from user input, and the resulting system is checked for solubility. If a unique solution exists, it is computed; if there are infinitely many solutions, a particular solution can be determined by further processing with a secondary script.

In appendix B.2, the documentation for a hierarchy of libraries developed for the analysis of Molecular Dynamics trajectories is shown. The implementation features a trajectory reader class that is able to efficiently iterate configurations that are concatenated in large `.xyz` files. The information for each timestep can be read into a designated data structure called 'Frame', which also contains methods for manipulating and analyzing configurations of linear chain molecules.

B.1 LBWeights

In section 4.2, a procedure for the determination of Lattice Boltzmann weights is introduced. This procedure was implemented in the Python script `LBWeights.py`. Presented below is the documentation of this script that was automatically generated from the docstrings in the code with the documentation tool `sphinx` [112].

LBweights Documentation

Burkhard Dünweg, Dominic Spiller

Jun 22, 2021

CONTENTS:

- genindex
- modindex
- search

CHAPTER

ONE

LATTICE-BOLTZMANN-WEIGHTS

"LBweights.py" is a Python script that calculates the weights of the (user-supplied) shells of a Lattice Boltzmann model on a simple cubic lattice, based upon numerically solving the Maxwell-Boltzmann constraint (MBC) equations. The script supports arbitrary spacial dimensions and arbitrary tensor ranks up to which the MBCs must hold. The script requires a Python installation (version 3.5 or 2.7) as well as NumPy. It assumes that the speed of sound is a free parameter and hence needs more shells than a model whose speed of sound takes on a definite value that is required for consistency. The output is typically given in the form: weights as a function of sound speed. There are cases where the supplied set of velocities does not admit any solution; in this case the script aborts. There are also cases where it admits infinitely many solutions; in this case an additional script "Continue.py" is used, which builds upon data that the main script stores on file.

In case of a unique solution, the script also calculates the interval(s) of sound speed for which all the weights are positive. At the borders of these intervals, at least one of the weights is zero, such that the corresponding shell may be discarded and one obtains a "reduced model". In this way, the script is able to reproduce well-known models like D2Q9, D3Q19, D3Q15, etc., but can also easily find higher-order models with significantly more speeds.

For Continue.py, the user has to supply a well-defined value of the sound speed (or an interval plus step size for scanning several values). Moreover, it requires the specification of a shell (or of a set of shells) whose weight (or sum of weights) is to be minimized. Continue.py then finds an optimal solution to the thus-specified linear programming problem. Continue.py therefore requires the package cvxpy, see http://www.cvxpy.org/ .

A significant part of the code is not in the main scripts but rather in a collection of functions in "Functions.py", which must be available to "LBweights.py" and "Continue.py".

Tedious tasks like the construction of velocity shells from the velocity modulus are done by the script.

Apart from being useful for researchers and practitioners, the script may perhaps also be used in a classroom setting.

A detailed description of the underlying mathematical theory, together with illustrative examples, is given in the paper "Semi-automatic construction of Lattice Boltzmann models" by Dominic Spiller and Burkhard Duenweg, see http://arxiv.org/abs/2004.03509 (original at Physical Review E, https://journals.aps.org/pre/abstract/10.1103/PhysRevE.101.043310 / open access).

More extensive documentation can be found at https://bduenweg.github.io/Lattice-Boltzmann-weights/ .

1.1 Installation

```
$ git clone https://github.com/BDuenweg/Lattice-Boltzmann-weights.git
$ virtualenv venv
$ source venv/bin/activate
$ cd Lattice-Boltzmann-weights
$ pip install -r requirements.txt
```

CHAPTER

TWO

LBWEIGHTS.PY

Calculation of the weights of an LB model. You can either supply the input data interactively or by the following command line arguments:

2.1 Usage

```
LBweights.py [-h] [-d D] [-m M] [-c C [C ...]] [-s S]
                    [-y] [--test] [--quiet] [--write-latex]

optional arguments:
  -h, --help     show this help message and exit
  -d D           spacial dimension of the lattice
  -m M           Maximum tensor rank
  -c C [C ...]   Space separated list of the radii c_i^2 of
                 the desired velocity shells
  -s S           Random number generator seed
  -y             Answer all prompts with yes (may overwrite
                 file data.npz)
  --test         Test, whether a set of weights that can be
                 written as a linear parametric equation
                 w = w_0 + lambda_1 w_1 + lambda_2 w_2
                 solves the equation A.w == b for given
                 speed of sound.
                 Weights and speed of sound are entered
                 interactively by the user.
  --quiet        Turn off most of the output
  --write-latex  Write unique solution to the file
                 "latex_tables.dat" in form of a latex
                 table. This will append to any existing
                 file.
```

Calculate LB model vectors and weights for a simple cubic lattice of arbitrary dimension

The method is described in D. Spiller's and B. Duenweg's paper "Semi-automatic construction of Lattice Boltzmann models" Therefore explanations in the code are not very detailed

Exit codes:

- 0: System has unique solution
- 1: System has no solution
- 2: System is underdetermined and requires further examination

- 3: System has unique solution but there is no physically valid range of existence
- 127: General error

LBweights.**Analysis**(*SpacialDimension*, *MaxTensorRank*, *ListOfTensorDimensions*, *GrandTotalList*, *Arguments*)

Performs the analysis for a given set of parameters

Parameters

- **SpacialDimension** (*int*) – Spacial dimension
- **MaxTensorRank** (*int*) – Maximum tensor rank M
- **ListOfTensorDimensions** (*list*) – List of the dimensions of tensor space for tensors of rank $2, 4, \ldots, M$.
- **GrandTotalList** (*list*) – List of lists. The s-th sublist contains all velocity vectors of shell s.
- **Arguments** (*dict*) – Dictionary of arguments as returned by function ParseArguments()

Returns

Return codes:

- 0: System has unique solution
- 1: System has no solution
- 2: System is underdetermined and requires further examination
- 3: System has unique solution but there is no physically valid range of existence
- 127: General error

Return type int

LBweights.**GetInputData**(*Arguments=None*, *ListOfThrowawayStrings=None*)

Parse command line arguments. You can optionally give a list with the subshells that you want to discard.

Parameters

- **Arguments** (*dict*) – Dictionary of command line arguments. This is useful, if the function is used in an automated script that does not rely on user input.
- **ListOfThrowawayStrings** (*list*) – List of indices of the subshells to be discarded. This is useful, if the function is used in an automated script that does not rely on user input.

Returns Tuple `(SpacialDimension, MaxTensorRank, ListOfTensorDimensions, GrandTotalList, Arguments)`

Return type tuple

CHAPTER

THREE

CONTINUE.PY

Find optimal weights for an underdetermined problem. This requires the file `data.npz` to be present in the directory that can be written by `LBweights.py` if an underdetermined problem is encountered. You can either supply the input data interactively or by the following command line arguments:

3.1 Usage

```
Continue.py [-h] [-c C [C ...]] [-m M [M ...]]


optional arguments:
  -h, --help    show this help message and exit
  -c C [C ...]  Range/value of c_s^2 to consider, either in
                the form <min> <max> <incr> or a single
                value.
  -m M [M ...]  List of indices of the weights that are to
                be minimized. You can use -1 to refer to the
                last shell etc.
```

Contains routines to treat the case of infinitely many solutions.

Exit codes:

- 0: No optimal solution found
- 1: Optimal solution found
- 127: General error

`Continue.`**`ParseArguments`**`()`
Function to parse command line options.

Returns Dictionary of command line options

Return type dict

`Continue.`**`Solve`**`(`*V*, *ReducedRhs*, *NumberOfRows*, *ShellSizes*, *CsSquared*, *MinimizeWeights*`)`
Solve the minimization problem via convex optimization. See: https://www.cvxpy.org/

Parameters

- **V** (*numpy.ndarray*) – Orthogonal matrix that results from the singular value decomposition A=U.S.V
- **ReducedRhs** (*numpy.ndarray*) – Pruned matrix that has the inverse singular values on the diagonal.

- **NumberOfRows** (*int*) – Number of rows of A
- **ShellSizes** (*list*) – List of shell sizes (int) NOT including zero shell
- **CsSquared** (*float*) – Speed of sound squared
- **MinimizeWeights** (*list*) – List of indices of the weights that shall be minimized in the procedure

Returns

cvxpy problem. Problem.status indicates whether or not the problem could be solved.

Return type cvxpy.problems.problem.Problem

CHAPTER

FOUR

FUNCTIONS.PY

Collection of helper functions for LBweights.py and Continue.py

Functions.**LINEWIDTH**
Line width for console output.

Type int

Functions.**QUIET**
Flag to suppress standard output.

Functions.**AbsSquared**(*Vector*)
Return the squared absolute value of numpy array.

Parameters **Vector** (*numpy.ndarray*) – Vector that is supposed to be squared

Returns Return the squared absolute value of Vector

Functions.**AnalyzeTensorDimension**(*CurrentTensorRank*)
Recursive generation of lists that specify what types of tensors of rank CurrentTensorRank are compatible with cubic invariance and also fully symmetric under index exchange. For rank 2, these are just multiples of the 2nd rank unit tensor δ_{ij}. Thus tensor dimension is one. For rank 4, these are multiples of δ_{ijkl} and multiples of (δ_{ij}, δ_{kl} + perm.). Thus tensor dimension is two. For rank 6, we get another tensor δ_{ijklmn}, but also all possible products of the lower-rank deltas. Hence tensor dimension is three. For each new (even) rank M we get another delta with M indexes, plus all possible products of the lower-order delta tensors So, for rank two we get `[[2]]` (1d) for rank four `[[4], [2,2]]` (2d) for rank six `[[6], [4,2], [2,2,2]]` (3d) for rank eight `[[8], [6, 2], [4,4], [4,2,2], [2,2,2,2]]` (5d) and so on. The routine takes care of that "and so on". This is most easily done in a recursive fashion.

Parameters **CurrentTensorRank** (*int*) – Tensor rank

Returns Dimension of tensor space list: List compatible tensors

Return type int

Functions.**CloseEnough**(*A*, *W*, *B*, *M*, *RelTol=1e-05*)
Test the condition

$$\left|\sum_j A_{ij}w_j - b_i\right| < \varepsilon\sqrt{\sum_j \left(A_{ij}w_j\right)^2 + \left(\frac{m_i}{2}\right)^2 b_i} \text{ for all } i$$

Parameters

- **A** (*numpy.ndarray*) – Matrix A
- **W** (*numpy.ndarray*) – Vector $\vec{w}$
- **B** (*numpy.ndarray*) – Vector $\vec{b},\ b_i = c_{\mathrm{s}}^{m_i}$

- **M** (*numpy.ndarray*) – Vector $\vec{m}$
- **RelTol** (*float*) – Relative tolerance ε

Returns True if condition satisfied, False otherwise.

Return type bool

Functions.**ComputeSubshell**(*Velocity*, *Group*)

Compute the (sub)shell that is being spanned by Velocity wrt. Group.

Parameters

- **Velocity** (*numpy.ndarray*) – Velocity vector
- **Group** (*list*) – List of transformation matrices that form the cubic group

Returns List of velocity vectors that form the velocity shell spanned by Group

Return type list

Functions.**Contains**(*Array*, *List*)

Checks whether given numpy array is contained in list. The all() function is defined on numpy arrays and evaluates True if all elements are True.

Parameters

- **Array** (*numpy.ndarray*) – numpy array
- **List** (*list*) – List of numpy arrays

Returns True if Array is contained in List, False otherwise.

Return type bool

Functions.**ContainsInSublist**(*Array*, *ListOfLists*)

Checks whether given numpy array is contained in a list of lists. The all() function is defined on numpy arrays and evaluates True if all elements are True.

Parameters

- **Array** (*numpy.ndarray*) – numpy array
- **List** (*list*) – List of Lists of numpy arrays

Returns True if Array is contained in ListOfLists, False otherwise.

Return type bool

Functions.**DoubleFactorial**(*Number*)

Implementation of the double factorial. $n!! = n(n-2)(n-4)\ldots$

Parameters **Number** (*int*) – Number

Returns Number !!

Return type int

Functions.**Echo**(*String='\n'*, *Linewidth=70*)

Formatted printing If QUIET is set (i.e. via command line option –quiet) this is suppressed.

Parameters

- **String** (*str*) – String to be printed to the console
- **Linewidth** (*int*) – Maximum line width of console output

Returns None

Functions.**EchoError**(*String='\n'*, *Linewidth=70*)
Formatted printing Prints irregardless of value of QUIET

Parameters

- **String** (*str*) – String to be printed to the console
- **Linewidth** (*int*) – Maximum line width of console output

Returns None

Functions.**EnterWeights**(*TotalNumberOfShells*, *i_par=0*)
Gets vector of weights from user input

Parameters

- **TotalNumberOfShells** (*int*) – Number of shells INCLUDING zero-shell
- **i_par** (*int*) – Solution vector index (parametric solutions are written as $\vec{w} = \vec{w}_0 + \lambda_1\vec{w}_1 + \lambda_2\vec{w}_2 + \ldots$)

Returns Vector of weights

Return type numpy.ndarray

Functions.**EvaluateWeights**(*W0List*, *SolutionMatrix*, *CsSquared*)
Calculate numerical weights from their polynomial coefficients

Parameters

- **W0List** (*list*) – List of polynomial coefficients for zero shell
- **SolutionMatrix** (*numpy.ndarray*) – Solution matrix Q
- **CsSquared** (*float*) – Speed of sound squared

Returns List of numerical weights $[w_0, w_1, \ldots]$

Return type list

Functions.**FillLeftHandSide**(*SpacialDimension*, *MaxTensorRank*, *ListOfTensorDimensions*, *TotalNumberOfShells*, *GrandTotalList*)
Construct the $R \times N_s$ matrix A

Parameters

- **SpacialDimension** (*int*) – Spacial dimension
- **MaxTensorRank** (*int*) – Highest tensor rank (M) to consider.
- **ListOfTensorDimensions** (*list*) – List of the dimensions of tensor space for tensors of rank $2, 4, \ldots, M$.
- **TotalNumberOfShells** (*int*) – Total number of velocity shells N_s
- **GrandTotalList** (*list*) – List of lists. The s-th sublist contains all velocity vectors of shell s.

Returns Matrix A

Return type numpy.ndarray

Functions.**FillRightHandSide**(*MaxTensorRank*, *ListOfTensorDimensions*)
Construct the matrix D : $D_{r\mu} = \delta_{m_r\mu}$

Parameters

- **MaxTensorRank** (*int*) – Maximum tensor rank M
- **ListOfTensorDimensions** (*list*) – List of the dimensions of tensor space for tensors of rank $2, 4, \ldots, M$.

Returns Matrix D

Return type numpy.ndarray

Functions.**FindRangeOfExistence**(*W0List*, *SolutionMatrix*)

Make use of the function "roots" that needs the coefficients in reverse order, in order to find the roots of the weight polynomials. If inbetween two roots all weights are positive, add them to list CompressedRoots

Parameters

- **W0List** (*list*) – List of polynomial coefficients for zero shell
- **SolutionMatrix** (*numpy.ndarray*) – Solution matrix Q

Returns List CompressedRoots of roots that form valid intervals for the speed of sound.

Return type list

Functions.**FindVelocities**(*SpacialDimension*, *SquaredVelocity*)

Scans the cubic lattice for lattice velocity with squared length SquaredVelocity

Parameters

- **SpacialDimension** (*int*) – SpacialDimension
- **SquaredVelocity** (*int*) – Squared length of compatible lattice velocities

Returns List of compatible lattice velocity vectors

Return type list

Functions.**Frexp10**(*Float*)

Returns exponent and mantissa in base 10

Parameters **Float** (*float*) – Original number

Returns (Mantissa, Exponent)

Return type tuple

Functions.**GetGroup**(*SpacialDimension*)

Compute the cubic group. Each transformation matrix in the group is made up of 2d unit vectors of type $(0 \ldots 0, +-1, 0 \ldots 0)$. We will identify a vector with i-th component 1 and 0 elsewhere by the number i. A vector with i-th component -1 and 0 elsewhere is identified by the number $-i$. The cubic group then consists of all orthogonal matrices, with columns made up of the above unit vectors. In general there are $d!2^d$ such transformations.

Parameters **SpacialDimension** (*int*) – Spacial dimension

Returns A list of all transformation matrices in the cubic group

Return type list

Functions.**GetListOfSubshells**(*Shell*, *Group*)

Applies all group transformations to all velocities in shell and returns all distinct shells that result.

Parameters

- **Shell** (*list*) – List of velocity vectors
- **Group** (*list*) – List of transformation matrices that form the cubic group

Returns List of distinct velocity shells

Return type list

Functions.**IndicatorFunction**(*W0List*, *SolutionMatrix*, *CsSquared*)
Tests, whether solution yields all positive weights.

Parameters

- **W0List** (*list*) – List of polynomial coefficients for zero shell
- **SolutionMatrix** (*numpy.ndarray*) – Solution matrix Q
- **CsSquared** (*float*) – Speed of sound squared

Returns True if all weights positive, False otherwise

Return type bool

Functions.**LatticeSum**(*RandomVector*, *ListOfVelocities*, *TensorRank*)
Calculate the sum $A_{rs} = \frac{1}{(m_r-1)!!} \sum_{i \in s} (\vec{c}_i \cdot \vec{n}_r)^{m_r}$

for tensor rank r and shell s.

Parameters

- **RandomVector** (*numpy.ndarray*) – r-th random unit vector
- **ListOfVelocities** (*list*) – List of velocity vectors in shell s
- **TensorRank** (*int*) – Tensor rank r

Returns A_{rs}

Return type float

Functions.**MakeRandomVector**(*SpacialDimension*)
Generate a random vector uniformly distributed on the unit sphere.

Parameters **SpacialDimension** (*int*) – Spacial dimension d

Returns Vector of length one with random orientation in d-dimensional space.

Return type list

Functions.**OutputRangeOfExistence**(*CompressedRoots*)
Screen output of the intervals of the speed of sound that yield all positive weights.

Parameters **CompressedRoots** (*list*) – List of roots that form valid intervals for the speed of sound.

Returns Number of valid intervals

Return type int

Functions.**ParseArguments**()
Function to parse command line options.

Returns Dictionary of command line options

Return type dict

Functions.**RatApprox**(x)
Calculates numerator and denominator for a floating point number x and returns the output as a string.

Parameters **x** (*float*) – Number to approximate as fraction.

Returns Approximate fraction as string

Return type str

Functions.**TestSolution**(*GrandTotalList*, *MaxTensorRank*, *SpacialDimension*, *ListOfTensorDimensions*, *Solution=None*, *RelTol=1e-05*)

Test validity of the equation $A\vec{w} = \vec{b}$ for given weights w and speed of sound c_s^2. A solution is deemed valid, if

$$\left| \sum_j A_{ij} w_j - b_i \right| < \varepsilon_0 + \varepsilon \sqrt{\left(\sum_j A_{ij} w_j\right)^2 + \left(\frac{m_i}{2}\right)^2 b_i} \text{ for all } i$$

The weights can be given as a linear parametric equation

$$\vec{w} = \vec{w}_0 + \lambda_1 \vec{w}_1 + \lambda_2 \vec{w}_2 + \dots$$

Parameters

- **GrandTotalList** (*list*) – List of lists. The s-th sublist contains all velocity vectors of shell s.
- **MaxTensorRank** (*int*) – Maximum tensor rank M
- **SpacialDimension** (*int*) – SpacialDimension
- **ListOfTensorDimensions** (*list*) – List of the dimensions of tensor space for tensors of rank $2, 4, \dots, M$.
- **Solution** (*list*) – Solution that is to be tested in the form `[CsSquared, [[w_00, w_01, ...], [[w_10, w_11, ...], ...]` If None is given, the user is prompted to enter a solution by hand.
- **RelTol** (*float*) – Relative tolerance ε

Returns 0 if solution is valid, otherwise 1

Return type int

Functions.**ToMatrix**(*Array*)

Convert an array of unit vector representations to proper matrix. For example `[0,2,1]` will be converted to `[[1,0,0], [0,0,1], [0,1,0]]`.

Parameters **Array** (*numpy.ndarray*) – Array of integers

Returns Transformation matrix

Return type numpy.ndarray

Functions.**Type**(*Shell*)

Method to determine typical velocity vector for Shell.

Parameters **Shell** (*list*) – List of velocity vectors, e.g. `[[0,1],[1,0]]`

Returns Typical velocity vector, e.g. `(0,1)`

Return type tuple

Functions.**WriteLatexNumber**(*Value*, *Outfile*, *Precision=8*, *Rational=False*)

Write Value to `Outfile` in a Latex compatible way

Parameters

- **Value** (*float*) – Value
- **Outfile** – Output file
- **Precision** (*int*) – Number of digits
- **Rational** (*bool*) – Approximate numbers by fractions

Returns None

Functions.**WriteLatexTables**(*CompressedRoots*, *W0List*, *SolutionMatrix*, *GrandTotalList*, *MaxTensorRank*, *Precision=8*, *Rational=False*, *Filename='latex_tables.tex'*)

Write unique solution to a file in form of a latex table. This will append to any existing file.

Parameters

- **CompressedRoots** (*list*) – List of roots that form the valid intervals for the speed of sound
- **W0List** (*list*) – List of polynomial coefficients for zero shell
- **SolutionMatrix** (*numpy.ndarray*) – Solution matrix Q
- **GrandTotalList** (*list*) – List of lists. The s-th sublist contains all velocity vectors of shell s.
- **MaxTensorRank** (*int*) – Maximum tensor rank M
- **Precision** (*int*) – Number of digits
- **Rational** (*bool*) – Approximate numbers by fractions

Returns None

Functions.**YesNo**(*Question*)

Ask for yes or no answer and return a Boolean.

Parameters **Question** (*str*) – String that is printed when function is called.

Returns True, if answer is in `["YES", "Y", "yes", "y", "Yes", \CR]` False, if answer is in `["NO", "N", "no", "n", "No"]`

Return type bool

PYTHON MODULE INDEX

INDEX

B.2 Analysis software

In order to process the large amount of data produced by the Molecular Dynamics simulation, custom software in C++ was developed. Analysis programs are based on a hierarchy of libraries offering methods for the iterative reading of trajectory files and data structures and data structures for chain molecule configurations. Presented below is the documentation of this software that was automatically generated from the docstrings in the code with the documentation tool `doxygen` [113].

1 README

This project was developed for the analysis of Molecular Dynamics trajectories produced by the simulation package ESPResSo++. The main focus is on trajectories in the `.xyz` format which are concatenations of individual `.xyz` files. As simulations of large systems can easily produce trajectories in the tens or even hundreds of gigabytes, efficiency becomes an important factor in processing these files. Because of this, C++ was chosen as a programming language to develop the analysis tools which are based on a hierarchy of libraries:

trajectory.hpp (p. 40) defines a trajectory reader which can efficiently iterate large xyz trajectories. It starts reading the file at the first configuration and allows to jump ahead without explicitly reading every line into memory.

From every position a **Frame** (p. 3) object can be read which contains particle types, coordinates and optionally velocities. This class is defined in the fole **frame.hpp** (p. 35) which also defines various functions to modify and analyze the configurations. This includes the calculation of the dynamic structure factor by mapping the configuration onto a density lattice and performing a Fast Fourier Transform, as well as the calculation of the Minkowski functionals in three dimensions.

By the **Molecule** (p. 22) class introduced in **molecule.hpp** (p. 39), a **Frame** (p. 3) can be interpreted as a set of linear chain molecules (polymers). Because of the molecules' chain-like nature, it is enough to store references to the first and last particles. Therefore, a **Molecule** (p. 22) object can operate on the data stored in a **Frame** (p. 3) object while needing very little additional memory for itself. The **Molecule** (p. 22) class provides various functions to analyze the properties of polymer molecules.

Properties calculated from **Frame** (p. 3) objects and **Molecule** (p. 22) objects at successive points in time can be stored and processed with the **Timeseries** (p. 26) class.

The classes introduced above allow to easily create compact tailor-made programs for the analysis and manipulation of large xyz trajectories.

In the below example the average squared end-to-end vector of all molecules is calculated at successive configurations in the trajectory and written to an output file:

```
#include <iomanip>
#include "trajectory.hpp"
#include "couf.hpp"
using namespace std;
int main(int argc, char **argv)
{
    // set output precision
    constexpr size_t precision = 10;
    // get input file from command line
    const char *infile  = argv[1];
    // get molecule size from command line argument
    const size_t particles_per_molecule = static_cast<size_t>(
          atoi(couf::parse_arguments(argc, argv, "--ppm", "0")));
    // open output file
    ofstream outfile("ete_squared.dat");
    outfile.precision(precision);
    // define trajectory reader with molecule size
    Trajectory traj{infile, particles_per_molecule};
    // iterate Frames
    while(!traj.is_null())
    {
        outfile « scientific « setw(precision + 2);
        // write step to outfile
        outfile « traj.index() « ' ';
        // calculate the average squared end-to-end vector of all molecules in
        // the current frame and write to outfile
        outfile « traj->mean(&Molecule::end_to_end_squared) « '\n';
        // advance to next frame according to command line arguments
        traj.loop_advance(argc, argv);
    }
    outfile.close();
    exit(0);
}
```

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

4 Class Documentation

4.1 Frame Class Reference

`#include <frame.hpp>`

Public Member Functions

- **Frame** (Real3D **box**)
- **Frame** (std::vector< Real3D > coordinates, std::vector< Real3D > velocities, Real3D **box**, size_t **particles_per_molecule**=0)
- **Frame** (std::ifstream &stream, const size_t **particles_per_molecule**=0, const char format='x')
- **Frame** (const char *filename, const size_t **particles_per_molecule**=0, const char format='x')
- void **read_xyz** (const std::string &filename, const size_t **particles_per_molecule**=0)
- void **read_xyz** (std::ifstream &stream, const size_t **particles_per_molecule**=0)
- bool **has_velocities** () const
- bool **is_null** () const
- size_t **size** () const
- Real3D **box** () const
- double **box** (size_t i) const
- bool **is_square** () const
- size_t **particles_per_molecule** () const
- size_t **number_of_molecules** () const
- auto **c_cbegin** () const
- auto **c_cend** () const
- auto **v_cbegin** () const
- auto **v_cend** () const
- auto **t_cbegin** () const
- auto **t_cend** () const
- Real3D **coordinate** (const size_t index) const

- Real3D **folded_coordinate** (const size_t index) const
- Real3D **velocity** (const size_t index) const
- int **type** (const size_t index) const
- **Molecule** **molecule** (const size_t index) const
- void **set_box** (const Real3D **box**)
- void **set_particles_per_molecule** (const size_t **particles_per_molecule**)
- void **set_number_of_molecules** (const size_t **number_of_molecules**)
- void **set_types** (size_t max_type)
- void **set_types** ()
- void **reduce_types** (size_t max_type)
- void **add_particle** (Real3D **coordinate**, int **type**=0)
- void **add_particle** (Real3D **coordinate**, Real3D **velocity**, int **type**=0)
- void **add_molecule** (const **Molecule** m, const Real3D displacement=Real3D(0.))
- void **remove_particle** (const size_t index)
- void **clear** ()
- void **fold** ()
- void **fold_2d** ()
- void **shift_coordinates** (const Real3D shift)
- void **scale_box** (const double factor)
- void **crop_box** (const double xmin, const double xmax, const double ymin, const double ymax, const double zmin, const double zmax)
- void **crop_box** (const double lx, const double ly, const double lz)
- void **rotate_x** (const double angle)
- void **rotate_y** (const double angle)
- void **rotate_z** (const double angle)
- void **rotate** (const Real3D v, const double angle)
- std::vector< **Frame** > **divide_box** (const int nx, const int ny=1, const int nz=1) const
- **Frame** **slice** (const double thickness, const Real3D x0=Real3D(0.), const Real3D norm=Real3D(-1., -1., 2.)) const
- **Frame** **slice_square** (const double thickness, const double height=0.) const
- **Frame** **slice_rectangle** (const double thickness) const
- **Frame** **multiply** (const size_t mx=2, const size_t my=2, const size_t mz=2) const
- bool **consistent** () const
- double **mean_distance** () const
- double **smallest_distance** () const
- template<typename T >
  T **mean** (T(Molecule::*f)(void)) const
- template<typename T >
  std::vector< T > **vector** (T(Molecule::*f)(void)) const
- double **max_bond_length** () const
- double **mean_squared_displacement_cm** (**Frame** earlier_frame)
- double **mean_squared_displacement** (**Frame** earlier_frame)
- void **write_xyz** (const char *filename, const bool append=false, const size_t precision=11) const
- void **write_vtk** (const char *filename, const bool append=false, const size_t precision=11) const
- void **write_pdb** (const char *filename, const bool append=false) const
- void **write_binary** (const char *filename, const bool append=false) const
- void **make_sphere** (const double radius)
- void **make_cube** (const double L)

Friends

- std::ostream & **operator<<** (std::ostream &os, const **Frame** &frame)
- bool **operator==** (const **Frame** &lhs, const **Frame** &rhs)

4.1.1 Detailed Description

The **Frame** (p. 3) class that stores particle coordinates and velocities as well as the box dimensions. By setting _particles_per_molecule to a nonzero value N, the system is considered to consist of linear chain molecules of the given length N. Note, that the coordinates have to be ordered accordingly, i.e. the first particle is the first particle in chain one and the last particle is the last particle in the last chain and so on. If _particles_per_molecule_ == 0 all particles are considered to be in one large molecule (zero molecules if system is empty). Each particle can also be assigned an integer type.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Frame()

```
Frame::Frame (
        const char * filename,
        const size_t particles_per_molecule = 0,
        const char format = 'x' )  [inline]
```

Initialize **Frame** (p. 3) with given three-dimensional box. Particle coordinates and velocities are read from an input file using designated functions. A molecule size can be given as well.

Parameters

in	*stream*	Input file.
in	*particles_per_molecule*	Number of particles in one molecule.
in	*format*	File format. 'x' corresponds to .xyz format. At the moment only .xyz files are supported.

4.1.3 Member Function Documentation

4.1.3.1 add_molecule()

```
void Frame::add_molecule (
        const  Molecule m,
        const Real3D displacement = Real3D(0.)  )  [inline]
```

Add a **Molecule** (p. 22) object to the **Frame** (p. 3). The coordinates of the particles in the molecule can optionally be displaced by some vector.

Parameters

in	*m*	**Molecule** (p. 22) object.
in	*displacement*	Displacement vector.

4.1.3.2 add_particle() [1/2]

```
void Frame::add_particle (
```

```
            Real3D coordinate,
            int type = 0 )  [inline]
```

Add a particle with given type and coordinate vector to the **Frame** (p. 3).

Parameters

in	*coordinate*	Coordinate vector.
in	*type*	Particle type.

4.1.3.3 add_particle() [2/2] void Frame::add_particle (

```
            Real3D coordinate,
            Real3D velocity,
            int type = 0 )  [inline]
```

Add a particle with given type, coordinate and velocity vector to the **Frame** (p. 3).

Parameters

in	*coordinate*	Coordinate vector.
in	*velocity*	Velocity vector.
in	*type*	Particle type.

4.1.3.4 box() [1/2] Real3D Frame::box ( ) const [inline]

Returns

Box dimensions.

4.1.3.5 box() [2/2] double Frame::box (

```
            size_t i ) const  [inline]
```

Parameters

in	*i*	Box dimension in question.

Returns

Box size in dimension i.

4.1.3.6 c_cbegin() `auto Frame::c_cbegin ( ) const [inline]`

Returns

Const-iterator of coordinate vector start.

4.1.3.7 c_cend() `auto Frame::c_cend ( ) const [inline]`

Returns

Const-iterator of coordinate vector end.

4.1.3.8 clear() `void Frame::clear ( ) [inline]`

Remove all particles from the system and set box size to zero in all directions.

4.1.3.9 consistent() `bool Frame::consistent ( ) const [inline]`

Test **Frame** (p. 3) for consistency by checking whether the number of particles is compatible with the total number of particles in the system.

Returns

true if consistent, false otherwise.

4.1.3.10 coordinate() `Real3D Frame::coordinate ( const size_t index ) const [inline]`

Parameters

in	*index*	Index of the particle in question.

Returns

Coordinate vector of the particle.

4.1.3.11 crop_box() [1/2] `void Frame::crop_box ( const double lx,`

```
            const double ly,
            const double lz )  [inline]
```

Remove all particles that are not within the box defined by the given limits.

Parameters

in	*lx*	Maximum x-coordinate.
in	*ly*	Maximum y-coordinate.
in	*lz*	Maximum z-coordinate.

4.1.3.12 crop_box() [2/2]

```
void Frame::crop_box (
        const double xmin,
        const double xmax,
        const double ymin,
        const double ymax,
        const double zmin,
        const double zmax )  [inline]
```

Remove all particles that are not within the box defined by the given limits.

Parameters

in	*xmin*	Minimum x-coordinate.
in	*xmax*	Maximum x-coordinate.
in	*ymin*	Minimum y-coordinate.
in	*ymax*	Maximum y-coordinate.
in	*zmin*	Minimum z-coordinate.
in	*zmax*	Maximum z-coordinate.

4.1.3.13 divide_box()

```
std::vector< Frame> Frame::divide_box (
        const int nx,
        const int ny = 1,
        const int nz = 1 ) const  [inline]
```

Divide box into nx x ny x nz equal-sized subboxes.

Parameters

in	*nx*	Number of subboxes in the x-direction.
in	*ny*	Number of subboxes in the y-direction.
in	*nz*	Number of subboxes in the z-direction.

Returns

Vector of subboxes.

4.1.3.14 fold()

```
void Frame::fold ( )  [inline]
```

Fold all coordinates in the **Frame** (p. 3) according to the periodic boundary conditions.

4.1.3.15 fold_2d() `void Frame::fold_2d ( ) [inline]`

Fold all coordinates in the **Frame** (p. 3) according to the periodic boundary conditions while ignoring the z-component. This is useful for pseudo-2d systems where the z-coordinate fluctuates around 0.

4.1.3.16 folded_coordinate() `Real3D Frame::folded_coordinate ( const size_t index ) const [inline]`

Parameters

in	*index*	Index of the particle in question.

Returns

Folded coordinate vector of the particle.

4.1.3.17 has_velocities() `bool Frame::has_velocities ( ) const [inline]`

Returns

True if **Frame** (p. 3) contains velocities.

4.1.3.18 is_null() `bool Frame::is_null ( ) const [inline]`

Returns

True if **Frame** (p. 3) does not contain any particles.

4.1.3.19 is_square() `bool Frame::is_square ( ) const [inline]`

Returns

True if box is cubic.

4.1.3.20 make_cube() `void Frame::make_cube ( const double L ) [inline]`

Make a cube of particles with edge length on a cubic lattice with lattice constant 1.

Parameters

in	*L*	Edge length.

4.1.3.21 make_sphere() `void Frame::make_sphere ( const double radius ) [inline]`

Make a sphere of particles with given radius on a cubic lattice with lattice constant 1.

Parameters

in	*radius*	Radius of the sphere.

4.1.3.22 max_bond_length() `double Frame::max_bond_length ( ) const [inline]`

Returns

Size of the largest bond in the system.

4.1.3.23 mean() `template<typename T > T Frame::mean ( T(Molecule::*)(void) f ) const [inline]`

Calculates mean value of function f for all molecules that are contained in the system.

Parameters

in	*f*	function that is to applied to the molecules.

4.1.3.24 mean_distance() `double Frame::mean_distance ( ) const [inline]`

Returns

Average distance between particles.

4.1.3.25 mean_squared_displacement() `double Frame::mean_squared_displacement ( Frame earlier_frame ) [inline]`

Calculate the bead mean-squared displacement with respect to reference frame.

Parameters

in	*Earlier_frame*	Reference frame.

Returns

Mean-squared displacement.

4.1.3.26 molecule()

```
Molecule Frame::molecule (
        const size_t index ) const  [inline]
```

Parameters

in	*index*	Index of the **Molecule** (p. 22) in question.

Returns

Molecule (p. 22) object corresponding to given index.

4.1.3.27 multiply()

```
Frame Frame::multiply (
        const size_t mx = 2,
        const size_t my = 2,
        const size_t mz = 2 ) const  [inline]
```

This produces a frame that is extended by periodic images of the original frame.

Parameters

in	*mx*	How many copies are in the x-direction.
in	*my*	How many copies are in the y-direction.
in	*mz*	How many copies are in the z-direction.

Returns

Multiplied frame.

4.1.3.28 number_of_molecules()

```
size_t Frame::number_of_molecules ( ) const  [inline]
```

Returns

Number of molecules in the system.

4.1.3.29 particles_per_molecule() size_t Frame::particles_per_molecule ( ) const [inline]

Returns

Molecule (p. 22) size.

4.1.3.30 read_xyz() [1/2] void Frame::read_xyz (const std::string & *filename*, const size_t *particles_per_molecule = 0*) [inline]

Read types, coordinates and velocities from and .xyz file.

Parameters

in	*filename*	Filename of the .xyz file.
in	*particles_per_molecule*	**Molecule** (p. 22) size.

4.1.3.31 read_xyz() [2/2] void Frame::read_xyz (std::ifstream & *stream*, const size_t *particles_per_molecule = 0*) [inline]

Read types, coordinates and velocities from and .xyz filestream.

Parameters

in	*stream*	Ifstream of the open .xyz file.
in	*particles_per_molecule*	**Molecule** (p. 22) size.

4.1.3.32 reduce_types() void Frame::reduce_types (size_t *max_type*) [inline]

Iterate all particle types. If a certain maximum value is exceeded the enumeration starts again at 0.

Parameters

in	*max_type*	All set types are smaller than this value.

4.1.3.33 remove_particle() void Frame::remove_particle (const size_t *index*) [inline]

Remove particle from the **Frame** (p. 3).

Parameters

in	*index*	Index of the particle that is to be removed.

4.1.3.34 rotate()

```
void Frame::rotate (
        const Real3D v,
        const double angle )  [inline]
```

Rotate all coordinates by given angle about a given axis.

Parameters

in	*v*	Rotation axis.
in	*angle*	Rotation angle.

4.1.3.35 rotate_x()

```
void Frame::rotate_x (
        const double angle )  [inline]
```

Rotate all coordinates about the x-axis by given angle.

Parameters

in	*angle*	Rotation angle.

4.1.3.36 rotate_y()

```
void Frame::rotate_y (
        const double angle )  [inline]
```

Rotate all coordinates about the y-axis by given angle.

Parameters

in	*angle*	Rotation angle.

4.1.3.37 rotate_z()

```
void Frame::rotate_z (
        const double angle )  [inline]
```

Rotate all coordinates about the y-axis by given angle.

Parameters

in	*angle*	Rotation angle.

4.1.3.38 scale_box() `void Frame::scale_box ( const double factor ) [inline]`

Scale box size and the coordinates of the particles by a given factor.

Parameters

in	*factor*	Scaling factor.

4.1.3.39 set_box() `void Frame::set_box ( const Real3D box ) [inline]`

Set box dimensions.

Parameters

in	*box*	Vector of box dimensions.

4.1.3.40 set_number_of_molecules() `void Frame::set_number_of_molecules ( const size_t number_of_molecules ) [inline]`

Set number of molecules in the system. The routine checks whether or not the given value is compatible with the total number of particles. If this is the case, the molecule size is set accordingly.

Parameters

in	*particles_per_molecule*	**Molecule** (p. 22) size.

4.1.3.41 set_particles_per_molecule() `void Frame::set_particles_per_molecule ( const size_t particles_per_molecule ) [inline]`

Set molecule size.

Parameters

in	*particles_per_molecule*	**Molecule** (p. 22) size.

4.1.3.42 set_types() [1/2] void Frame::set_types ( ) [inline]

Set particle types according to their index. I.e. particle 0 has type 0 and so on.

4.1.3.43 set_types() [2/2] void Frame::set_types (size_t *max_type*) [inline]

Set particle types according to their index. I.e. particle 0 has type 0 and so on. If a certain maximum value is exceeded the enumeration starts again at 0. This can be useful for visualization with VMD which only allows for a finite number of types.

Parameters

in	*max_type*	All set types are smaller than this value.

4.1.3.44 shift_coordinates() void Frame::shift_coordinates (const Real3D *shift*) [inline]

Shift all coordinates by a given vector. This was originally implemented to test for translational invariance.

Parameters

in	*shift*	Shift vector.

4.1.3.45 size() size_t Frame::size ( ) const [inline]

Returns

Number of particles in the **Frame** (p. 3).

4.1.3.46 slice() **Frame** Frame::slice (const double *thickness*, const Real3D *x0 = Real3D(0.)*, const Real3D *norm = Real3D(-1., -1., 2.)*) const [inline]

Takes a slice of given thickness along the plane given by an initial point x0 and a normal vector norm out of the configuration.

Parameters

in	*thickness*	Thickness of the slice.
in	*x0*	Initial point.
in	*norm*	Normal vector.

Returns

Sliced **Frame** (p. 3).

4.1.3.47 slice_rectangle()

```
Frame Frame::slice_rectangle (
        const double thickness ) const  [inline]
```

Takes a rectangular slice of given thickness out of the configuration by slicing the box diagonally. This increases the surface of the slice.

Parameters

in	*thickness*	Thickness of the slice.
in	*height*	z-coordinate of the xy plane.

Returns

Sliced **Frame** (p. 3).

4.1.3.48 slice_square()

```
Frame Frame::slice_square (
        const double thickness,
        const double height = 0.  )  const  [inline]
```

Takes a square slice of given thickness around the xy-plane at given height out of the configuration.

Parameters

in	*thickness*	Thickness of the slice.
in	*height*	z-coordinate of the xy plane.

Returns

Sliced **Frame** (p. 3).

4.1.3.49 smallest_distance()

```
double Frame::smallest_distance ( ) const  [inline]
```

Returns

Smallest distance between particles.

4.1.3.50 t_cbegin() `auto Frame::t_cbegin ( ) const [inline]`

Returns

Const-iterator of type vector start.

4.1.3.51 t_cend() `auto Frame::t_cend ( ) const [inline]`

Returns

Const-iterator of type vector end.

4.1.3.52 type() `int Frame::type ( const size_t index ) const [inline]`

Parameters

in	*index*	Index of the particle in question.

Returns

Type of the particle.

4.1.3.53 v_cbegin() `auto Frame::v_cbegin ( ) const [inline]`

Returns

Const-iterator of velocity vector start.

4.1.3.54 v_cend() `auto Frame::v_cend ( ) const [inline]`

Returns

Const-iterator of velocity vector end.

4.1.3.55 vector() template<typename T >

```
std::vector<T> Frame::vector (
            T(Molecule::*)(void) f ) const  [inline]
```

Returns the value of function f for each molecule in the system in form of a vector.

Parameters

in	*f*	function that is to be applied to the molecules.

Returns

vector of function values.

4.1.3.56 velocity()

```
Real3D Frame::velocity (
            const size_t index ) const  [inline]
```

Parameters

in	*index*	Index of the particle in question.

Returns

Velocity vector of the particle.

4.1.3.57 write_binary()

```
void Frame::write_binary (
            const char * filename,
            const bool append = false ) const  [inline]
```

Write configuration to binary file.

Parameters

in	*filename*	Output filename.
in	*append*	Whether or not to append to existing files.
in	*precision*	Floating point precision of the output.

4.1.3.58 write_pdb()

```
void Frame::write_pdb (
            const char * filename,
            const bool append = false ) const  [inline]
```

Write configuration to .pdb file.

Parameters

in	*filename*	Output filename.
in	*append*	Whether or not to append to existing files.
in	*precision*	Floating point precision of the output.

4.1.3.59 write_vtk()

```
void Frame::write_vtk (
        const char * filename,
        const bool append = false,
        const size_t precision = 11 ) const  [inline]
```

Write configuration to .vtk file.

Parameters

in	*filename*	Output filename.
in	*append*	Whether or not to append to existing files.
in	*precision*	Floating point precision of the output.

4.1.3.60 write_xyz()

```
void Frame::write_xyz (
        const char * filename,
        const bool append = false,
        const size_t precision = 11 ) const  [inline]
```

Write configuration to .xyz file.

Parameters

in	*filename*	Output filename.
in	*append*	Whether or not to append to existing files.
in	*precision*	Floating point precision of the output.

4.1.4 Friends And Related Function Documentation

4.1.4.1 operator<<

```
std::ostream& operator<< (
        std::ostream & os,
        const  Frame & frame )  [friend]
```

Give **Frame** (p. 3) information when the **Frame** (p. 3) object is passed i.e. to cout.

4.1.4.2 operator== bool operator== (const **Frame** & *lhs*, const **Frame** & *rhs*) [friend]

Compare two frame objects. Returns true if coordinates, velocities, box dimensions and molecule sizes are the same by value and false otherwise.

The documentation for this class was generated from the following file:

- lib/ **frame.hpp**

4.2 Molecule Class Reference

Public Member Functions

- **Molecule** (r3dcit coordinates_begin, r3dcit coordinates_end, r3dcit velocities_begin, r3dcit velocities_end)
- **Molecule** (r3dcit coordinates_begin, r3dcit coordinates_end)
- size_t **size** () const
- bool **has_velocities** () const
- bool **is_null** () const
- Real3D **coordinate** (size_t index) const
- Real3D **velocity** (size_t index) const
- Real3D **bond** (const int index) const
- double **bond_length** (const int index) const
- bool **consistent** () const
- Real3D **center_of_mass** ()
- Real3D **rouse_mode_0** ()
- Real3D **rouse_mode** (const size_t p)
- double **radius_of_gyration_squared** ()
- Real3D **end_to_end** ()
- double **end_to_end_squared** ()
- double **mean_squared_displacement** (const **Molecule** &earlier_molecule) const
- double **mean_bond_length** () const
- double **max_bond_length** () const
- **Molecule** & **operator=** (const **Molecule** &)=delete

Friends

- std::ostream & **operator**<< (std::ostream &os, const **Molecule** &molecule)

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Molecule() [1/2] Molecule::Molecule (r3dcit *coordinates_begin*, r3dcit *coordinates_end*, r3dcit *velocities_begin*, r3dcit *velocities_end*) [inline]

Initialize **Molecule** (p. 22) with references to the start/end coordinates and velocities.

4.2.1.2 Molecule() [2/2] Molecule::Molecule (r3dcit *coordinates_begin*, r3dcit *coordinates_end*) [inline]

Initialize **Molecule** (p. 22) with references to the start/end coordinates.

4.2.2 Member Function Documentation

4.2.2.1 bond() Real3D Molecule::bond (const int *index*) const [inline]

Parameters

in	*index*	Index within the **Molecule** (p. 22) of the bond in question.

Returns

Bond vector of the particle with the given index.

4.2.2.2 bond_length() double Molecule::bond_length (const int *index*) const [inline]

Parameters

in	*index*	Index within the **Molecule** (p. 22) of the bond in question.

Returns

Bond length of the particle with the given index.

4.2.2.3 center_of_mass() Real3D Molecule::center_of_mass ( ) [inline]

Returns

The center of mass vector.

4.2.2.4 consistent() bool Molecule::consistent ( ) const [inline]

Test for any bonds that might be too large.

4.2.2.5 coordinate() Real3D Molecule::coordinate (size_t *index*) const [inline]

Parameters

in	*index*	Index within the **Molecule** (p. 22) of the particle in question.

Returns

Coordinate vector of the particle with the given index.

4.2.2.6 end_to_end() `Real3D Molecule::end_to_end ( ) [inline]`

Returns

The end-to-end vector.

4.2.2.7 end_to_end_squared() `double Molecule::end_to_end_squared ( ) [inline]`

Returns

The squared end-to-end vector.

4.2.2.8 has_velocities() `bool Molecule::has_velocities ( ) const [inline]`

Returns

Whether or not the **Molecule** (p. 22) stores velocities.

4.2.2.9 is_null() `bool Molecule::is_null ( ) const [inline]`

Returns

Whether or not there are any particles in the **Molecule** (p. 22).

4.2.2.10 max_bond_length() `double Molecule::max_bond_length ( ) const [inline]`

Returns

The largest bond length.

4.2.2.11 mean_bond_length() double Molecule::mean_bond_length ( ) const [inline]

Returns

The average bond length.

4.2.2.12 operator=() **Molecule**& Molecule::operator= (const **Molecule** &) [delete]

Disable the assignment operator.

4.2.2.13 radius_of_gyration_squared() double Molecule::radius_of_gyration_squared ( ) [inline]

Returns

The squared radius of gyration.

4.2.2.14 rouse_mode() Real3D Molecule::rouse_mode (const size_t *p*) [inline]

Parameters

in	*p*	Index of the Rouse mode.

Returns

The corresponding Rouse mode.

4.2.2.15 size() size_t Molecule::size ( ) const [inline]

Returns

The number of beads in the **Molecule** (p. 22).

4.2.2.16 velocity() Real3D Molecule::velocity (size_t *index*) const [inline]

Parameters

in	*index*	Index within the **Molecule** (p. 22) of the particle in question.

Returns

Velocity vector of the particle with the given index.

4.2.3 Friends And Related Function Documentation

4.2.3.1 operator<< `std::ostream& operator<< ( std::ostream & os, const Molecule & molecule ) [friend]`

Print **Molecule** (p. 22) info.

The documentation for this class was generated from the following file:

- lib/ **molecule.hpp**

4.3 Timeseries< T > Class Template Reference

`#include <timeseries.hpp>`

Public Member Functions

- **Timeseries** (std::vector< T > data, double **timestep**=0, double **initial_time**=0, std::string **comment**="")
- **Timeseries** (const char *filename, size_t column=1, size_t offset=0)
- void **read** (const char *filename, size_t column=1, size_t offset=0)
- void **read_all** (const char *filename, size_t offset=0)
- void **read** (std::ifstream &stream, size_t column=1, size_t offset=0)
- void **read_all** (std::ifstream &stream, size_t offset=0)
- size_t **size** () const
- double **timestep** () const
- std::string **comment** () const
- double **initial_time** () const
- void **push_back** (T val)
- void **emplace_back** (T &&val)
- void **set_timestep** (double **timestep**)
- void **set_comment** (std::string **comment**)
- void **set_initial_time** (double **initial_time**)
- double **time** (const int step) const
- std::vector< T >::const_iterator **begin** () const
- std::vector< T >::const_iterator **end** () const
- T **mean** () const
- T **stdev** () const
- double **autocorrelation_function** (size_t span) const

- **Timeseries** & **operator+=** (const **Timeseries** &ts)
- **Timeseries** **operator+** (const **Timeseries** &ts) const
- **Timeseries** & **operator-=** (const **Timeseries** &ts)
- **Timeseries** **operator-** (const **Timeseries** &ts) const
- **Timeseries** & **operator*=** (const double a)
- **Timeseries** **operator*** (const double a)
- **Timeseries** & **operator/=** (const double a)
- **Timeseries** **operator/** (const double a)
- bool **operator==** (const **Timeseries** &ts) const
- bool **operator!=** (const **Timeseries** &ts) const
- void **coarsen** (const size_t factor)
- void **write** (const char *filename) const
- void **clear** ()
- T **operator[ ]** (size_t i) const
- T const & **operator[ ]** (size_t i)

Friends

- std::ostream & **operator<<** (std::ostream &os, const **Timeseries** &timeseries)

4.3.1 Detailed Description

template<typename T>
class Timeseries< T >

The **Timeseries** (p. 26) class stores a time series of data points and contains some functions for their analysis. A data point can be a vector of points as well as the type is templated. A time step, initial time and a comment can be given.

4.3.2 Member Function Documentation

4.3.2.1 begin() `template<typename T >`

`std::vector<T>::const_iterator` **Timeseries**`< T >::begin ( ) const [inline]`

Returns

Constant reference to first data piont in form of an iterator.

4.3.2.2 clear() `template<typename T >`

`void` **Timeseries**`< T >::clear ( ) [inline]`

Clear all data from **Timeseries** (p. 26).

4.3.2.3 comment() template<typename T >

```
std::string  Timeseries< T >::comment ( ) const  [inline]
```

Returns

The comment that is set.

4.3.2.4 end() template<typename T >

```
std::vector<T>::const_iterator  Timeseries< T >::end ( ) const  [inline]
```

Returns

Constant reference to last data piont in form of an iterator.

4.3.2.5 initial_time() template<typename T >

```
double  Timeseries< T >::initial_time ( ) const  [inline]
```

Returns

The initial time that is set.

4.3.2.6 operator"!=() template<typename T >

```
bool  Timeseries< T >::operator!= (
            const  Timeseries< T > & ts ) const  [inline]
```

Test whether the data points of two **Timeseries** (p. 26) are not equal by value.

4.3.2.7 operator*() template<typename T >

```
Timeseries  Timeseries< T >::operator* (
            const double a )  [inline]
```

Multiply two **Timeseries** (p. 26) element wise by a scalar.

4.3.2.8 operator*=() template<typename T >

```
Timeseries&  Timeseries< T >::operator*= (
            const double a )  [inline]
```

Multiply current **Timeseries** (p. 26) element wise by a scalar.

4.3.2.9 operator+() template<typename T >

```
Timeseries Timeseries< T >::operator+ (
            const Timeseries< T > & ts ) const [inline]
```

Add two **Timeseries** (p. 26) element wise.

4.3.2.10 operator+=() template<typename T >

```
Timeseries& Timeseries< T >::operator+= (
            const Timeseries< T > & ts ) [inline]
```

Add a second timeseries to the current element wise.

4.3.2.11 operator-() template<typename T >

```
Timeseries Timeseries< T >::operator- (
            const Timeseries< T > & ts ) const [inline]
```

Substract a second **Timeseries** (p. 26) from the current element wise.

4.3.2.12 operator-=() template<typename T >

```
Timeseries& Timeseries< T >::operator-= (
            const Timeseries< T > & ts ) [inline]
```

Substract two **Timeseries** (p. 26) element wise.

4.3.2.13 operator/() template<typename T >

```
Timeseries Timeseries< T >::operator/ (
            const double a ) [inline]
```

Divide two **Timeseries** (p. 26) element wise by a scalar.

4.3.2.14 operator/=() template<typename T >

```
Timeseries& Timeseries< T >::operator/= (
            const double a ) [inline]
```

Divide current **Timeseries** (p. 26) element wise by a scalar.

4.3.2.15 operator==() template<typename T >

```
bool Timeseries< T >::operator== (
            const Timeseries< T > & ts ) const [inline]
```

Test whether the data points of two **Timeseries** (p. 26) are equal by value.

4.3.2.16 push_back() template<typename T >

```
void Timeseries< T >::push_back (
            T val ) [inline]
```

Returns

Append new value to the end of the **Timeseries** (p. 26).

4.3.2.17 set_initial_time() template<typename T >

```
void  Timeseries< T >::set_initial_time (
            double initial_time )  [inline]
```

Set the initial time.

4.3.2.18 set_timestep() template<typename T >

```
void  Timeseries< T >::set_timestep (
            double timestep )  [inline]
```

Set the timestep.

4.3.2.19 size() template<typename T >

```
size_t  Timeseries< T >::size ( ) const  [inline]
```

Returns

The total number of data points.

4.3.2.20 timestep() template<typename T >

```
double  Timeseries< T >::timestep ( ) const  [inline]
```

Returns

The timestep that is set.

4.3.2.21 write() template<typename T >

```
void  Timeseries< T >::write (
            const char * filename ) const  [inline]
```

Write **Timeseries** (p. 26) to file.

4.3.3 Friends And Related Function Documentation

4.3.3.1 operator<< template<typename T >

```
std::ostream& operator<< (
            std::ostream & os,
            const  Timeseries< T > & timeseries )  [friend]
```

Print info.

The documentation for this class was generated from the following file:

- lib/ **timeseries.hpp**

4.4 Trajectory Class Reference

`#include <trajectory.hpp>`

Public Member Functions

- **Trajectory** (const **Trajectory** &)=delete
- **Trajectory** (const std::string &filename, size_t **particles_per_molecule**=0, size_t offset=0)
- bool **is_null** () const
- bool **is_good** () const
- bool **clear_ahead** ()
- size_t **index** () const
- size_t **frames_read** () const
- const **Frame** & **frame** () const
- double **timestep** () const
- size_t **particles_per_molecule** () const
- size_t **number_of_molecules** () const
- size_t **size** ()
- void **reset** ()
- void **set_frame** (**Frame** **frame**)
- std::streampos **tellg** ()
- void **advance** (const size_t number=1)
- void **go_to_last_frame** ()
- void **loop_advance** (int argc, char **argv)
- void **move_to** (const size_t number)
- void **move_to** (const std::streampos pos)
- **Trajectory** & **operator=** (const **Trajectory** &)=delete
- **Frame** **operator*** () const
- **Frame** * **operator->** ()
- **Frame operator[ ]** (size_t **index**)
- **Trajectory** & **operator++** ()
- **Trajectory** & **operator+=** (const size_t number)
- template<typename T >
  double **mean** (T(Molecule::*f)(void), size_t step=1)
- template<typename T >
  Timeseries< T > **timeseries** (T(Frame::*f)(void), size_t step=1, size_t max=std::numeric_limits< int >::max())
- template<typename T >
  Timeseries< std::vector< T > > **timeseries** (T(Molecule::*f)(void), size_t step=1, size_t max=std::numeric_limits< int >::max())
- std::vector< **Timeseries**< std::vector< Real3D > > > **timeseries_set** (std::vector< Real3D(Molecule::*)()> vofp={ & **Molecule::end_to_end**, & **Molecule::center_of_mass** }, size_t step=1, size_t max=std::numeric_limits< int >::max())
- template<typename T >
  Timeseries< T > **timeseries_mean** (T(Molecule::*f)(void), size_t step=1, size_t max=std::numeric_limits< int >::max())
- template<typename T >
  Timeseries< T > **timeseries_single_molecule** (T(Molecule::*f)(void), size_t i_mol=0, size_t step=1, size_t max=std::numeric_limits< int >::max())
- void **write_xyz** (const char *filename, const bool append=true, const size_t precision=11)

Friends

- std::ostream & **operator<<** (std::ostream &os, const **Trajectory** &trajectory)

4.4.1 Detailed Description

The **Trajectory** (p. 31) class is a reader class to read trajectory files given by concatenated xyz configurations. As the simulations often produce huge files a focus is put on efficiency. Supported formats: .xyz The class can be used like a forward iterator. Dereferencing returns the current **Frame** (p. 3) object.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Trajectory() [1/2] Trajectory::Trajectory (const **Trajectory** &) [delete]

Disable copy constructor.

4.4.2.2 Trajectory() [2/2] Trajectory::Trajectory (const std::string & *filename*, size_t *particles_per_molecule* = *0*, size_t *offset* = *0*) [inline]

Construct **Trajectory** (p. 31) reader on given file.

Parameters

in	*filename*	Input file name.
in	*particles_per_molecule*	Number of particles in one molecule.
in	*offset*	Number of Frames to be skipped in the beginning.

4.4.3 Member Function Documentation

4.4.3.1 advance() void Trajectory::advance (const size_t *number* = *1*) [inline]

Jump ahead a given number of Frames.

4.4.3.2 frame() const **Frame**& Trajectory::frame ( ) const [inline]

Returns

Current frame on top of the **Trajectory** (p. 31).

4.4.3.3 frames_read() `size_t Trajectory::frames_read ( ) const [inline]`

Returns

The number of frames that have already been read.

4.4.3.4 go_to_last_frame() `void Trajectory::go_to_last_frame ( ) [inline]`

Go to last Frames.

4.4.3.5 index() `size_t Trajectory::index ( ) const [inline]`

Returns

Current frame index.

4.4.3.6 is_good() `bool Trajectory::is_good ( ) const [inline]`

Check whether trajectory is readable.

4.4.3.7 is_null() `bool Trajectory::is_null ( ) const [inline]`

Check whether trajectory is unreadable.

4.4.3.8 loop_advance()

```
void Trajectory::loop_advance (
        int argc,
        char ** argv ) [inline]
```

Advance the **Trajectory** (p. 31) in a way that is given by command line arguments. –offset gives the number of Frames to skip in the beginning. –max gives the highest **Frame** (p. 3) index to be considdered. –step gives the step size. I.e. for a step of 2 only every second **Frame** (p. 3) is read. Alternatively the step size can be increased exponentially with the –exp argument.

Parameters

in	*argc*	Length of argument array.
in	*argv*	Command line argument array.

4.4.3.9 move_to()

```
void Trajectory::move_to (
        const size_t number ) [inline]
```

Move to frame with given inded.

4.4.3.10 number_of_molecules() size_t Trajectory::number_of_molecules ( ) const [inline]

Returns

Number of molecules in the system.

4.4.3.11 operator*() Frame Trajectory::operator* ( ) const [inline]

Dereferencing the **Trajectory** (p. 31) returns the frame currently on top.

4.4.3.12 operator++() Trajectory& Trajectory::operator++ ( ) [inline]

Advance by one frame.

4.4.3.13 operator=() Trajectory& Trajectory::operator= (const Trajectory &) [delete]

Disable assign operator.

4.4.3.14 particles_per_molecule() size_t Trajectory::particles_per_molecule ( ) const [inline]

Returns

Molecule (p. 22) size.

4.4.3.15 reset() void Trajectory::reset ( ) [inline]

Clear the **Trajectory** (p. 31).

4.4.3.16 size() size_t Trajectory::size ( ) [inline]

Returns

Total number of Frames. Warning: The complete **Trajectory** (p. 31) must be iterated to find this number. Depending on the size this might take a while.

4.4.3.17 timeseries() template<typename T >

```
Timeseries<T> Trajectory::timeseries (
            T(Frame::*)(void) f,
            size_t step = 1,
            size_t max = std::numeric_limits<int>::max() ) [inline]
```

Generate **Timeseries** (p. 26) object where the data points are calculated from the following frames with function f.

Parameters

in	*f*	Function of **Frame** (p. 3).
in	*step*	Number of frames to advance each step.
	[max]	max Highest index to consider.

4.4.3.18 timestep() `double Trajectory::timestep ( ) const [inline]`

Returns

Time step.

4.4.4 Friends And Related Function Documentation

4.4.4.1 operator<< `std::ostream& operator<< ( std::ostream & os, const Trajectory & trajectory ) [friend]`

Print info.

The documentation for this class was generated from the following file:

- lib/ **trajectory.hpp**

5 File Documentation

5.1 lib/frame.hpp File Reference

```
#include "molecule.hpp"
#include "fftw3.h"
#include <typeinfo>
#include <memory>
#include <random>
#include <set>
#include <iomanip>
```

Classes

- class **Frame**

Functions

- double **read_lattice** (const **Frame** &frame, double *lattice, const size_t side_length, Real3D *velocity_↩ lattice=nullptr, const size_t dim=3)
- double **read_lattice** (const char *filename, double *lattice, const size_t side_length, size_t dim=3)
- void **write_lattice** (const **Frame** &f, const char *filename, const size_t side_length)
- std::vector< std::array< double, 2 > > **linearize_lattice** (const fftw_complex *const lattice_transformed, const size_t side_length, const double lattice_constant, const double bin_width, const double norm=1., const size_t dim=3)
- template<typename T > std::vector< std::array< double, 2 > > **structure_factor** (const T input, const size_t side_length, const double lattice_constant, const double bin_width=0.1, const double norm=1., const size_t dim=3)
- template<> std::vector< std::array< double, 2 > > **structure_factor**< **double** * > (double *lattice, const size_t side_length, const double lattice_constant, const double bin_width, const double norm, const size_t dim)
- double **structure_factor** (const **Frame** &frame, const Real3D q)
- std::vector< Real3D > **lattice_vectors_inside_shell** (const double radius, const double thickness, const double lattice_constant)
- std::vector< double > **mean_structure_factor** (const **Frame** &frame, const double q, const size_t n_↩ rand=128)
- template<typename T > std::array< double, 6 > **minkowski_functionals** (const T input, const size_t side_length, const double threshold=-1., const char norm='n', const bool natural_units=false)
- template<> std::array< double, 6 > **minkowski_functionals**< **double** * > (double *const lattice, const size_t side_length, const double threshold, const char norm, const bool natural_units)

5.1.1 Function Documentation

5.1.1.1 lattice_vectors_inside_shell()

```
std::vector<Real3D> lattice_vectors_inside_shell (
        const double radius,
        const double thickness,
        const double lattice_constant )
```

Compute the set of lattice vectors that lie within a spherical shell of certain thickness.

Parameters

in	*radius*	Radius of shell.
in	*thickness*	Thickness of shell.
in	*lattice_constant*	Lattice constant.

Returns

List of vectors.

5.1.1.2 linearize_lattice()

```
std::vector<std::array<double, 2> > linearize_lattice (
        const fftw_complex *const lattice_transformed,
```

```
            const size_t side_length,
            const double lattice_constant,
            const double bin_width,
            const double norm = 1.,
            const size_t dim = 3 )
```

Calculate the Structure factor S(q) from the Fourier transform of the lattice density. This gives the structure factor depending on the VECTOR q. From this a histogram is produced where each bin contains the average structure factor for a certain range of absolute values of q.

Parameters

in	*lattice_transformed*	Fourier transformation of a density lattice.
in	*side_length*	Number of lattice sites in one direction.
in	*lattice_constant*	Lattice constant.
in	*bin_width*	bin width of the q values

Returns

vector of pairs (q, S(q)).

5.1.1.3 mean_structure_factor()

```
std::vector<double> mean_structure_factor (
            const  Frame & frame,
            const double q,
            const size_t n_rand = 128 )
```

Method for the brute-force calculation of the structure factor S(q). All lattice vectors in a shell of thickness lattice↩ _constant around q are considered if there are less than n_rand. If there are more, n_rand vectors are chosen at random.

Parameters

in	*frame*	Input **Frame** (p. 3).
in	*q*	Absolute value of vave vector to consider.
in	*n_rand*	Number of random orientations to consider.

Returns

Vector of tuples (<q>, <S(q)>, \sigma(q)).

5.1.1.4 minkowski_functionals()

```
template<typename T >
std::array<double, 6> minkowski_functionals (
            const T input,
            const size_t side_length,
            const double threshold = -1.,
            const char norm = 'n',
            const bool natural_units = false )
```

Calculate Minkowski functionals (MFs)in 3 dimensions: V_0: volume V_1: area V_2: mean curvature (4n) V_3: mean curvature (8n) V_4: Euler-Poincare characteristic (6n) V_5: Euler-Poincare characteristic 26n) The configuration is first mapped onto a black and white lattice. Each cell-center of the lattice has 8 neighbors. Hence there are 2^8 different neighborhoods. The MFs are additive and rotationally invariant. By symmetry the number of neighborhoods that are unique wrt. the MFs reduces to 22. The MFs are computed in the lattice centers.

See paper Arns, Knackstedt, Pinczewski, Mecke Phys. Rev. E 63 2001

Parameters

in	*input*	Input lattice configuration either as **Frame** (p. 3) or filename const char *
in	*side_length*	Linear lattice size of the interpolation lattice.
in	*threshold*	Lattice sites with density >= threshold will be interpreted as 'black'.
in	*natural_units*	Normalize results by appropriate power of side_length in order to make it dimensionless.

Returns

std::array of the 6 Minkowski functionals

5.1.1.5 read_lattice() [1/2]

```
double read_lattice (
        const char * filename,
        double * lattice,
        const size_t side_length,
        size_t dim = 3 )
```

Read a lattice from file. Lattice must be cubic.

Parameters

in	*filename*	Input file name.
out	*lattice*	Array that stores the lattice.
in	*side_length*	Linear size of the lattice.

Returns

Total lattice density which is equal to the particle number.

5.1.1.6 read_lattice() [2/2]

```
double read_lattice (
        const  Frame & frame,
        double * lattice,
        const size_t side_length,
        Real3D * velocity_lattice = nullptr,
        const size_t dim = 3 )
```

Projects a configuration onto a lattice using second-order extrapolation. The closer a particle is to a lattice site the higher its contribution to the lattice site's density. The box must be cubic. Repeated application adds the new configuration on top of the old.

Parameters

in	*frame*	Input configuration.
out	*lattice*	Array that stores the lattice.
in	*side_length*	Number of lattice sites in one direction.
out	*velocity_lattice*	Optional array that stores the velocity lattice.
in	*Dimension*	of the configuration (either 2 or 3).

5.1.1.7 structure_factor()

```
double structure_factor (
        const  Frame & frame,
        const Real3D q )
```

Calculate the exact structure factor for given wave vector q

Parameters

in	*frame*	Input **Frame** (p. 3).
in	*q*	wave vector.

Returns

S(q)

5.1.1.8 write_lattice()

```
void write_lattice (
        const  Frame & f,
        const char * filename,
        const size_t side_length )
```

Write lattice configuration to disk.

Parameters

in	*f*	**Frame** (p. 3) to get the lattice from.
in	*filename*	Output filename.
in	*side_length*	Number of lattice sites in one direction.

5.2 lib/molecule.hpp File Reference

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <cstdlib>
#include <vector>
```

```
#include <string>
#include <stdexcept>
#include <algorithm>
#include <omp.h>
#include <assert.h>
#include <random>
#include "timeseries.hpp"
#include "Real3D.hpp"
#include "couf.hpp"
```

Classes

- class **Molecule**

5.2.1 Detailed Description

The **Molecule** (p. 22) class allows for the modelling of chain molecules. For a **Molecule** (p. 22) object only the reference to the first and last bead of the molecule is stored in form of iterators. It is based on a full vector of coordinates (and optionally a vector of velocities) that may correspond to multiple molecules and is stored at another place (typically in a **Frame** (p. 3) object).

5.3 lib/timeseries.hpp File Reference

```
#include "couf.hpp"
#include "vector.hpp"
#include <algorithm>
#include <sstream>
#include <iomanip>
```

Classes

- class **Timeseries**< **T** >

5.4 lib/trajectory.hpp File Reference

```
#include "frame.hpp"
#include <list>
```

Classes

- class **Trajectory**

Functions

- template<typename T >
  double **correlation_function** (T(Molecule::*f)(void), **Trajectory** &trajectory, const size_t span)

Index