



UvA-DARE (Digital Academic Repository)

Bridging time and length scales in dissipative particle dynamics

Backer, J.A.

Publication date

2006

Document Version

Final published version

[Link to publication](#)

Citation for published version (APA):

Backer, J. A. (2006). *Bridging time and length scales in dissipative particle dynamics*. [Thesis, fully internal, Universiteit van Amsterdam].

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

**Bridging time and length scales
in
Dissipative Particle Dynamics**

Jantien Backer

**Bridging time and length scales
in
Dissipative Particle Dynamics**

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
Prof. mr. P. F. van der Heijden
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in de Aula der Universiteit

op dinsdag 9 mei 2006, te 12.00 uur

door

Jantien Annet Backer

geboren te Amsterdam

Promotiecommissie:

Promotor:

- Prof. dr. P. D. Iedema

Copromotor:

- Dr. ir. H. C. J. Hoefsloot

Overige leden:

- Prof. dr. W. J. Briels
- Prof. dr. ir. B. Smit
- Prof. dr. P. M. A. Sloot
- Dr. C. P. Lowe
- Dr. ir. S. M. Willemsen

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research reported in this thesis was carried out at the Van 't Hoff Institute for Molecular Sciences, Faculty of Science, University of Amsterdam (Nieuwe Achtergracht 166, 1018 WV, Amsterdam, The Netherlands).

Contents

1	Introduction	1
1.1	Length and time scales in simulations	1
1.2	Scope of this thesis	3
2	Dissipative Particle Dynamics	5
2.1	Introduction	5
2.2	Pair potential	5
2.3	Thermostat	7
2.4	Integration algorithm	8
2.5	Boundary conditions	10
2.6	DPD extensions	11
2.7	Applications of DPD	12
3	Computational efficiency of Dissipative Particle Dynamics	15
3.1	Introduction	15
3.2	Theory of force calculation	16
3.2.1	Implementations	17
3.2.2	Time constants	19
3.2.3	Update frequency	20
3.2.4	Radius of Verlet list	21
3.3	Computational details	22
3.4	Results and discussion	23
3.4.1	Optimization of Verlet radius	23
3.4.2	Computation time as a function of temperature	24
3.4.3	Implementation diagrams	28
3.5	Conclusion	28
4	Viscosity measurement in Dissipative Particle Dynamics	31
4.1	Introduction	31
4.2	Periodic Poiseuille flow method	33
4.3	Computational details	35
4.4	Results and discussion	37
4.4.1	Validation of the periodic Poiseuille flow method	37
4.4.2	Comparison of viscosity measurement techniques in DPD	41

4.4.3	Comparison of viscosity measurement techniques for a Lennard-Jones fluid	43
4.5	Conclusion	44
5	Coarse-graining Dissipative Particle Dynamics	45
5.1	Introduction	45
5.2	Coarse-graining procedure for DPD	46
5.2.1	Mass density	47
5.2.2	Number of interactions per particle	47
5.2.3	Pressure	48
5.2.4	Shear viscosity	50
5.2.5	Temperature	51
5.3	Computational details	51
5.4	Results and discussion	52
5.4.1	Pressure of the coarse-grained system	52
5.4.2	Viscosity of the coarse-grained system	52
5.4.3	Performance	57
5.5	Conclusion	58
6	Combining length scales in Dissipative Particle Dynamics	61
6.1	Introduction	61
6.2	Combination of length scales in one system	62
6.2.1	Local refining	63
6.2.2	Local coarse-graining	64
6.2.3	Size of effective overlap region	65
6.3	Computational details	66
6.4	Results and discussion	67
6.4.1	Combined system in equilibrium	67
6.4.2	Combined system under flow conditions	69
6.4.3	Performance	72
6.5	Conclusion	73
7	Wormlike chain model	75
7.1	Introduction	75
7.2	Wormlike chain model	76
7.2.1	Stretching force	76
7.2.2	Bending force	77
7.3	Computational details	79
7.4	Results and discussion	81
7.4.1	Wormlike chain of two beads	81

7.4.2	Wormlike chain of N beads	83
7.5	Conclusion	85
8	Multiple time step algorithm for explicit solvent simulations	87
8.1	Introduction	87
8.2	Multiple time step model	88
8.2.1	Multiple time step algorithm	88
8.2.2	Computational efficiency	91
8.3	Computational details	93
8.3.1	Simulations without polymer	93
8.3.2	Simulations with polymer	94
8.4	Results and discussion	97
8.4.1	Validation multiple time step algorithm	97
8.4.2	Wormlike chain in solution	100
8.4.3	Performance	104
8.5	Conclusion	107
	Bibliography	109
	Summary	117
	Samenvatting	121
	Dankwoord	125

Introduction

1.1 Length and time scales in simulations

All processes in nature have characteristic time and length scales. Cosmological behaviour is expressed in billions of years and light years, while molecular motion is measured in picoseconds and Ångströms. For the simulation of such processes, and all processes in between, a variety of models is available, each with its specific time and length scale range. As an example, we will consider a (small) polymer, C_9H_{20} , in figure 1.1. When we are interested in the reactivity of the carbon-hydrogen bonds, we need quantum mechanics to determine the electronic structure around the atoms. On a higher level, we might only want to have information about the conformation of the polymer. Then we can neglect the electrons and hydrogen atoms. Instead, we use a united atom model with an effective potential that mimics the Van der Waals interaction between atoms. When we would like to know how the polymer behaves in a solvent, the exact conformation is not of interest. A number of atoms (three in figure 1.1) are clustered into larger particles that interact through a soft potential. When the individual polymers are not the subject of interest, we can abandon the particle models. Instead, we use continuum equations to calculate for instance the concentration of polymers.

With four different types of models, one can examine the same polymer at different scales. Depending on the choice, processes on the quantum, atomistic, mesoscale or continuum level are studied. The respective temporal and spatial scales of these simulation techniques, are schematically represented in figure 1.2. As the overlaps indicate, time and length scales can be accessible

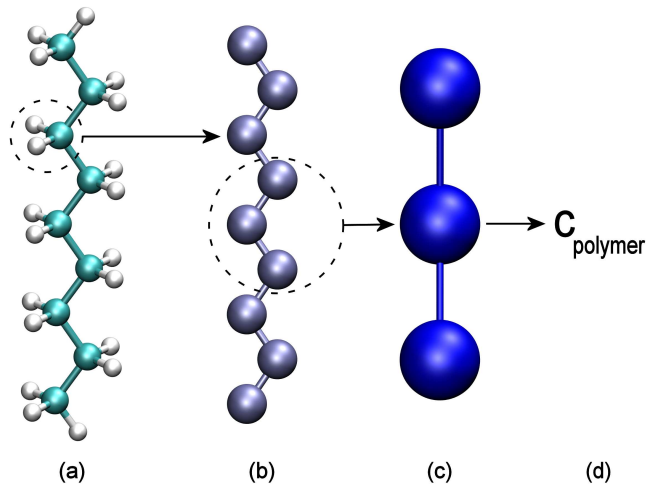


Figure 1.1: Polymer C_9H_{20} modelled by (a) atoms, (b) united atoms, (c) soft particles and (d) concentration.

by different techniques. The choice depends on the process to be modelled.

When the difference between time and length scales of two processes is large, neither process will be influenced by the other. The electronic structure around the atoms will not significantly affect the flow field in the reactor, or the other way around. However, when differences are smaller, they can influence each other. For instance, when the polymer is surrounded by a “bad” solvent, it will collapse to a spheroid, which will affect the flow field. Conversely, the flow field can apply shear stress on the spheroid, which will change its shape. In these cases, the separate processes need to be simulated simultaneously to include the mutual effects. Other examples of such systems containing disparate scales, are rupture of cell membranes and protein folding.

The simplest way to simulate processes of disparate scales, is to resort to a small scale model and simulate large systems for a long time to examine the larger scale process, but this is often computationally challenging. An alternative is to couple two differently scaled models in a hybrid scheme. The output of one model serves as the input for the other, and vice versa. A combination of both approaches is to couple the different scales within the framework of one model, using refinement techniques. In this thesis, we will examine these kind of techniques in a mesoscale model. Applying refinement should make the

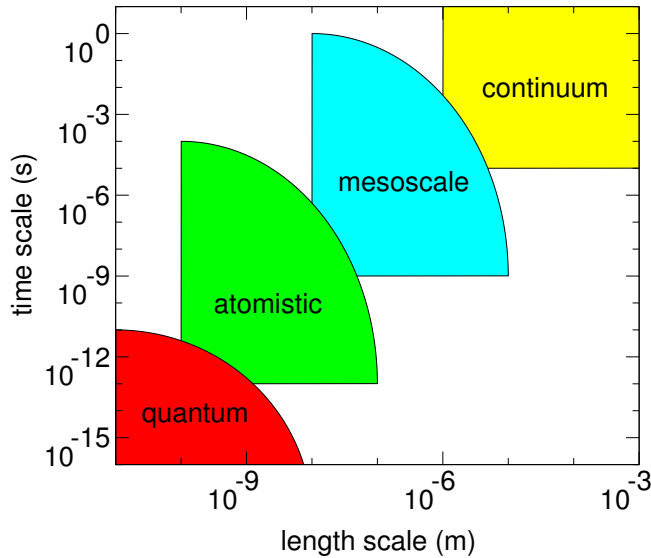


Figure 1.2: Schematic representation of characteristic time and length scales of simulation techniques.

model computationally more efficient, which allows for larger time and length scales to simulate.

1.2 Scope of this thesis

The modelling technique we will use, is Dissipative Particle Dynamics (DPD). Two reasons make DPD suitable for our purposes. In the first place, its characteristic scale is the mesoscale. At this level many complex phenomena occur that are a combination of disparately scaled processes. Secondly, DPD has been introduced only fourteen years ago, so the theoretical aspects of the model are still subject of ongoing research. **Chapter 2** introduces the model equations and mentions improvements that were proposed over the years. Extensions and derived models are discussed, as well as applications of DPD in various complex systems. Next, the computational efficiency of the model is examined in **chapter 3**. It will become clear that this depends on how the calculation of the pairwise forces is implemented. We derive a theoretical method to identify the best implementation, based on the input parameters. As we will only focus on the flow dynamics of isothermal systems, viscosity plays an important

role in our simulations. We devise a technique to measure the viscosity - the periodic Poiseuille flow method - in **chapter 4**, that is more efficient than the commonly used techniques. We will use this method throughout this thesis. Next, a refinement technique for the length scales in DPD is constructed. First, we derive a coarse-grained description of the DPD model in **chapter 5**, that models the same system as the original DPD model, only with less particles. Mass density, pressure, viscosity and temperature are identical in both descriptions. In **chapter 6** they are combined in one system to enable simultaneous simulation. The coupling scheme allows for mass and momentum transfer between the two different scales. Finally, we would like to construct a similar scheme for the time scales and apply this to a relevant problem. For this purpose a wormlike chain model in solution is very suitable, as the polymer must be simulated with a smaller time step compared to the solvent particles. In **chapter 7** we will first describe the wormlike chain model, and examine its static properties. The next step is to devise a multiple time step algorithm, that combines the different time scales in one system. However, this proves to be impossible in DPD, because the time step is intrinsically intertwined with the thermostat equations. Instead we use the Lowe-Andersen thermostat to construct the multiple time step algorithm in **chapter 8**. We apply it to the wormlike chain model in solution to examine the dynamic properties.

Dissipative Particle Dynamics

2.1 Introduction

In 1992 Hoogerbrugge and Koelman introduced a new model that they called Dissipative Particle Dynamics (DPD).¹ After a slow start, the model gained momentum when in 1995 Español and Warren formulated the fluctuation-dissipation theorem for DPD² that ensures the proper thermodynamic equilibrium. The result is a mesoscopic particle model that obeys the Navier-Stokes equations,³ that is isotropic and Galilean invariant and that has the correct Maxwell-Boltzmann distribution. The transport coefficients³⁻⁷ as well as the dynamic behaviour^{8,9} of DPD have been extensively studied over the years. In this section we give an overview of the DPD model. First we will describe the model itself, that is simply a collection of point masses, each of which behaves according to Newton's laws of motion. The description focuses on the pair potential, the thermostat, the integration algorithm and the boundary conditions. When appropriate, improvements and alternatives are mentioned. We conclude with some extensions of the model that have been proposed, as well as an overview of the various fields in which DPD is applied.

2.2 Pair potential

Mesoscopic particle models differ from more molecular descriptions by their pair potential. The latter often use the Lennard-Jones potential u_{LJ} to mimic molecular interactions:¹⁰

$$\mathbf{u}_{\text{LJ}}(\mathbf{r}) = 4\epsilon_{\text{LJ}} \left[\left(\frac{\sigma_{\text{LJ}}}{r} \right)^{12} - \left(\frac{\sigma_{\text{LJ}}}{r} \right)^6 \right], \quad (2.1)$$

where r is the particle-particle distance, ϵ_{LJ} is the depth of the attractive well and σ_{LJ} is the distance where the potential changes from attractive to repulsive. This potential approaches infinity at small particle separations, which drastically limits the permissible time step. Mesoscopic potentials on the other hand are often soft repulsive, that allow for larger time steps. This is in no way specific to DPD, but it is sometimes referred to as the ‘‘DPD-potential’’, \mathbf{u}_{soft} :

$$\begin{aligned} \mathbf{u}_{\text{soft}}(\mathbf{r}) &= \frac{1}{2} \mathbf{a} r_c \left(1 - \frac{r}{r_c} \right)^2 & \text{if } r \leq r_c \\ &= 0 & \text{if } r > r_c, \end{aligned} \quad (2.2)$$

where \mathbf{a} is the phase repulsion and r_c the cut-off radius, beyond which the potential is zero. When the particle separation r is zero, the potential is maximal but finite. It means that there is a small but non-zero chance that the point masses coincide. This is comprehensible when a particle is regarded as a fluid element that can move straight through another. The derivative of the potential with respect to the particle separation r_{ij} gives the conservative force \vec{f}_{ij}^{C} , that particles i and j exert on each other:

$$\begin{aligned} \vec{f}_{ij}^{\text{C}} &= \mathbf{a}_{ij} \left(1 - \frac{r_{ij}}{r_c} \right) \hat{\mathbf{r}}_{ij} & \text{if } r_{ij} \leq r_c \\ &= 0 & \text{if } r_{ij} > r_c. \end{aligned} \quad (2.3)$$

Here the distance between the particles is $r_{ij} = |\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j|$. This force, as well as the other forces, works along the unit vector $\hat{\mathbf{r}}_{ij} = (\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j)/r_{ij}$ between the particles. The phase repulsion is positive as the potential is purely repulsive, and it is symmetric, $\mathbf{a}_{ij} = \mathbf{a}_{ji}$, to ensure momentum conservation. The choice of the parameter depends on the fluid to model. When $\mathbf{a} = 0$, one models the so-called ‘‘ideal DPD fluid’’. Based on the compressibility of water, Groot and Warren proposed the relation $\mathbf{a}n \approx 75k_{\text{B}}T$,¹¹ with n the number density and $k_{\text{B}}T$ the thermal energy. However, this is only valid when one DPD particle contains one water molecule. Groot and Rabone corrected this for the

case where one DPD particle contains three water molecules.¹² By applying a different phase repulsion between unlike particle types, one can induce diverse phase behaviour. For example, Kranenburg *et al.* examined the phase repulsion sets used for phospholipid-water models.¹³

Pagonabarraga and Frenkel took a more fundamental approach to the phase repulsion. They formulated a density functional theory for DPD, or many-body DPD (MDPD), where the potential depends on the local density.^{14,15} Trofimov *et al.* added a correction for particle correlations in strongly nonideal systems¹⁶ and Warren applied MDPD to model vapour-liquid coexistence.¹⁷

2.3 Thermostat

The most important feature of DPD is its thermostat. It consists of two counteracting forces, the dissipative and random force. The dissipative force \vec{f}_{ij}^D between particle i and j describes the friction between the particles:

$$\begin{aligned} \vec{f}_{ij}^D &= -\gamma w^D(r_{ij}) (\hat{r}_{ij} \cdot \vec{v}_{ij}) \hat{r}_{ij} & \text{if } r_{ij} \leq r_c \\ &= 0 & \text{if } r_{ij} > r_c . \end{aligned} \quad (2.4)$$

This force depends linearly on the friction coefficient γ and the projection of the velocity difference $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$ on the interparticle vector \hat{r}_{ij} . The weight function $w^D(r_{ij})$ depends on the interparticle distance. While the dissipative force drains energy from the system, the random force \vec{f}_{ij}^R provides the system with energy:

$$\begin{aligned} \vec{f}_{ij}^R &= \sigma w^R(r_{ij}) \theta_{ij} \hat{r}_{ij} & \text{if } r_{ij} \leq r_c \\ &= 0 & \text{if } r_{ij} > r_c . \end{aligned} \quad (2.5)$$

Here σ is the noise amplitude, $w^R(r_{ij})$ the weight function for the random force and θ_{ij} a random number with zero mean and Δt^{-1} variance, where Δt is the time step. Español and Warren formulated the fluctuation-dissipation theorem² to balance the energy provided by the random force with the energy

drained by the dissipative force. The system will equilibrate at a thermal energy of $k_B T$ (where k_B is Boltzmann's constant and T the system temperature), when the following relations are fulfilled:

$$\begin{aligned}\sigma^2 &= 2\gamma k_B T, \\ w^D(\mathbf{r}_{ij}) &= [w^R(\mathbf{r}_{ij})]^2.\end{aligned}\tag{2.6}$$

The weight function for the random force $w^R(\mathbf{r}_{ij})$ is usually chosen as a function that decreases linearly till zero at the cut-off radius:

$$\begin{aligned}w^R(\mathbf{r}_{ij}) &= 1 - \frac{r_{ij}}{r_c} & \text{if } r_{ij} \leq r_c \\ &= 0 & \text{if } r_{ij} > r_c,\end{aligned}\tag{2.7}$$

from which the weight function for the dissipative force $w^D(\mathbf{r}_{ij})$ directly follows (eq. 2.6). Another thermostat that also works on particle pairs, is the Lowe-Andersen thermostat.¹⁸ With a probability of $\Gamma \Delta t$ (where Γ is a constant bath collision parameter), a new relative velocity $[\vec{v}_{ij}]' \cdot \hat{\mathbf{r}}_{ij}$ is generated from a Gaussian distribution with variance $2k_B T/m$. The new velocities become:

$$\begin{aligned}\vec{v}'_i &= \vec{v}_i + \frac{1}{2} \hat{\mathbf{r}}_{ij} ([\vec{v}_{ij}]' - \vec{v}_{ij}) \cdot \hat{\mathbf{r}}_{ij}, \\ \vec{v}'_j &= \vec{v}_j - \frac{1}{2} \hat{\mathbf{r}}_{ij} ([\vec{v}_{ij}]' - \vec{v}_{ij}) \cdot \hat{\mathbf{r}}_{ij}.\end{aligned}\tag{2.8}$$

As the thermostat generates the new velocities directly from the correct Boltzmann distribution, the temperature is very rigorously controlled. We will use this thermostat, when the original DPD thermostat does not suffice. Recently, a linear combination of the Lowe-Andersen thermostat and the Nosé-Hoover thermostat,¹⁹ was shown to achieve a large range of Schmidt numbers.²⁰

2.4 Integration algorithm

The particle positions $\vec{\mathbf{r}}_i$ and velocities $\vec{\mathbf{v}}_i$ evolve according to Newton's laws of motion:

$$\begin{aligned}\dot{\vec{r}}_i &= \vec{v}_i \\ m\dot{\vec{v}}_i &= \vec{f}_i.\end{aligned}\tag{2.9}$$

The force \vec{f}_i is a summation of the pairwise conservative, dissipative, and random forces, that particle i experiences from its neighbouring particles:

$$\vec{f}_i = \sum_{i \neq j} \left(\vec{f}_{ij}^D + \vec{f}_{ij}^R + \vec{f}_{ij}^C \right).\tag{2.10}$$

For integrating the equations of motion with a discrete time step Δt , the velocity-Verlet algorithm¹⁰ is most commonly used in molecular simulations:

$$\begin{aligned}\vec{r}_i(t + \Delta t) &= \vec{r}_i(t) + \Delta t \vec{v}_i(t) + \frac{1}{2}(\Delta t)^2 \vec{f}_i(t), \\ \vec{f}_i(t + \Delta t) &= g(\vec{r}_i(t + \Delta t)), \\ \vec{v}_i(t + \Delta t) &= \vec{v}_i(t) + \frac{1}{2}\Delta t \left(\vec{f}_i(t) + \vec{f}_i(t + \Delta t) \right).\end{aligned}\tag{2.11}$$

The function g denotes the calculation of the force based on the current positions of the particles. However, in DPD the dissipative forces are also determined by the particle velocities, which in turn depend on the force. To overcome this dependency, Groot and Warren applied a modified velocity-Verlet scheme:¹¹

$$\begin{aligned}\vec{r}_i(t + \Delta t) &= \vec{r}_i(t) + \Delta t \vec{v}_i(t) + \frac{1}{2}(\Delta t)^2 \vec{f}_i(t), \\ \tilde{v}_i(t + \Delta t) &= \vec{v}_i(t) + \lambda \Delta t \vec{f}_i(t), \\ \vec{f}_i(t + \Delta t) &= g(\vec{r}_i(t + \Delta t), \tilde{v}_i(t + \Delta t)), \\ \vec{v}_i(t + \Delta t) &= \vec{v}_i(t) + \frac{1}{2}\Delta t \left(\vec{f}_i(t) + \vec{f}_i(t + \Delta t) \right).\end{aligned}\tag{2.12}$$

The velocity \tilde{v}_i serves as a guess for the velocity at $t + \Delta t$. Groot and Warren found an optimal value of $\lambda = 0.65$. However, this is not a self-consistent

procedure. As a result, this algorithm suffers from a deviating temperature, depending on the time step.^{21–24} A solution to this issue is the self-consistent algorithm of Pagonabarraga *et al.*, that introduces an iteration procedure to converge the velocity and force to consistent values.²² Based on analogies with stochastic dynamics, Otter and Clarke assumed the dissipative and random force are not constant over the entire time step.²⁵ Shardlow applied a splitting method,²⁶ which was recently combined with the Lowe-Andersen thermostat by Peters.²⁴ Several studies^{27–30} compared the different integration methods. In the most recent work by Nikunen *et al.*, the Shardlow scheme is found to be best for integrating the DPD equations.³⁰

2.5 Boundary conditions

The simplest way to restrict particles to the simulation domain is to use periodic boundary conditions.¹⁰ Lees and Edwards modified these conditions to introduce a constant shear in one direction. Their boundary conditions are still periodic, but alter the position and velocity of the periodic images.³¹ Many applications require a solid wall, *i.e.* an impenetrable boundary, that - at a high level of coarse-graining - imposes the correct wall velocity (“no-slip”), without ordering the fluid particles near the wall.

In early DPD work, solid boundaries were modelled by a high-density layer of frozen particles that all move at the speed of the wall (see for instance ref. 32, 33). To prevent particles from penetrating the wall, Revenga *et al.* examined different reflection methods to contain them.³⁴ Even so, the layer of frozen particles has a structuring effect on the fluid, when conservative forces are present. Moreover, the no-slip condition cannot be imposed by wall particles with a constant velocity, that are just outside the simulation domain. At the exact boundary between the wall and the fluid, the velocity will be some average between the wall and fluid velocity. To shift this average to the wall velocity, the density of the solid boundary was increased, which consequently aggravates the structuring effect. Diminishing the density fluctuations induced by frozen layers, is still a subject of ongoing research.^{35,36}

Willemsen *et al.* argued that the structuring effect and the no-slip boundary condition are two separate problems, that can be solved separately.³⁷ They achieved no-slip by continuing the velocity profile in the boundary, thus ensuring the exact wall velocity at the boundary between wall and fluid. At the

same time, equilibrated ghost particles provide the conservative force exerted by the wall on the fluid. Visser *et al.* improved this method by adding a mirrored twin system to account for the dissipative and conservative wall forces,³⁸ thus omitting the necessity of ghost particles.

All current methods still struggle with the incorporation of complex geometries, which is very important in technical applications (*e.g.* baffles in a stirred tank reactor). Also, when a fluid consists of more phases with different phase repulsions, the correct wall force has not yet been established.

2.6 DPD extensions

Since its introduction and first theoretical support, DPD has been further developed by various authors. One of the first extensions was energy conservation, that was introduced almost simultaneously by Bonet Avalos and Mackie³⁹ and by Español.⁴⁰ Both articles add a variable to store the internal energy of a particle. This enabled the study of thermal processes, such as heat conduction⁴¹ and phase change.⁴² In 1999 Español also added the volume of the particles as a model variable, to construct a thermodynamically consistent model.⁴³

We already mentioned the many-body DPD variant of Pagonabarraga and Frenkel,^{14,15} and its extension to vapour-liquid equilibria.¹⁷ The soft repulsive conservative force was replaced by an average depending on the local density. Also the thermostatting forces were critically adjusted. Cotter and Reich changed the white Gaussian noise term into a coloured equivalent, in what they named the extended DPD (EDPD) model.⁴⁴

DPD has been combined with other simulation techniques. Willemsen *et al.* were the first to combine DPD with Monte Carlo,⁴⁵ which was further developed for polymer-solvent mixtures by Wijmans *et al.*⁴⁶ However, they only use the soft repulsive force, which is not exclusively characteristic for the DPD model. Other hybrid DPD models include combinations with smoothed particle hydrodynamics (“smoothed DPD”⁴⁷) and Lorentz gases.⁴⁸

In 2003 Groot incorporated electrostatic interactions in his DPD simulations. His method couples all particles to a local electrostatic field and explicitly solves its field equations.⁴⁹ Novik and Coveney introduced different friction coefficients to simulate binary mixtures with different viscosities,⁵⁰ although their approach is slightly simplistic. More sophisticated choices for the different

friction coefficients are provided by Visser *et al.*,⁵¹ based on a thorough study of interfacial fluid behaviour. Recently, DPD has been extended for simulations at constant pressure.^{52,53}

2.7 Applications of DPD

Before we can apply DPD to any problem, we need to find the right parameters for the simulations. From a macroscopic view, one can consider dimensionless parameters, such as the Reynolds number, to match the simulations to real experiments (see for instance ref. 37). However, fixing one dimensionless parameter often unsettles another, so the dimensionless parameter needs to be carefully chosen. By employing equations from kinetic theory, Dzwinel and Yuen matched the DPD interaction terms on macroscopic fluid properties.⁵⁴ From a microscopic view on the other hand, the DPD parameters are matched to a coarse-grained microscopic description. In this way parameters for polymers⁵⁵⁻⁵⁷ as well as phospholipids^{12,13} have been derived for DPD.

Polymers in DPD are created by connecting the particles with a spring force. DPD work on polymers started with dilute **polymer solutions**.^{58,59} These polymer-solvent mixtures can phase separate, for instance by addition of surfactant,⁶⁰ applying pressure⁶¹ or depending on the structure of the polymer.^{62,63} Polymers in solution that are at one end grafted to a surface, so-called polymer brushes, have been studied between two surfaces^{64,65} and under shear.^{66,67}

When the concentration of polymers increases, the behaviour changes. Spenley compared scaling laws⁶⁸ for DPD in dilute polymer solution and in **polymer melts** (*i.e.* when no solvent particles are present). When the polymer consists of two blocks of different particles, the polymer melt will show microphase separation. Depending on the composition and phase repulsion, these diblock copolymer melts form various structures, such as a lamellar, hexagonal or micellar phase.⁶⁹⁻⁷¹ Closely related are liquid-crystalline systems, for which DPD simulations show isotropic, smectic and nematic phases.^{72,73} In polymer melts, entanglement of the chains can have a considerable effect on rheology. Pan and Manke added a segmental repulsion force to account for entanglements.⁷⁴ Here it is appropriate to mention the method of Padding and Briels that explicitly detects and prevents crossings,⁷⁵ although it has not been applied on DPD.

Several studies⁷⁶⁻⁷⁹ consider **colloidal suspensions**. The colloids are mod-

elled in DPD by freezing the relative positions of the particles that shape the rigid body. By summing and redistributing all forces over those particles, they move as a solid object, although this method does not prevent fluid particles from penetrating the colloid.

When different types of DPD particles repel each other, they will phase separate. The dynamic behaviour of these **binary mixtures** of immiscible fluids (*e.g.* oil and water) is examined by various authors.^{50,80,81} The addition of **surfactants** stabilizes the interface between the two phases. Surfactant molecules are constructed by linking hydrophilic head particles to hydrophobic tail particles. The surfactants form a monolayer on the interface with the tails pointing to the organic phase, and the heads settling in the water phase. Rekvig *et al.* studied the effect of surfactant structure on the interfacial tension^{82,83} and bending modulus^{84,85} of the monolayer, as well as film rupture.⁸⁶

When the organic phase is absent, the surfactant molecules are left with only water. This type of **amphiphilic systems** is representative for many biological systems. In this context the surfactants are usually called lipids. Depending on the lipid concentration and structure, they can form various amphiphilic mesophases.⁸⁷⁻⁸⁹ In more detail, threadlike micelles⁹⁰ and fluid vesicles⁹¹⁻⁹³ are studied with DPD.

The most important phase of an amphiphilic system is the class of **lipid bilayers**, as they represent cell membranes. The bilayer consists of two monolayers, where the hydrophobic tails point to each other. In order to simulate a realistic membrane, the bilayer needs to be in a tensionless state.^{94,95} Depending on the structure of the phospholipids, the bilayer forms different phases,⁹⁶⁻⁹⁸ including a rippled phase^{99,100} and an interdigitated phase.^{95,101} Also rupture¹² and fusion¹⁰² of membranes are studied with DPD. Finally, the inhomogeneity of the lipid bilayer can induce simulation artifacts.¹⁰³

We conclude with some applications for DPD in **fluid dynamics**. In these simulations either the phenomena, fluids or geometries to be modelled, are complex. When this were not the case, classical flow solvers, such as finite element methods, would be preferable. Examples of complex phenomena are thin film evolution¹⁰⁴ and drop break-up under gravitation or shear.¹⁰⁵ Fluid dynamics problems involving complex fluids are for instance shear simulations on melting polymers,¹⁰⁶ a polymer drop¹⁰⁷ or a DNA molecule.¹⁰⁸ Complex geometries for fluid dynamics include a square capillary,^{37,106} and a periodic array of cylinders¹⁰⁹ or square rods.¹¹⁰

Computational efficiency of Dissipative Particle Dynamics

3.1 Introduction

Over the last decades the computation power of processors has increased rapidly,¹¹¹ accessing ever larger simulation lengths and times. But also the way an algorithm is implemented can affect the time spent on running the simulation. Computational efficiency increases when the same processor produces equally accurate results in less time. This is beneficial for all simulations where computation time is the limiting factor, such as the slow dynamics of complex fluids. The efficiency can be increased in various ways. On a low level, the simulation can be optimized for the processor and available memory. Also the choice of compiler can affect the performance considerably. On a higher level the code itself can be improved by minimizing the total number of computations. As the methods on a low level are hardware specific, we will focus on the latter, more general, method for improving efficiency.

In this chapter we will examine the DPD code in detail. First we need to determine which part of the code is computationally most expensive. The total simulation time $\mathsf{T}^{\text{total}}$ spent on one time step is the sum of three contributions: the time to calculate the pair interactions, $\mathsf{T}^{\text{interact}}$, the time to update the particle positions and velocities, T^{move} and the time to sample properties such as the pressure and temperature, $\mathsf{T}^{\text{sample}}$:

$$\mathsf{T}^{\text{total}} = \mathsf{T}^{\text{interact}} + \mathsf{T}^{\text{move}} + \mathsf{T}^{\text{sample}} \quad (3.1)$$

Note that any overhead created by input or output is not taken into account. Generally, the largest contribution to the total simulation time, is the time to calculate the interaction between particle pairs, T^{interact} . This is mainly due to the selection of interacting particles from all possible pairs, which is a costly process. Reducing the time spent on this selection procedure can increase the overall efficiency considerably. On the other hand, the time evolution of the positions and velocities that takes T^{move} , only involves single loops over all particles, which is a relatively cheap procedure. Using a different update algorithm does not change this and optimization will only marginally improve the efficiency. The time spent on sampling, T^{sample} , depends on the property to be measured. For instance the temperature measurement is a simple procedure, while determining the radial distribution function or static structure factor takes a considerable part of the total simulation time. Therefore we will not take the sampling time into consideration, but we will focus on reducing the time used for the force calculation. We will examine different implementations of the pair selection procedure, *viz.* the cell list, the Verlet list and a combination of both. For each implementation we derive a general expression for T^{interact} , depending on a number of time constants and parameters, such as the time step, temperature, *etc.* We simplify these expressions to remove the dependency of the time constants, so they are based only on the simulation parameters. The objective is to provide a selection criterion to choose the implementation that best fits the simulation parameters from a computational point of view.

3.2 Theory of force calculation

In this section we will describe the three implementations and derive expressions for the time to execute the force calculations. These depend on four time constants, the update frequency and the radius of the Verlet list, that will be discussed in the subsequent paragraphs. We will consider a cubic system of size $L_x L_y L_z$ containing N particles (at a number density n) with particle mass m at a temperature $k_B T$. The time step is Δt , the cut-off radius for the interactions is r_c and the Verlet radius is r_v .

3.2.1 Implementations

Cell list

For the application of a cell list, the simulation domain is divided into cells of volume r_c^3 , provided that all sizes of the simulation domain are multiples of r_c . In a single loop over all N particles, the cell to which a particle belongs is determined. This takes a typical amount of time of τ_{cell} per particle. The position of each particle is compared to the positions of the particles in its own cell and the particles in the adjacent cells. In three dimensions this means 27 cells are considered, each containing $n r_c^3$ particles. The distance between each particle pair is calculated and compared to the cut-off radius, to select the interacting pairs. The time needed for this selection is called τ_{select} . Each particle will typically undergo interaction with $4\pi n r_c^3/3$ selected particles, which takes a time of τ_{force} . The total expression for the time $T_{\text{cell}}^{\text{interact}}$ to calculate the forces using a cell list, is:

$$T_{\text{cell}}^{\text{interact}} = N\tau_{\text{cell}} + \frac{1}{2}N27nr_c^3\tau_{\text{select}} + \frac{1}{2}Nn\frac{4}{3}\pi r_c^3\tau_{\text{force}}. \quad (3.2)$$

The factor $\frac{1}{2}$ in the last two terms arises from Newton's third law: the interaction between particle pair ij is equivalent to the interaction between particle pair ji . Therefore this pair only needs to be considered once.

Verlet list

The Verlet or neighbour list is a list for each particle, that stores all particles within a sphere of radius r_v , that is slightly larger than the cut-off radius r_c . The construction of the list requires a double loop over all particles to select the particles within r_v . The time needed for the selection is again τ_{select} that was introduced in the previous section. In total $4\pi n r_v^3/3$ particles are selected per particle. To add such a particle to the Verlet list, takes a time of τ_{verlet} . Due to the double loop, the construction of the Verlet list is a very expensive procedure, but it is not necessary to repeat it every time step, but only at an update frequency $f_{\text{upd}} < 1$. When calculating the interactions for a particle, its Verlet list provides the $4\pi n r_v^3/3$ particles to be considered. The distance is calculated and compared to the cut-off radius, taking again an amount of

time τ_{select} . Finally, for the particles that are within the cut-off radius, in total $4\pi n r_c^3/3$, the forces are calculated, taking τ_{force} per pair. Thus, the total time to calculate the interactions using a Verlet list, $T_{\text{verlet}}^{\text{interact}}$, is:

$$\begin{aligned} T_{\text{verlet}}^{\text{interact}} = & f_{\text{upd}} \left(\frac{1}{2} N^2 \tau_{\text{select}} + \frac{1}{2} N n \frac{4}{3} \pi r_v^3 \tau_{\text{verlet}} \right) \\ & + \frac{1}{2} N n \frac{4}{3} \pi r_v^3 \tau_{\text{select}} + \frac{1}{2} N n \frac{4}{3} \pi r_c^3 \tau_{\text{force}}. \end{aligned} \quad (3.3)$$

Again, only half the interactions are considered as particle pairs ij and ji are equivalent. Estimations for the update frequency f_{upd} and the Verlet radius r_v will be treated later.

Combination of cell list and Verlet list

The double loop over all particles in the Verlet list implementation can be avoided, when it is combined with a cell list. Now the simulation domain must be divided into cells of size r_v^3 . For each particle, this cell list provides $27 n r_v^3$ particles for pair selection. But instead of directly calculating the forces for pairs within the cut-off radius r_c , the pairs within the Verlet radius r_v are stored in the Verlet list. This whole procedure occurs every $1/f_{\text{upd}}$ time steps. The rest of the force calculation is identical to the Verlet list implementation. The total time $T_{\text{cell+verlet}}^{\text{interact}}$ to calculate the pair interactions using a combination of cell and Verlet list, is:

$$\begin{aligned} T_{\text{cell+verlet}}^{\text{interact}} = & f_{\text{upd}} \left(N \tau_{\text{cell}} + \frac{1}{2} N n 27 r_v^3 \tau_{\text{select}} + \frac{1}{2} N n \frac{4}{3} \pi r_v^3 \tau_{\text{verlet}} \right) \\ & + \frac{1}{2} N n \frac{4}{3} \pi r_v^3 \tau_{\text{select}} + \frac{1}{2} N n \frac{4}{3} \pi r_c^3 \tau_{\text{force}}. \end{aligned} \quad (3.4)$$

This expression does not take into account that the domain is not necessarily an exact multiple of r_v . In fact, in the first τ_{select} -term, r_v^3 should be replaced by the discrete version $r'_{v,x} r'_{v,y} r'_{v,z}$. This discrete Verlet radius is defined as $r'_{v,k} = L_k / (\text{int})(L_k / r_v)$ where $k \in \{x, y, z\}$. This correction introduces the system size in the equation. We will omit it here, to be able to give general guidelines for the choice of implementation.

time constant	estimate	time to
τ_{cell}	0.08	put one particle in cell list
τ_{verlet}	0.004	put one particle in Verlet list
τ_{select}	0.04	check selection criterion
τ_{force}	0.2	calculate pair interaction

Table 3.1: Time constants for force calculation

3.2.2 Time constants

The three implementations discussed in the previous section, involved four time constants, each for a specific task. Measuring the computation time of a simulation that solely performs one of the four tasks gives a rough estimate of the value of the associated time constant. These estimates are listed in Table 3.1 to give an idea of their relative values.

The time constants to put one particle in a cell list, τ_{cell} , or in a Verlet list, τ_{verlet} , differ considerably. The reason is that the construction of the Verlet list only adds the particle number to the list and increments its size by one. But for the construction of the cell list, the cell to which a particle belongs, needs to be determined first, before putting it in the appropriate cell list.

The selection constant τ_{select} accounts for the calculation of the particle distance, checks for any periodic boundary crossings and then compares the distance to the criterion r_c or r_v . The periodic boundary check can often be avoided. In the cell list implementation, the relative position of the cells is known beforehand. An extra variable s_k (where $k \in \{x, y, z\}$) with the possible values $-1, 0$ or 1 marks the periodic boundary. When calculating the particle distance the quantity $s_k L_k$ is added to the distance in direction k . A similar idea for the Verlet implementations is the approach of Bekker *et al.*¹¹² The Verlet list is split into several lists, one for every periodic image. For each particle, a loop over all images provides the particles and variable s_k , thus avoiding the periodic boundary check. Only for the double loop in the Verlet implementation this approach cannot be applied. To be able to compare the different implementations, we will not use the Bekker implementation, but check for periodic boundary crossings during τ_{select} .

With these rough estimates for the time constants, we return to the expressions of the total time for force calculation. The contributions of the τ_{cell} - and τ_{verlet} -

terms are small compared to the contributions of the τ_{select} - and τ_{force} -terms. Furthermore, the τ_{force} contribution is constant for all implementations. When we are only interested in finding the fastest implementation (and not how much faster), it is sufficient to compare the τ_{select} contributions. Therefore, we can simplify the expressions for $T_{\text{cell}}^{\text{interact}}$, $T_{\text{verlet}}^{\text{interact}}$ and $T_{\text{cell+verlet}}^{\text{interact}}$ to T_{cell}^* , T_{verlet}^* and $T_{\text{cell+verlet}}^*$ respectively, that are independent of any time constant.

$$T_{\text{cell}}^* = 27Nnr_{\text{c}}^3, \quad (3.5)$$

$$T_{\text{verlet}}^* = f_{\text{upd}}N^2 + \frac{4}{3}\pi Nnr_{\text{v}}^3, \quad (3.6)$$

$$T_{\text{cell+verlet}}^* = f_{\text{upd}}27Nnr_{\text{v}}^3 + \frac{4}{3}\pi Nnr_{\text{v}}^3 \quad (3.7)$$

These simplified T^* values do not estimate the total times for the different implementations nor their ratio, but should simply indicate the fastest implementation.

3.2.3 Update frequency

For the performance of the Verlet list implementations, we need an estimation for the update frequency f_{upd} . The list must be updated when another particle that is not in the Verlet list enters the cut-off sphere with radius r_{c} . This chance exists when a particle has moved more than $\frac{1}{2}(r_{\text{v}} - r_{\text{c}})$; at that time the Verlet lists for all particles are updated. The particle that causes the Verlet list updating, is - on average - the fastest particle. When we assume that this particle is moving at a constant velocity v_{max} , the update frequency is:

$$f_{\text{upd}} = \frac{\Delta t}{\Delta t_{\text{upd}}} = \frac{2\Delta t v_{\text{max}}}{(r_{\text{v}} - r_{\text{c}})}. \quad (3.8)$$

There are two possible ways to monitor the particle displacements. One possibility is to add the actual displacement to the total displacement of the particle since the last Verlet list update. When the cumulative displacement exceeds

the criterion, the Verlet list is updated. An alternative is to store the particle positions when the Verlet list is updated and compare them to the actual positions at each time step, giving the effective displacement. When the mean free path of the particles is large, no difference is observed between the two different methods. In other words, the cumulative and effective displacement are identical, which means the fastest particle has moved in a straight line. However, when the mean free path is small, the particles will move more erratically, and the cumulative displacement will be larger than the effective. The latter method will then be more efficient to use, as it takes longer to fulfil the update criterion. However, we will use the cumulative method to be able to compare the measured update frequencies to the predictions of equation 3.8.

To use equation 3.8 we need an estimate for the velocity of the fastest particle, v_{\max} . The Maxwell-Boltzmann distribution gives the chance $p(v)$ that a particle at a temperature $k_B T$ and with a mass m will have velocity v :

$$p(v) = 4\pi \left(\frac{m}{2\pi k_B T} \right)^{3/2} v^2 \exp\left(-\frac{mv^2}{2k_B T}\right). \quad (3.9)$$

This velocity v is the maximal velocity if all other $(N-1)$ particles have a smaller velocity. Therefore we need to multiply $p(v)$ by $\left[\int_0^v p(y) dy\right]^{N-1}$ to calculate the chance that v is indeed the maximal velocity. To find the expected value of v_{\max} this chance is multiplied by v and integrated over all possible values of v :

$$E(v_{\max}) = (N-1) \int_0^\infty v p(v) \left[\int_0^v p(y) dy\right]^{N-1} dv. \quad (3.10)$$

The factor $(N-1)$ normalizes the expression. The value for v_{\max} is easy to compute numerically.

3.2.4 Radius of Verlet list

The Verlet radius r_v strongly influences the performance of the Verlet implementations. To keep the update frequency low, r_v should be large, but to keep the fraction of ineffective particles outside the cut-off radius small, r_v should be

small. The value that minimizes the computation time, is a trade-off between these two effects.

For the Verlet implementation, the optimal value for the Verlet list is determined by solving $\partial T_{\text{verlet}}^{\text{interact}}/\partial r_{\text{v}} = 0$ for r_{v} . We neglect the τ_{verlet} -term, while the τ_{force} -term vanishes, because it is independent of r_{v} . Then the optimal value $r_{\text{v,opt}}$ in the Verlet list implementation, is given by:

$$0 = -\Delta t v_{\text{max}} N + 2\pi n r_{\text{v,opt}}^2 (r_{\text{v,opt}} - r_{\text{c}})^2. \quad (3.11)$$

For the combination of the cell and Verlet list, the optimal Verlet radius is determined in the same way, *i.e.* by solving $\partial T_{\text{cell+verlet}}^{\text{interact}}/\partial r_{\text{v}} = 0$ (neglecting τ_{cell} and τ_{verlet}):

$$0 = \Delta t v_{\text{max}} \left(81 r_{\text{v,opt}}^2 (r_{\text{v,opt}} - r_{\text{c}}) - 27 r_{\text{v,opt}}^3 \right) + 2\pi r_{\text{v,opt}}^2 (r_{\text{v,opt}} - r_{\text{c}})^2. \quad (3.12)$$

3.3 Computational details

To validate the derivations of the previous section, we first examine the effect of the Verlet radius on the computation time. We consider a cubic system of volume $(10r_{\text{c}})^3$ containing $N = 6000$ particles (*i.e.* number density $n = 6$), at a thermal energy $k_{\text{B}}T = 0.5$, friction coefficient $\gamma = 20$ and phase repulsion $\alpha_{ij} = 0$ (*i.e.* an ideal DPD fluid). The friction coefficient and phase repulsion do not affect our predictions, which is also confirmed by simulations. For this reason, they will not be varied. The particle mass m and the cut-off radius r_{c} are unity and the time step is $\Delta t = 0.01$. After an equilibration time of 10 time units, the CPU time is measured for 100 time units. The Verlet radius for the Verlet implementation ranges from $1.5 \leq r_{\text{v}} \leq 2.9$, and for the combined implementation from $1.1 \leq r_{\text{v}} \leq 2.0$.

Next, we vary the temperature in a system of $6 \times 6 \times 6 r_{\text{c}}^3$ containing $N = 1296$ particles (*i.e.* number density $n = 6$), at a friction coefficient $\gamma = 20$, phase repulsion $\alpha_{ij} = 0$ and time step $\Delta t = 0.015$. The particle mass and the cut-off radius are unity again. After an equilibration time of 15 time units, the CPU time is measured for 750 time units. The thermal energy ranges from $0.1 \leq k_{\text{B}}T \leq 1.0$. The maximal velocity v_{max} and update frequency f_{upd} are

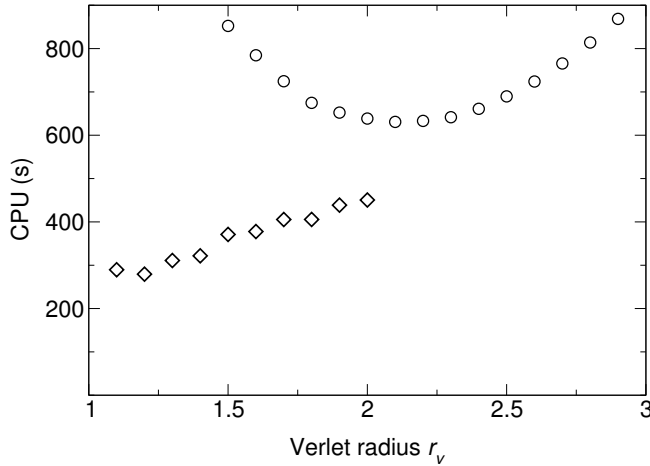


Figure 3.1: Measured CPU time as a function of the Verlet radius r_v for Verlet implementation (○) and combination of cell and Verlet implementation (◇), at $N = 6000$, $n = 6$ and $k_B T = 0.5$

measured to compare to the theoretical values. We measure the computation times to identify cross-over points and compare them to those predicted by the T^* values.

By balancing the T^* values, we can determine the cross-over points for the implementations at different parameter settings. From these points, we can construct an implementation diagram, that indicates the fastest implementation in that parameter space. We will show such implementation diagrams of the number of particles, number density and temperature versus the time step, varied around $N = 6000$, $n = 6$ and $k_B T = 0.5$.

3.4 Results and discussion

3.4.1 Optimization of Verlet radius

We determine the optimal Verlet radius in simulations to compare it to the derived prediction. For the Verlet implementation, solving equation 3.11 provides an optimal value of $r_{v,opt} = 2.09$. Figure 3.1 shows that this value agrees excellently with the measured computation times. The optimal Verlet radius

for the combination of cell and Verlet list is $r_{v,opt} = 1.26$ (from eq. 3.12). This value agrees well with the measured computation times in figure 3.1. This curve is more irregular than the curve for the Verlet implementation, because of the system size effect. However, it does not influence the optimal value for the Verlet radius. So, we can use the computed Verlet radii for the simulations, instead of experimentally determining the best value.

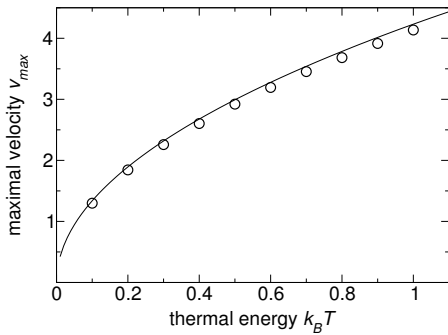
Note that when the Verlet radius is not optimal, the computation time increases rapidly. However, it is common practice to fix the value of the Verlet radius for all parameter sets (*e.g.* at a value of $1.5r_c$), which can be harmful for the computational performance.

3.4.2 Computation time as a function of temperature

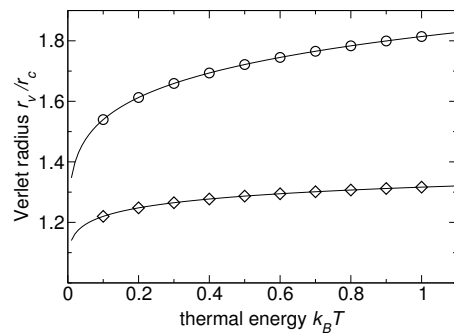
Next, the three implementations are tested as a function of temperature. Figure 3.2 shows the trends of the maximal velocity, Verlet radius and update frequency. The first figure (fig. 3.2(a)), compares the measured average velocity of the fastest particle to the predicted maximal velocity. They agree well, the measurements being slightly smaller than theory. Not surprisingly, the maximal velocity increases for higher temperatures.

We use the maximal velocities to determine the optimal Verlet radius, that is shown in figure 3.2(b). The theoretical values are used for the simulations, so the “measurements” and theory coincide. The radius for the Verlet list is larger than for the combined lists. This is because the construction of the Verlet list in the first implementation is computationally more expensive (due to the N^2 -term). Therefore, the optimal Verlet radius is higher, as it lowers the number of times this procedure must be carried out. Also at higher temperatures, the Verlet radius for both lists increases. Due to the larger maximal velocity, the lists need to be constructed more often. A higher Verlet radius lowers this frequency to minimize the computation time.

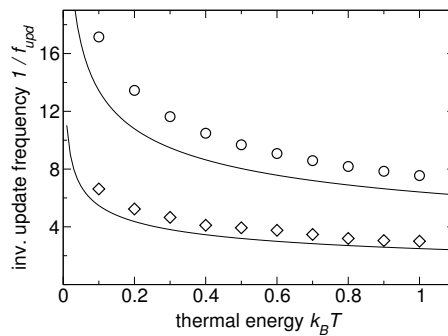
With the maximal velocity and Verlet radius, we can plot the inverse update frequency in figure 3.2(c). Here the theoretical curve and measurements differ considerably. In the simulations the number of time steps between updates is higher than predicted. The discrepancy is caused by the fact that the particle that causes the Verlet list update, is not necessarily the fastest at each time step. This effect becomes more pronounced at lower frequencies, as is confirmed by the measurements at low temperatures.



(a)



(b)



(c)

Figure 3.2: (a) Maximal velocity, (b) Verlet radius and (c) inverse update frequency as a function of the temperature; for Verlet list (\circ), combination with cell list (\diamond) and theory ($—$), at $N = 1296$ and $n = 6$.

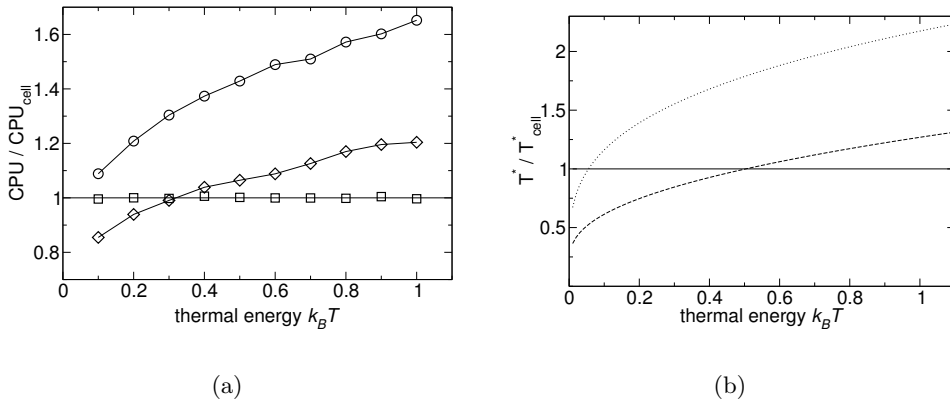


Figure 3.3: (a) Computation time as a function of temperature, for cell list (\square), Verlet list (\circ) and combination with cell list (\diamond); (b) T^* values as a function of temperature, for cell list ($—$), Verlet list (\cdots) and combination of Verlet and cell list ($- -$), at $N = 1296$ and $n = 6$.

The measured computation times and T^* values as a function of the temperature, are shown in figures 3.3(a) and 3.3(b). We cannot compare them directly, as the first measures the actual computation time, while the latter only takes the selection terms into account. As a result the scales of the respective y-axes differ. However, we can compare the cross-over points of the implementations in both graphs. The computation times and T^* values are normalized on the results of the cell list. The Verlet implementation does not have a break-even point with the cell list implementation in the measured temperature range, but the T^* values are equal at $k_B T = 0.06$, which is lower than the lowest simulated temperature. The cross-over between the cell list and combined list implementations is predicted at $k_B T = 0.51$, while measurements show a break-even point at $k_B T = 0.32$. The reason is that we did not take the system size effect into account in our derivation for the combined list implementation. When the system size is not an exact multiple of r_v^3 , it must be divided into larger cells, which increases the number of particle pairs to consider. Due to the higher computation time, the break-even point shifts to a lower temperature. This effect will be less pronounced in larger systems.

It is important to identify the parameter regimes that are significant for DPD. As most DPD simulations are performed for liquids, it should be verified whether the parameters are in the liquid regime. This is determined by Schmidt

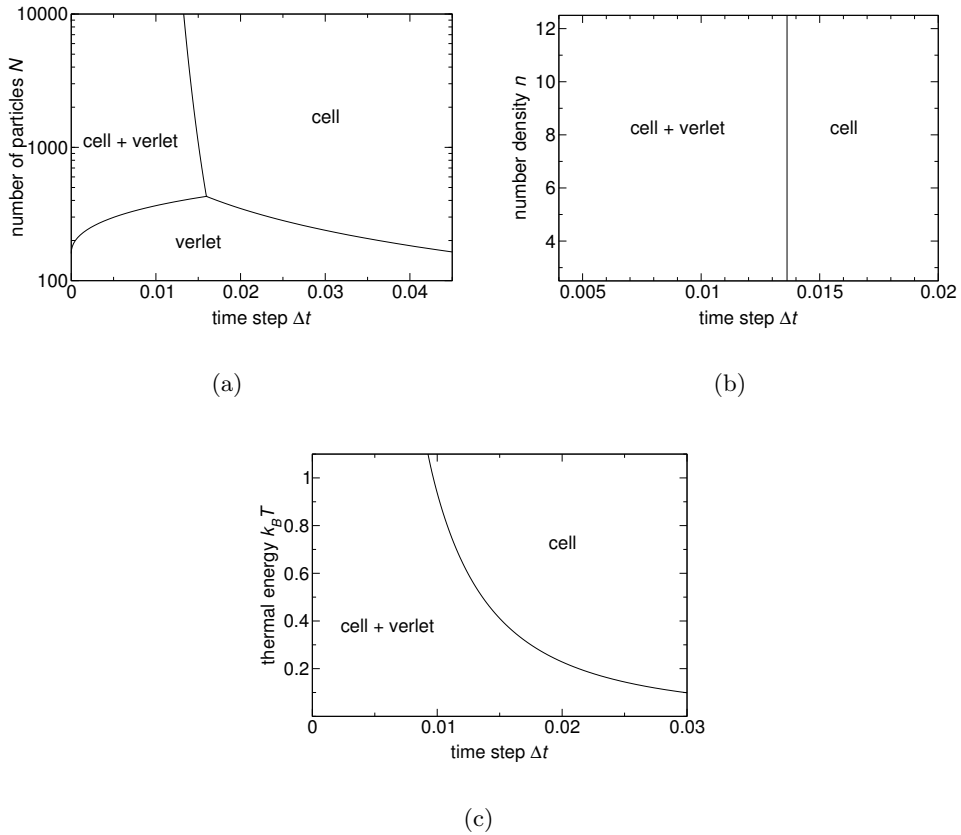


Figure 3.4: Implementation diagrams of (a) number of particles, (b) number density and (c) temperature versus time step; at $N = 6000$, $n = 6$ and $k_B T = 0.5$.

number, that is the ratio of the kinematic viscosity over the diffusion constant. The fluid is in a gaseous regime at low Schmidt numbers, when momentum transport by diffusion prevails over the transport by friction or drag. Conversely, at high Schmidt numbers, when diffusion is negligible compared to viscosity, the fluid is in a liquid regime. In figure 3.3 this regime corresponds to low temperatures, at which the combined list implementation is fastest.

3.4.3 Implementation diagrams

An implementation diagram shows the areas in which a specific implementation is the fastest, as a function of two parameters. Figure 3.4 shows three diagrams for the number of particles, number density and thermal energy versus the time step. While one is varied, the other two parameters are fixed, the fixed values being $N = 6000$, $n = 6$ and $k_B T = 0.5$.

Figure 3.4(a) shows the implementation diagram for the number of particles versus the time step. At small particle numbers the Verlet implementation is favourable, because the “N-squared” term is not harmful yet. At higher numbers and large time step the cell list is preferable over the combination with a Verlet list. This is because the larger time step forces the Verlet list to be updated more frequently, making the combined implementation less profitable. The implementation diagram in figure 3.4(b) shows the number density versus the time step. At the fixed number of particles of $N = 6000$, the Verlet list implementation is never favourable. Only one cross-over point remains, as the choice between the cell list and the combination does not depend on the number density. The diagram for the temperature versus the time step is shown in figure 3.4(c). Again the Verlet list implementation does not play a significant role. As already shown in figure 3.3 the combined implementation performs better at lower temperatures. To assess whether these theoretical diagrams can quantitatively identify the best implementation, they should be compared to measurements, such as has been done in the previous section 3.4.2.

The parameter settings at which these implementation diagrams are constructed, correspond to a Schmidt number of approximately unity. In fact, the liquid regime - at high Schmidt numbers - is of more interest for DPD simulations. Changing the parameters accordingly shifts the diagrams to the advantage of the implementation that combines the Verlet and cell list.

3.5 Conclusion

In this chapter we examined the computational efficiency of DPD. Three implementations were considered to calculate the pairwise forces in DPD, *viz.* the cell list, the Verlet list and the combination of both. For each implementation, we derived an expression for the total computation time. Neglecting the smaller contributions, we reduced the expressions to simplified theoretic-

cal values, that are only dependent on the input parameters. These values should predict the optimal value for the Verlet radius and identify the fastest implementation, based on the parameter setting.

The optimized values for the Verlet radius match the measurements very well. The simplified theoretical values can reasonably indicate the fastest implementation. However, both objectives are only tested for one parameter setting. For a full assessment of the theory, more measurements are required.

In general, DPD simulations are performed with such a large number of particles, that the Verlet implementation is always unprofitable. When DPD is used to model liquids, the diffusive motion of the particles should be limited (*i.e.* high Schmidt number). Depending on the time step, the implementation that combines the Verlet and cell list, is usually more efficient in this regime than the cell list implementation.

Viscosity measurement in Dissipative Particle Dynamics

4.1 Introduction

The viscosity of a fluid is the main parameter that determines its flow characteristics. Realistic fluids, for which the rheological behaviour is of intrinsic interest, can be simulated with particle models, notably Molecular Dynamics (MD).¹⁰ Recently, the idea of using simpler particle models, intended to reproduce hydrodynamic behaviour, has also been applied in an attempt to overcome the time-scale limitations inherent in Molecular Dynamics.^{1, 113, 114} In these simple model fluids the viscosity generally has to be measured because sufficiently accurate theoretical expressions are lacking.⁴ That is, ideally the viscosity would be an input parameter but in practice the simulation must be “calibrated”. There are several methods available to calculate viscosity, all with their relative advantages and disadvantages. In this article we describe a novel method that we show is advantageous in some circumstances. We begin with a review of the techniques currently available.

A division can be made between equilibrium and non-equilibrium methods. In the first the simulated system is first equilibrated and subsequently remains in equilibrium. The viscosity is then calculated from the stress-stress auto-correlation function through the Green-Kubo relation.¹⁰ Because the system is in equilibrium, simple periodic boundary conditions are adequate. Furthermore, the shear rate is by definition zero, so one is automatically in the linear regime. These facts make this method very appealing, but unfortunately the

large fluctuations in the equilibrium stress lead to a poor signal to noise ratio. An alternative equilibrium method, proposed by Palmer,¹¹⁵ is based on the transverse-current autocorrelation function. One can extract the viscosity from the decay of this function, if one assumes the hydrodynamic prediction for its functional form. This additional assumption means that it is not obviously preferable to the stress-stress autocorrelation function, although it is efficient for purely dissipative systems.¹¹⁶ Hess has compared these two equilibrium methods to non-equilibrium methods in Molecular Dynamics simulations.¹¹⁷ His results show that, despite their undoubted advantages, both equilibrium methods suffer from worse statistics than non-equilibrium methods.

In non-equilibrium methods the fluid is subjected to an external perturbation that may be constant or temporally varying. The properties of the non-equilibrium steady state or the decay to the equilibrated state are then related to the viscosity. One example is a periodic perturbation employed to generate an oscillatory velocity profile,¹¹⁸ depending on the frequency of the sinusoidal external force. Several measurements at different frequencies and extrapolation to zero-frequency are required to determine the viscosity. A more recent method¹¹⁹ imposes a pulsed Gaussian velocity profile on the system. The decay of the Gaussian peak gives an estimate of the viscosity. Müller-Plathe obtains a linear velocity profile by cleverly swapping impulses of spatially remote particles.¹²⁰ The interchanged momentum and the measured velocities provide the viscosity. More common is the converse procedure, imposing a linear profile at a fixed shear rate and measuring the resultant shear stress. This shear flow method is the underlying principle of most experimental rheometers. In simulations however, problems arise at the boundaries of the simulation domain, as periodic boundary conditions are not able to maintain a steady linear velocity profile. Ways around this include stochastic boundary conditions¹²¹ and Lees-Edwards boundary conditions.³¹ The latter retain periodicity but alter the position and velocity of the periodic images. An important advantage of the shear flow method is the constant shear rate, which enables one to study the dependence of the viscosity on the shear rate. For example, the viscosity of a Dissipative Particle Dynamics (DPD) model of a colloidal suspension was studied using this approach.⁷⁶ The shear flow method is also successfully applied to determine the viscosity in stochastic rotational dynamics (SRD) models.¹²² Less widely used is a similar “flow” method that uses a Poiseuille flow profile. The problem with this approach is that the boundary condition issues are still unresolved. Considerable density fluctuations near the system boundaries are present in MD simulations,¹²³ SRD simulations¹²⁴ as well as DPD simulations.¹²⁵ Relative to equilibrium methods extra care is needed when using any

non-equilibrium method. The use of external fields causes viscous heating. The energy supplied to the system causes the temperature to rise monotonically for a steady perturbation. In that case a heat sink is required to drain the excess energy, *e.g.* through the use of a thermostat. Furthermore, the flow needs to be in the linear regime and extrapolation to the zero perturbation limit may be required.

Concluding from the existing literature, the stress autocorrelation method and the shear flow method are favoured in most particle models to measure the viscosity. In this chapter we propose an improvement of the Poiseuille flow method by introducing periodicity, that solves the boundary problem. We compare the new method to the stress autocorrelation and shear flow method and show its advantages, both in accuracy of the results and in ease of implementation. For the comparison of the methods, DPD simulations are a useful tool. Viscous heating is absent because of the built-in thermostat, and DPD fluids exhibit Newtonian behaviour over a large range of parameters.¹ The periodic Poiseuille flow method is not limited to DPD but can also be applied in other particle models. To illustrate this, we use the same methodology to calculate the viscosity of a Lennard-Jones fluid in an MD simulation.

4.2 Periodic Poiseuille flow method

Poiseuille flow is obtained by applying a body force, such as a gravitational force or pressure gradient, to each particle. For instance, if gravity works in the z -direction on a fluid between two plates in the xy -plane, the resulting flow field has a linear shear stress profile and a parabolic velocity profile:¹²⁶

$$\tau_{xz} = \rho g_z \left(x - \frac{1}{2}D \right), \quad (4.1)$$

$$v_z = \frac{\rho g_z}{2\eta} \left(xD - x^2 \right). \quad (4.2)$$

In these equations τ_{xz} is the shear stress, v_z is the velocity directed parallel to the force, η is the dynamic viscosity, ρ is the mass density, g_z is the gravitational constant, x is the position between the two plates and D is the distance between the plates. Integration of equation 4.2 with respect to x over the distance between the plates D gives the system average of the velocity $\langle v_z \rangle$:

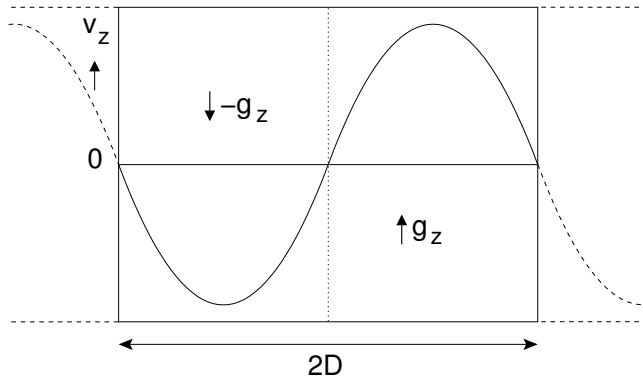


Figure 4.1: Schematic representation of the periodic Poiseuille flow method.

$$\langle v_z \rangle = \frac{1}{D} \int_0^D v_z(x) dx = \frac{\rho g_z D^2}{12\eta}. \quad (4.3)$$

Measurement of this system average directly gives the viscosity of the fluid. Note that this derivation assumes a constant density in the simulation domain and a constant viscosity over varying shear rates. The shear rate dv_z/dx ranges from zero, where the velocity is at a maximum, to a maximum value where the velocity is zero. So, in contrast to the shear flow method, this method is not suitable for determining the shear rate dependent viscosity. It is limited to fluids in the Newtonian regime.

Periodic boundary conditions are not sufficient, because they cannot keep the fluid at zero velocity at the edges of the simulation domain. To circumvent this problem Kauzlarić *et al.* implement stochastic boundary conditions¹²⁵ and Allahyarov and Gompper use a bounce back condition.¹²⁴ However in both cases the density profile shows artefacts at the boundary. An easier solution to this problem is to subdivide the system into two domains, and apply a body force in the opposite direction in the two domains. Figure 4.1 explains this idea schematically. Because of the periodic boundary conditions the counteracting body forces constrain the fluid at the positions where the plates would be. Note that to obtain the system average, the velocity v_z has to be measured in the direction of the external force.

Summarizing, the proposed Poiseuille flow method is limited to fluids in the linear regime. Despite this drawback it could have significant advantages over other methods. We expect a more accurate result compared to the shear flow method, as the system average of the velocity is less susceptible to noise than the shear stress average. This is because, in continuum terms, the shear stress is related to a derivative of the velocity field. The body forces are easily implemented, and when opposing forces are applied, ordinary periodic boundary conditions can support the parabolic flow fields.

4.3 Computational details

Three different viscosity measurement techniques, the stress autocorrelation method, the shear flow method and the periodic Poiseuille flow method, are compared in simulations. We use Dissipative Particle Dynamics (DPD), a mesoscale particle model that was introduced by Hoogerbrugge and Koelman¹ in 1992 and put on a sounder theoretical footing by Español and Warren.² In short, every particle is meant to represent a fluid element that experiences other fluid elements within a relatively small cut-off radius r_c . They are kept in motion by a random force and counteracted by a drag force depending on the velocity of surrounding particles. Together these counteracting forces act as a thermostat. All forces are finite and smooth allowing for large time steps. For our purposes DPD has many advantages over other particle methods. Because of the small cut-off radius r_c , the stress autocorrelation function does not suffer from finite size effects. The thermostat removes the risk of viscous heating and, due to the large time step, simulations do not take much computation time. Conservative forces are not needed for this study and will be left out.

Marsh *et al.* derived an estimate for the dynamic viscosity η based on kinetic theory.⁴ There are two contributions to the viscosity. The first term is a kinetic contribution originating from the motion of the individual particles and the second term is a dissipative contribution from the energy dissipation between particles:

$$\eta = \frac{45}{2\pi} \frac{m(k_B T)^2}{\sigma^2 r_c^3} + \frac{\pi}{1575} \frac{n^2 \sigma^2 r_c^5}{k_B T}. \quad (4.4)$$

Here n is the number density, σ the noise amplitude that drives the random

force, T the temperature and k_B Boltzmann's constant. Although the trends are well described by equation 4.4, the prediction deviates considerably from the actual viscosity. It can, however, help in the parameter choice. The viscosity of a liquid is mainly attributed to the dissipative term, whereas viscosity of a gas is largely determined by the kinetic contribution. As we are interested in measurements in the liquid regime, we should choose the parameters accordingly. Equation 4.4 shows that a large noise amplitude σ , low temperature $k_B T$ and a high number density n achieve this objective. We choose the parameters for the base case as noise amplitude $\sigma = 4.5$, number density $n = 6$ and temperature $k_B T = 0.5$, in units where the particle mass and the cut-off radius r_c are unity. Furthermore, the time step $\Delta t = 0.01$ and the simulation box size is $12 r_c$ in x -direction and $8 r_c$ in y - and z -direction. The simulation time after equilibration is 100 time units, but when calculating profiles for the periodic Poiseuille flow method one long run of 1000 time units is performed. For the comparison of the viscosity measurement methods the noise amplitude, number density and temperature are varied. We repeat the simulation for each parameter set for each measurement method ten times, from which the average viscosity and standard deviation are calculated.

For the stress autocorrelation function the shear stress in three directions is calculated at every time step. The maximum correlation time is set to 20 and the viscosity is calculated from the average over the last half of the integrated correlation function, where the stress-stress autocorrelation function is statistically indistinguishable from zero. The linear velocity profile required in the shear flow method is achieved with Lees-Edwards boundary conditions.³¹ For the shear flow method as well as the Poiseuille flow method the external force must be determined. For each parameter set a test run is performed to obtain an estimate for the viscosity. Based on this preliminary viscosity and the requirement for a small Reynolds number ($Re = \rho \langle v_z \rangle D / \eta$, here we take $Re = 4$) the average velocity in the flow direction is calculated. For the shear flow method the shear velocity is twice this average velocity. The corresponding body force for the Poiseuille flow method is calculated from equation 4.3. For both flow methods the estimated velocity profile is imposed on the particles at the start of the simulation. Measurements take place every 10 time steps and the velocities are corrected by the addition of a constant factor every 100 time steps to enforce the condition of zero total momentum.

The three methods are also used to measure the viscosity of a Lennard-Jones fluid with Molecular Dynamics at constant NVT.^{10,19} In contrast to the DPD simulations, only one state point is considered, at a reduced¹⁰ number density

$n = 0.8442$ and reduced temperature $T = 0.722$. The cut-off radius for the Lennard-Jones potential is $r_c = 2.5$ and the time step is $\Delta t = 0.001$. Because two of the methods involve an external force, a thermostat is required to keep the temperature constant. The DPD thermostat (*i.e.* the random and dissipative forces) can be used for this purpose,¹²⁷ but we choose the Lowe-Andersen thermostat¹⁸ instead. This avoids any time step dependence of the temperature. The collision parameter is $\Gamma = 20$ and the cut-off radius for the thermostat is $r_c^{\text{th}} = 1.1$. As this thermostat enhances viscosity, the system in equilibrium for the stress autocorrelation method must also be thermostatted. In this way we can compare the results of the three methods. For the stress autocorrelation method the average of the Green-Kubo integral from 10 to 30 time units is taken to determine the viscosity. For the shear flow method the shear velocity is $V_{\text{shear}} = 0.04$ and for the periodic Poiseuille flow method the body force is $g_z = 0.02$. These external forces result in a maximum velocity well below the kinetic velocities. The simulation domain is a relatively large cube of volume 16^3 . The simulations are run for 200 time units after an equilibration time of 50 time units and repeated ten times.

4.4 Results and discussion

In this section we first examine the DPD results of the periodic Poiseuille flow method for the base case. Next, we compare the DPD results of the stress autocorrelation method, the shear flow method and the periodic Poiseuille flow method over a range of parameters. Finally, we compare the three methods for determining the viscosity of a Lennard-Jones fluid at one state point.

4.4.1 Validation of the periodic Poiseuille flow method

A long simulation for the base case parameters ($n = 6$, $\sigma = 4.5$ and $k_B T = 0.5$) and an external force of $g_z = 0.055$ allows us to determine the profiles of various properties to assess the periodic Poiseuille flow method in detail. The density profile, shown in figure 4.2(a), is uniform except for statistical fluctuations. Density artefacts as found in other implementations of the Poiseuille flow^{124, 125} are absent in our method, as is required for a bulk Poiseuille flow.

The difference between the particle velocities and the local velocity based on the instantaneous viscosity is measured to plot the temperature profile. The

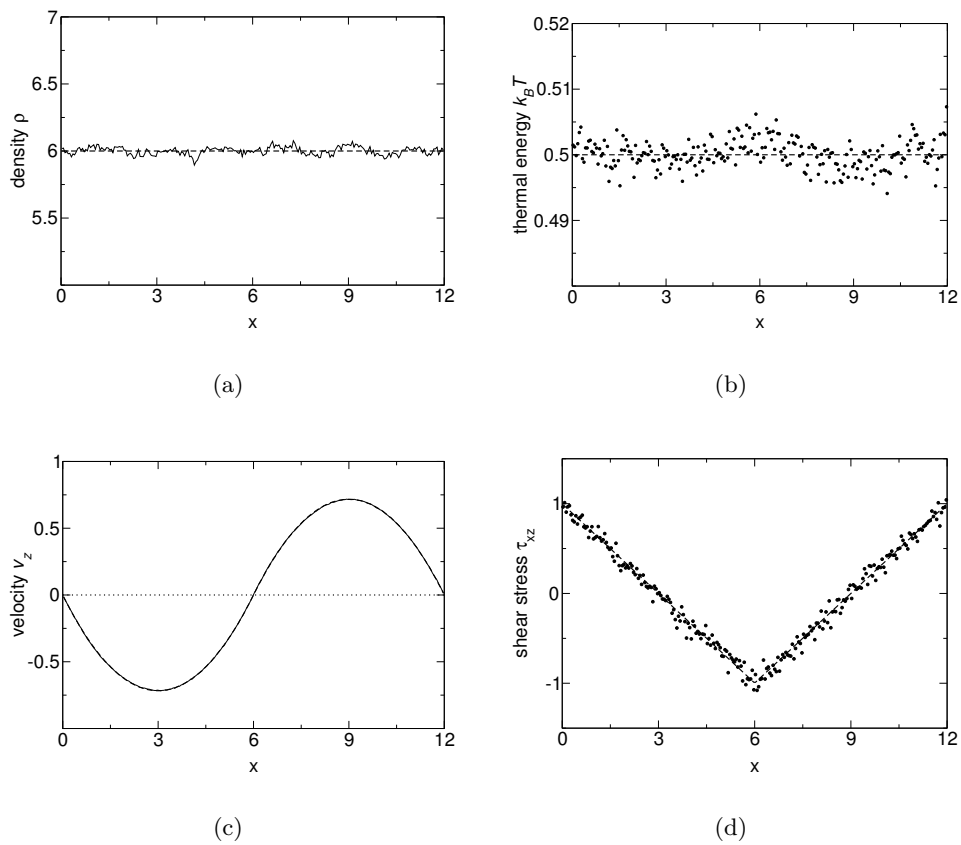


Figure 4.2: Profiles for periodic Poiseuille flow method for (a) density, (b) temperature, (c) velocity and (d) shear stress; measured profile (— and •) and theoretical value (—).

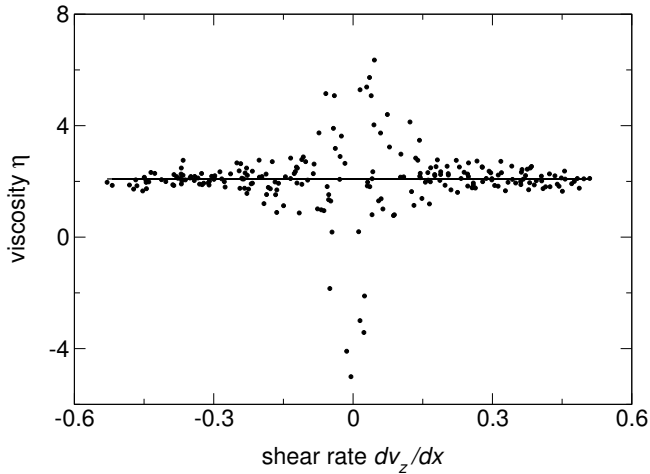


Figure 4.3: Viscosity as a function of shear rate: measurements (●) and value based on the average velocity measurement (—).

measured temperature was 2% lower than the input value, because the time step of $\Delta t = 0.01$ proved to be too large for the thermostat to work properly. This is a well known artefact of the simple DPD algorithm,^{4,22} which is resolved in the Lowe-Andersen thermostat.¹⁸ To distinguish between the effect of a large time step and viscous heating, the same simulation was performed with a smaller time step of $\Delta t = 0.001$. The resulting temperature profile is shown in figure 4.2(b). Although the shape of a parabolic profile is present, the deviations from the input value are sufficiently small to neglect viscous heating.

The measured velocity profile is shown in figure 4.2(c) together with the parabolic flow field (eq. 4.2) based on the measured viscosity. To within an accuracy of 0.1% the profile agrees with the analytic laminar Poiseuille flow field. This justifies the use of the average velocity as a means to calculate the viscosity (eq. 4.3).

The shear stress is determined from the pair interactions and particle velocities.¹⁰ In figure 4.2(d) we show that the shear stress profile is linear in accordance with equation 4.1, and shows a sudden change at the extreme values due to the switch of the direction of the external force. The deviations from the theoretical stress profile are not statistically significant.

The measured viscosity is $\eta = 2.09 \pm 0.02$, based on the overall average of the

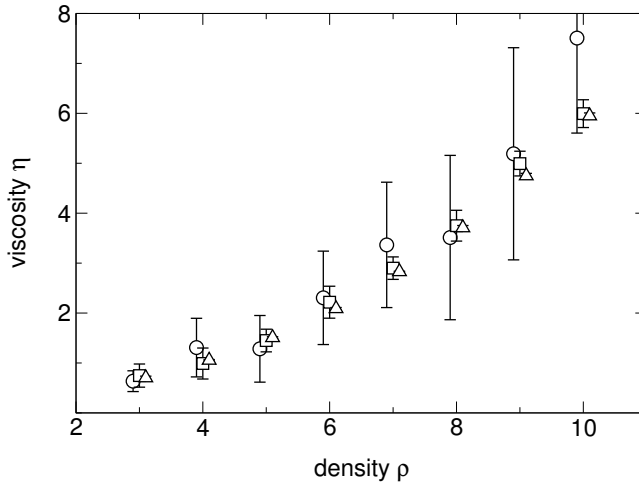


Figure 4.4: Viscosity as a function of density, measured with stress autocorrelation function (\circ), shear flow method (\square) and periodic Poiseuille flow method (\triangle); symbols are horizontally shifted for clarity purposes.

velocity. To test whether this viscosity is constant over all shear rates, *i.e.* whether the liquid is in the Newtonian regime, we determine the viscosity as a function of the shear rate. From the velocity profile the local velocity gradient at each position is computed from the velocities of the two neighbouring data points. As the slope of the parabolic profile is a linear function of the position, this procedure should be sufficiently accurate. For a Newtonian fluid the ratio of the measured shear stress and the local velocity gradient gives the local viscosity. This is plotted against the local shear rate in figure 4.3. The viscosity is equal to the measured overall viscosity for high and intermediate shear rates. The scatter at low shear rates reflects the larger numerical errors due to the small values for both the shear stress and the velocity gradient. This results in a large error in their ratio. Given this, we find no evidence that the fluid is not Newtonian. It should be emphasized again that this is not a way to measure shear rate dependent viscosities. If the viscosity is not constant, the measured average velocity will in fact reflect the effects of the spatially varying viscosity.

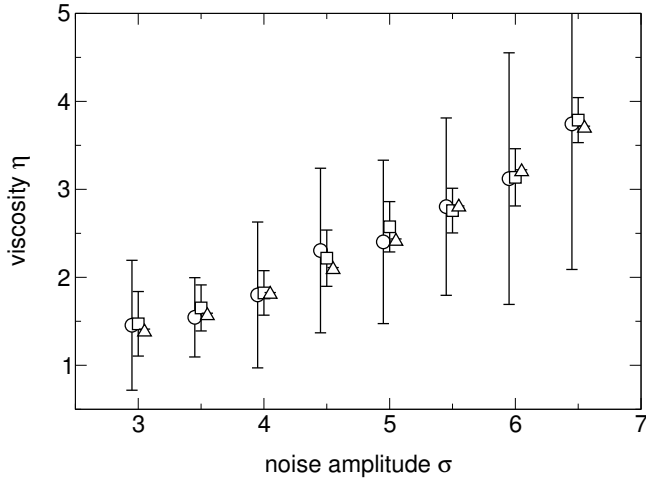


Figure 4.5: Viscosity as a function of noise amplitude, measured with stress autocorrelation function (\circ), shear flow method (\square) and periodic Poiseuille flow method (\triangle); symbols are horizontally shifted for clarity purposes.

4.4.2 Comparison of viscosity measurement techniques in DPD

We calculate the viscosities for different parameter sets with the three different methods. Figure 4.4 shows the viscosity with error bars as a function of density. The measured averages of the three methods agree well within their error bars, but the size of the error bars differs greatly. As expected, the stress autocorrelation method is the least accurate. Of the two flow methods the periodic Poiseuille flow method gives the better results. The error bars of these two methods do not change significantly over the density range, because the simulations are performed at similar Reynolds numbers.

The viscosity as a function of noise amplitude, plotted in figure 4.5, also shows that the averages from all methods are in agreement. Again the error bars of the shear flow method are smaller than for the autocorrelation method, but larger than for the Poiseuille flow method. The accuracy of the stress autocorrelation method deteriorates with increasing noise amplitude. This is because the dissipation in the system increases and the fluctuation in the stress measurements is mainly due to the dissipative part. This effect is not apparent for the shear flow method, as the collective motion dominates the chaotic motion of the particles. The error bars do not change due to the constant Reynolds

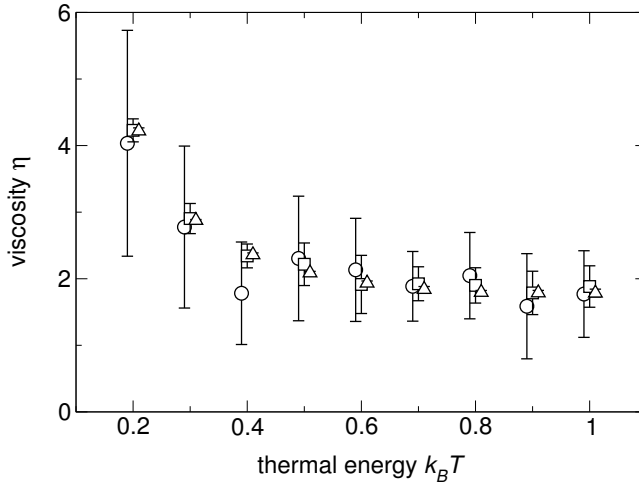


Figure 4.6: Viscosity as a function of temperature, measured with stress autocorrelation function (\circ), shear flow method (\square) and periodic Poiseuille flow method (\triangle); symbols are horizontally shifted for clarity purposes.

number. For the periodic Poiseuille flow method the standard error shows no discernible trend.

The measured viscosities at different temperatures in figure 4.6 agree within the statistical accuracy for the three methods. The stress autocorrelation method gives poor results at low temperatures, but gains in accuracy with increasing temperature. The cause is the smaller role of the dissipation in the stress measurements, as explained above. Again, the error bars of the flow methods are not affected by the change in temperature. The most accurate over all temperatures is the periodic Poiseuille flow method.

A comparison between the methods is made, based on the average accuracy of all parameter sets. We estimate that the periodic Poiseuille flow method is 10 times as accurate as the shear flow method and 30 times as accurate as the stress autocorrelation method. Of course these ratios depend on the parameter set and are only mentioned to illustrate the disparate relative performance. A more significant measure to compare the methods is the number of measurements needed to arrive at a result within a certain confidence interval. To this end we performed 100 independent simulations with the base case parameters for each measurement method. The 100 viscosity results are grouped in sets of N measurements, where N ranges from one to ten. The average deviation

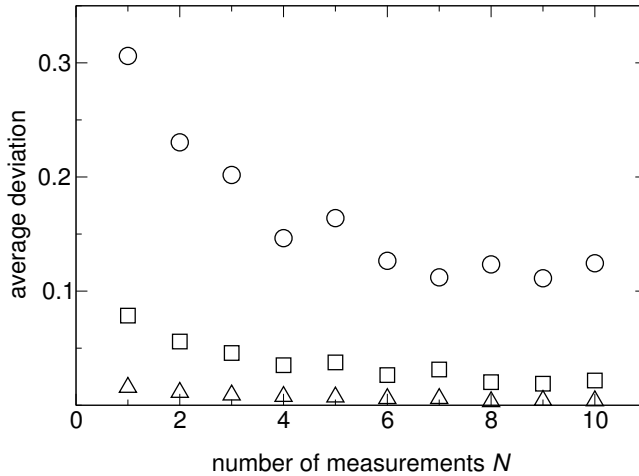


Figure 4.7: Average deviation of viscosity from average viscosity as a function of number of measurements for stress autocorrelation method (○), shear flow method (□) and periodic Poiseuille flow method (△).

of the sets from the overall average is plotted in figure 4.7 as a function of set size N . The figure shows that a single measurement of the autocorrelation method can deviate 30% from the actual viscosity, *i.e.* the average of 100 measurements. Even ten measurements lead to an average that is unacceptably inaccurate. The shear flow method does achieve acceptable results but requires multiple measurements, while for the periodic Poiseuille flow method all single measurements are within a 98% confidence interval. Thus, even one short simulation is sufficient to achieve accurate results.

4.4.3 Comparison of viscosity measurement techniques for a Lennard-Jones fluid

Finally, we test the three methods on a Lennard-Jones fluid at a reduced density of $\rho = 0.8442$ and reduced temperature of $T = 0.722$. At this state point, Meier *et al.* collected the results of several studies.¹²⁸ They report viscosities ranging from 2.7 to 4.0, depending on measurement technique, simulation time and system size. Our results are as follows:

- stress autocorrelation method: $\eta = 3.7 \pm 1.2$,
- shear flow method: $\eta = 3.4 \pm 0.8$,
- periodic Poiseuille flow method: $\eta = 3.4 \pm 0.2$,

well within the range quoted in reference 128. The average viscosities of the three methods are in agreement. As expected the stress autocorrelation method gives the least accurate results, while the periodic Poiseuille flow method is four times as accurate as the shear flow method. This example illustrates that the periodic Poiseuille flow method is not only limited to simple fluid models, but is applicable to realistic systems as well. The latter often involve long simulation times, that can be reduced considerably by our more accurate method. We should point though, that the non-uniform shear stress means the method is restricted to the Newtonian regime.

4.5 Conclusion

The Poiseuille flow method to measure the viscosity in a particle model is improved, guaranteeing a flat density profile throughout the simulation domain. Shear stress measurements are not required, as the system average of the velocity in the flow direction directly provides the viscosity of the fluid. Moreover, the application of opposing external forces enables the use of ordinary periodic boundary conditions.

Series of DPD simulations were performed to measure the viscosity with the stress autocorrelation method, the shear flow method and the periodic Poiseuille flow method. The methods give comparable results for the average of the measured viscosity, but the accuracy differs greatly. Being an equilibrium method, the stress autocorrelation method is the least accurate. The periodic Poiseuille flow method gives the best results, roughly ten times as accurate as the shear flow method. Also for a Lennard-Jones fluid our method proves to be advantageous. Thus, the periodic Poiseuille flow method reduces the computation time needed to determine the viscosity of a particle model fluid.

Coarse-graining Dissipative Particle Dynamics

5.1 Introduction

The behaviour of fluids spans a wide range of length and time scales. On a macroscale, the Navier-Stokes equation describes hydrodynamics of fluids.¹²⁶ At the other end of the spectrum, Molecular Dynamics (MD)¹⁰ capture microscale processes, that incorporate the Brownian motion of molecules. However, the behaviour of complex fluids occurs on a level between these extremes, the mesoscale. On this level of description, macroscale models provide too little detail and microscale models are computationally too expensive. Compared to MD, mesoscale models simulate typical lengths and times of micrometers and milliseconds, rather than nanometers and microseconds. Still, mesoscale particle models suffer from long computation times, compared to their continuum counterparts. Therefore, it can be beneficial to make a distinction between regions that require detailed simulation, and bulk areas that are of lesser interest. To achieve such a multiscale system, the mesoscale model must first be complemented with a coarse-grained description.

Moving up in level of description, or coarse-graining,¹²⁹ reduces the level of detail. For instance, in fully atomistic simulations the hydrogen bond vibrations can be eliminated by fixing the bond length¹³⁰ or by regarding the hydrogen atom and its associated heavy atom as a united atom,¹³¹ which allows for larger time steps. The length scale is further increased when several united atoms are lumped together into one particle. This is especially beneficial for

large molecules, like polymers,^{132–134} phospholipids^{12,13,135} and proteins.¹³⁶ In these procedures the atomistic representation is mapped on a coarse-grained model, by defining a coarse-grained particle at the centre of mass of successive atoms. Atomistic simulations provide the averages for *e.g.* bond potential and bond angles, that serve as input parameters for the coarse-grained model. In a similar procedure¹³⁷ unconnected MD particles are coarse-grained to dissipative particles, for which the interparticle forces are derived from the underlying MD system, thus linking Dissipative Particle Dynamics (DPD)^{1,2} to MD.

In this chapter we will define a coarse-grained description of DPD within its own framework. In this way different scales are combined in one model, which will form the basis for the multiscale system in the next chapter. For the coarse-graining procedure we need not only the interaction parameters between coarse-grained particles, but also the interaction parameters between normal and coarse-grained particles. To achieve this we will first examine which properties must be kept constant upon coarse-graining. Next, we will derive expressions for these quantities as a function of the extent of coarse-graining. From these, we derive the interaction parameters. Finally, the coarse-graining procedure is tested over a large range of parameters.

5.2 Coarse-graining procedure for DPD

The main motivation for our coarse-graining procedure is the reduction of computational effort. In particle systems this can be accomplished by reducing the number of particles. The most obvious criterion is to require that the computation time decreases linearly with the number of particles. Or, in other words, the computational effort per particle should stay constant. In the DPD model, this effort is proportional to the number of interactions that a particle experiences with its neighbors. Other choices, for instance a smaller number of interactions upon coarse-graining, are possible as well, but they could violate the minimal number of interactions (*i.e.* 4π)¹¹ for DPD to work properly. For the same reason, requiring a constant number of interactions also has another significant advantage. It enables further coarse-graining of the coarse-grained description (within the limitations of the DPD model). In principle, this allows for hierarchal coarse-grained systems, bridging even larger length scale differences.

Apart from this computational objective, other requirements for the coarse-graining procedure are physical. The normal particles as well as the coarse-grained particles are meant to describe the same fluid with the same flow dynamics. Consequently, both model fluids should have an identical mass density, pressure, shear viscosity and temperature. Properties other than these are sacrificed to gain computational efficiency. For instance, a proper description of the compressibility and the diffusion coefficient are lost in the process.

In this section we will derive a coarse-grained description for the DPD model, that meets this set of objectives. Each subsection treats a property that must be kept constant upon coarse-graining. The starting point is to coarse-grain the normal particles by a factor of two, but this is not a necessary restriction. In the following, the parameters for the normal particles are subscripted with “0” and those for the coarse-grained with “1”. Similarly, the interaction parameters between normal particles carry the subscript “00” and those between coarse-grained particles “11”. For interactions between mixed particle pairs the subscript is either “10” or “01”, which are always identical to conserve momentum.

5.2.1 Mass density

Keeping the mass density ρ constant in the coarse-graining process is straightforward. When the number of coarse-grained particles is half the number of normal particles, their particle masses should be doubled, or $m_1 = 2 m_0$.

5.2.2 Number of interactions per particle

The cut-off radius and the number density determine the number of interactions per particle. For a normal particle with cut-off radius $r_{c,00}$ and number density n the total number of interactions is $\frac{4\pi}{3} n^2 r_{c,00}^3$. To derive the coarse-grained and mixed cut-off radii, consider a mixed system containing normal particles with number density n_0 and coarse-grained particles with number density n_1 . The normal particles interact within their cut-off radius $r_{c,00}$ and interactions between coarse-grained particles take place within $r_{c,11}$. Similarly, a normal particle interacts with coarse-grained particles within $r_{c,01}$ and a coarse-grained particle interacts with normal particles within $r_{c,10}$. The number of interactions per particle N_{int} is the sum of these interactions divided by the number of

particles. When we assume uniform mixing:

$$N_{\text{int}} = \frac{4\pi}{3} \left(\frac{n_0^2 r_{c,00}^3 + n_1^2 r_{c,11}^3 + n_0 n_1 r_{c,10}^3 + n_1 n_0 r_{c,01}^3}{n_0 + n_1} \right). \quad (5.1)$$

The number densities n_0 and n_1 are expressed as a mass fraction φ of coarse-grained particles in the system. This gives $n_0 m_0 = (1 - \varphi)\rho$ and $n_1 m_1 = \varphi\rho$, where ρ is the constant mass density. Using $n = \rho/m_0$, $m_1 = 2m_0$, and $r_{c,10} = r_{c,01}$ for reasons of symmetry, equation 5.1 converts to:

$$N_{\text{int}}(\varphi) = \frac{4\pi}{3} \left(\frac{(1 - \varphi)^2 n^2 r_{c,00}^3 + \frac{1}{4}\varphi^2 n^2 r_{c,11}^3 + \varphi(1 - \varphi)n^2 r_{c,10}^3}{(1 - \varphi)n + \frac{1}{2}\varphi n} \right). \quad (5.2)$$

When $\varphi = 0$, equation 5.2 reduces to $N_{\text{int}} = \frac{4\pi}{3} n r_{c,00}^3$, which is the number of particles a normal particle will interact with in a normal particle field. This is the number that should stay constant upon coarse-graining. For a fully coarse-grained particle field, when $\varphi = 1$, equation 5.2 becomes $N_{\text{int}} = \frac{4\pi}{3} \frac{1}{2} n r_{c,11}^3$, and thus $r_{c,11} = \sqrt[3]{2} r_{c,00}$. This is also an intuitive result; a particle that is twice as heavy will, with the same mass density, occupy twice the volume. Now the full expression for the number of interactions per particle (eq. 5.2) can be solved for the cut-off radius for mixed pairs, giving $r_{c,10} = \sqrt[3]{3/2} r_{c,00}$, independent of the mass fraction φ .

5.2.3 Pressure

Before we can determine the parameters to keep the pressure constant, we need an expression for the pressure to use in the derivation. The virial theorem for a DPD system¹¹ expresses the pressure p as:

$$p = n k_B T + \frac{2\pi}{3} n^2 \int_0^{r_c} r f^C(r) g(r) r^2 dr, \quad (5.3)$$

where n is the number density, k_B Boltzmann's constant, T the temperature, $f^C(r) = \alpha(1 - r/r_c)$ the conservative force with α the phase repulsion, and $g(r)$ the radial distribution function. When no conservative forces are present,

equation (5.3) reduces to the ideal gas law. It is obvious that the pressure of this ideal gas cannot be kept constant, because the number density is reduced in the coarse-graining procedure. Fortunately, the conservative force provides a way around this and is therefore required for the coarse-grained particles. Assuming the radial distribution function to be unity, equation (5.3) becomes:

$$p = n k_B T + \frac{\pi}{30} a n^2 r_c^4. \quad (5.4)$$

The actual pressures that are measured in simulations are, on average, 5% lower than this prediction, because the radial distribution function is not unity due to the conservative forces. This is also obvious when we compare the prefactor $\frac{\pi}{30} \approx 0.105$ to the variable $\alpha \approx 0.101$, measured by Groot and Warren.¹¹ However, in the derivation of the mixed and coarse-grained phase repulsions, the simple pressure expression (eq. 5.4) is sufficient. For this derivation, we return to our uniformly mixed system containing normal and coarse-grained particles. The pressure consists of the ideal and excess contributions of the different particles, in terms of the mass fraction φ of the coarse-grained particles:

$$\begin{aligned} p(\varphi) = & (1 - \varphi) n k_B T + \frac{\pi}{30} a_{00} (1 - \varphi)^2 n^2 r_{c,00}^4 \\ & + \frac{1}{2} \varphi n k_B T + \frac{\pi}{30} a_{11} \frac{1}{4} \varphi^2 n^2 r_{c,11}^4 \\ & + \frac{\pi}{30} a_{10} \varphi (1 - \varphi) n^2 r_{c,10}^4, \end{aligned} \quad (5.5)$$

where again number density $n = \rho/m_0$. Analogously to the previous subsection, the phase repulsion for the different interaction types are used to balance the overall pressure. The two extremes of the mass fraction ($\varphi = 0$, purely normal and $\varphi = 1$, purely coarse-grained) provide the phase repulsion between coarse-grained particles a_{11} :

$$a_{11} = \frac{60}{\pi} \frac{k_B T}{n r_{c,11}^4} + 4 a_{00} \frac{r_{c,00}^4}{r_{c,11}^4}. \quad (5.6)$$

The first term corrects the difference in ideal pressures of the normal and coarse-grained particles and the second corresponds to the scaling of the excess pressure. Note that if the phase repulsion of the normal particles a_{00} is zero (meaning they constitute an ideal DPD fluid), the coarse-grained phase

repulsion can still be calculated, but is non-zero. Using expression (5.6), equation (5.5) is solved for the phase repulsion for mixed interactions α_{10} :

$$\alpha_{10} = \frac{15}{\pi} \frac{k_B T}{n r_{c,10}^4} + 2 \alpha_{00} \frac{r_{c,00}^4}{r_{c,10}^4}, \quad (5.7)$$

which is again independent of the mass fraction φ .

5.2.4 Shear viscosity

To simulate flow dynamics with coarse-grained particles, the procedure should not alter the viscosity. Marsh *et al.* derived an expression for the shear viscosity of an ideal DPD fluid.⁴ It consists of a dissipative and kinetic contribution, η^D and η^K :

$$\eta^D = \frac{2\pi}{1575} n^2 \gamma r_c^5, \quad (5.8)$$

$$\eta^K = \frac{45}{4\pi} \frac{m k_B T}{\gamma r_c^3}, \quad (5.9)$$

where γ is the friction coefficient. Quantitatively, these are poor predictions for the actual viscosity of an ideal DPD fluid, but they do describe the trends well. The missing conservative contribution is determined by the off-diagonal terms of the virial pressure tensor.³ As the pressure itself is kept constant upon coarse-graining, we assume the conservative contribution will be as well. In general the kinetic contribution in a liquid is very small compared to the other contributions. So, we will use the ideal dissipative contribution (eq. 5.8) to determine the coarse-grained and mixed parameters. In a mixed system this contribution depends on the composition φ :

$$\eta^D(\varphi) = \frac{2\pi}{1575} \left((1-\varphi)^2 n^2 \gamma_{00} r_{c,00}^5 + \frac{1}{4} \varphi^2 n^2 \gamma_{11} r_{c,11}^5 + \varphi(1-\varphi) n^2 \gamma_{10} r_{c,10}^5 \right). \quad (5.10)$$

Subsequently solving for γ_{11} and γ_{10} gives:

$$\gamma_{11} = 4\gamma_{00} \frac{r_{c,00}^5}{r_{c,11}^5}, \quad (5.11)$$

$$\gamma_{10} = 2\gamma_{00} \frac{r_{c,00}^5}{r_{c,10}^5}. \quad (5.12)$$

5.2.5 Temperature

The fluctuation-dissipation theorem,² formulated by Español and Warren, relates the noise amplitude to the friction coefficient, $\sigma^2 = 2\gamma k_B T$. It balances the energy supply to and removal from the system to equilibrate it at the set temperature T . For the coarse-grained and mixed noise amplitudes, σ_{11} and σ_{10} , this means:

$$\sigma_{11} = \sqrt{2\gamma_{11} k_B T}, \quad (5.13)$$

$$\sigma_{10} = \sqrt{2\gamma_{10} k_B T}. \quad (5.14)$$

5.3 Computational details

To test the coarse-graining procedure, we measure the pressure and viscosity of systems with a coarse-grained mass fraction of $\varphi = 0$ (purely normal), $\varphi = 1$ (purely coarse-grained) and $\varphi = 0.5$ (mixed). The normal particles have a cut-off radius $r_{c,00}$ and particle mass m_0 of unity. We choose the base case parameters (in DPD units¹¹) for the normal particles in the liquid regime: mass density $\rho = 6$, temperature $k_B T = 0.5$ and noise amplitude $\sigma_{00} = 4.5$. The phase repulsion $\alpha_{00} = 6.25$ follows from the expression: $\alpha = 75k_B T/\eta$,¹¹ that approximates the compressibility of water, when one DPD particle models one water molecule.¹² We examine how the pressure and viscosity depend on the density, noise amplitude, temperature and phase repulsion. While one parameter is varied, the other three are set at their base case values. The

simulations are performed in a periodic box of volume 12^3 and are run for 400 time units after an equilibration time of 100 time units with a time step of $\Delta t = 0.01$. We determine the pressure via the virial expression, and the viscosity via the periodic Poiseuille flow method.¹³⁸

5.4 Results and discussion

In this section we will examine whether the coarse-graining procedure fulfills the requirements that were formulated at the start of section 5.2. The mass density is constant by construction and the temperature is kept constant by the DPD thermostat. The number of interactions per particle is related to the performance that will be discussed in section 5.4.3. That leaves the pressure and viscosity to study.

5.4.1 Pressure of the coarse-grained system

Figure 5.1 shows the measured pressures of a system containing only normal particles, a system containing only coarse-grained particles and a mixed system with a coarse-grained mass fraction of $\varphi = 0.5$. The mass density, noise amplitude, temperature and phase repulsion are varied around the base case parameters of $\rho = 6$, $\sigma_{00} = 4.5$, $k_B T = 0.5$ and $a_{00} = 6.25$. The agreement of the measurements is excellent. As expected from equation 5.4, the pressure depends quadratically on the mass density, linearly on the temperature and the phase repulsion, and is independent of the noise amplitude.

Scaling the phase repulsion inevitably changes the structure of the fluid, characterized by the radial distribution function. Figure 5.2 compares the radial distribution functions of the normal, mixed and coarse-grained interactions. Despite the different cut-off radii, the shapes are similar. For a larger cut-off radius, the excluded volume at small distances increases, which is directly related to the larger phase repulsion.

5.4.2 Viscosity of the coarse-grained system

The viscosities are measured in a purely normal, a purely coarse-grained and a mixed system (with $\varphi = 0.5$). In the simulations the noise amplitude is

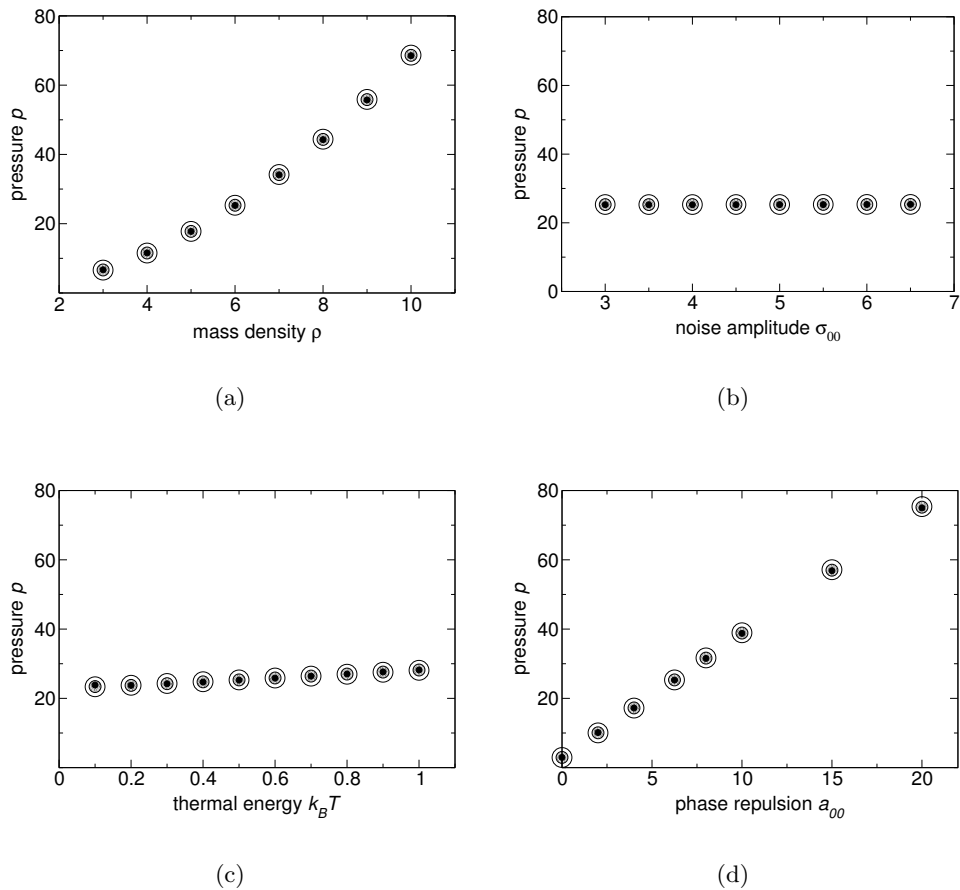


Figure 5.1: Pressure of normal (•), coarse-grained (○) and mixed system (●) at $\rho = 6$, $\sigma_{00} = 4.5$, $k_B T = 0.5$ and $a_{00} = 6.25$ as a function of (a) mass density, (b) noise amplitude, (c) temperature and (d) phase repulsion; error bars are smaller than smallest symbol.

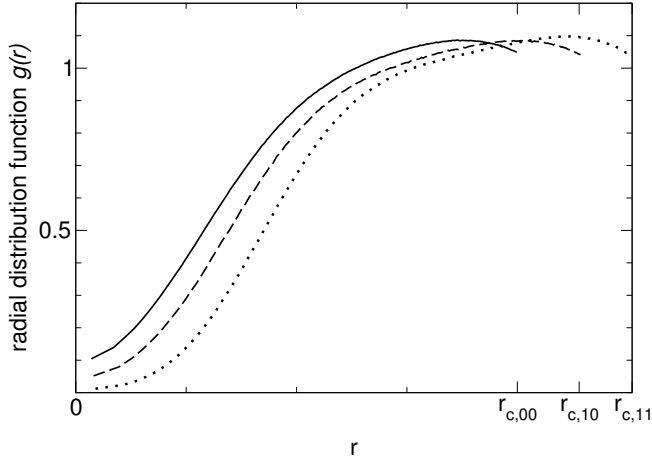


Figure 5.2: Radial distribution function of normal (—), mixed (– –) and coarse-grained (· · ·) interactions for mass density $\rho = 6$, temperature $k_B T = 0.5$ and phase repulsion $\alpha_{00} = 6.25$.

constant, instead of the friction coefficient that is used in the derivation. The two parameters are coupled through the fluctuation-dissipation theorem, $\sigma^2 = 2\gamma k_B T$, which modifies the dissipative contribution from equation 5.8 to:

$$\eta^D = \frac{\pi}{1575} \frac{n^2 \sigma^2 r_c^5}{k_B T}. \quad (5.15)$$

Consistent with this relation, the viscosity increases quadratically with the density (fig. 5.3(a)) and the noise amplitude (fig. 5.3(b)). Conversion of the noise amplitude into the friction coefficient yields a linear viscosity dependence, which confirms the fluid is in the liquid regime. Figure 5.3(c) shows that the viscosity is inversely proportional to the temperature. Finally, the effect of the phase repulsion is not included in the derivation of the ideal viscosity contributions, but as the conservative contribution is coupled to the pressure, the linear trend in figure 5.3(d) is not surprising. Note that at a phase repulsion of zero, *i.e.* for an ideal fluid of normal particles, the viscosity for the coarse-grained and the mixed system coincide reasonably well. Overall, the coarse-grained viscosity exceeds the normal viscosity by around 5%, while the viscosity of the mixed system lies in between. The differences are larger at higher values, where the error of the viscosity prediction becomes more pronounced. Consequently

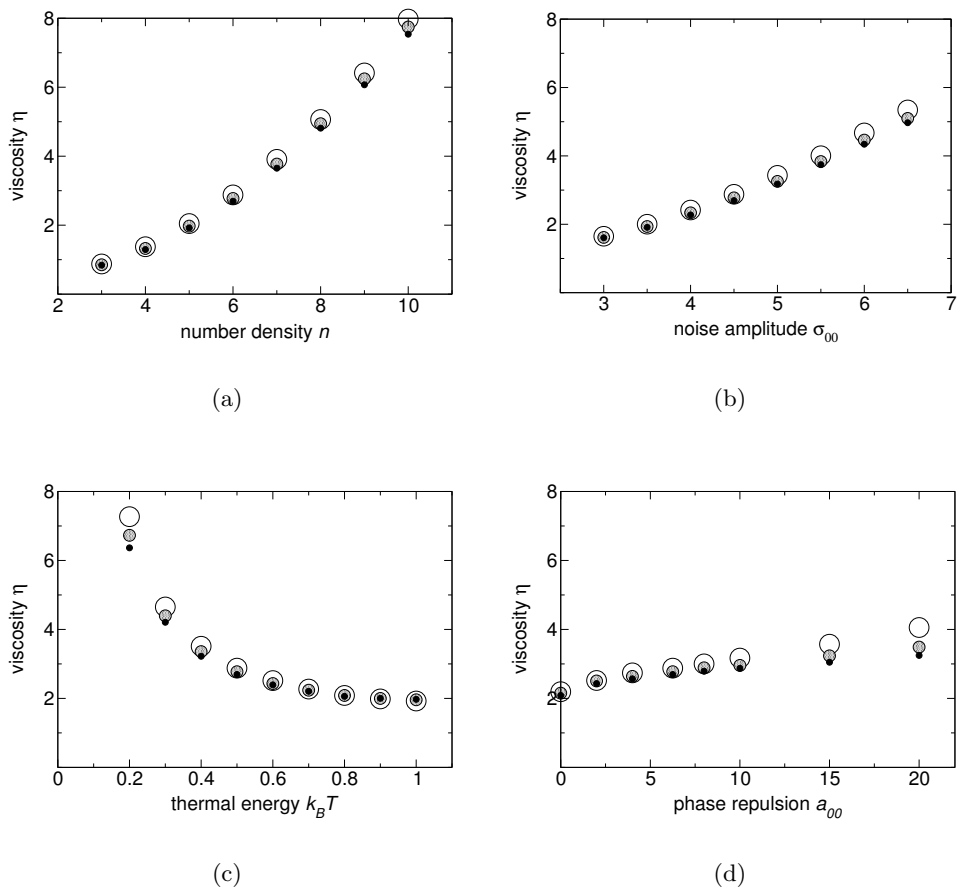


Figure 5.3: Viscosity of normal (●), coarse-grained (○) and mixed system (◐) at $\rho = 6$, $\sigma_{00} = 4.5$, $k_B T = 0.5$ and $a_{00} = 6.25$ as a function of (a) number density, (b) noise amplitude, (c) temperature and (d) phase repulsion; error bars are smaller than smallest symbol.

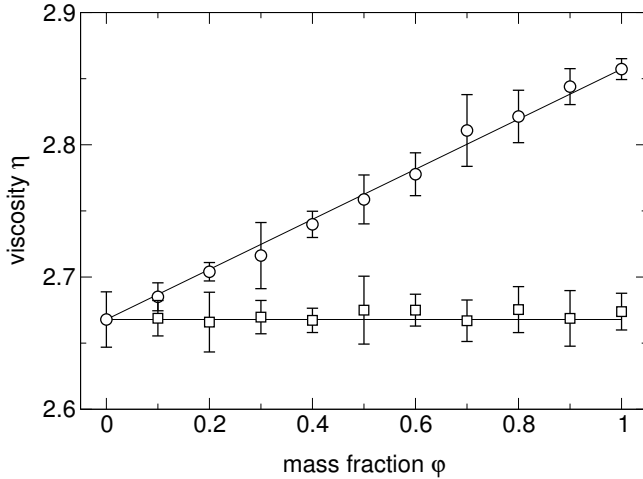


Figure 5.4: Uncorrected (○) and corrected (□) viscosities as a function of mass fraction ϕ of coarse-grained particles, at $\rho = 6$, $\sigma_{00} = 4.5$, $k_B T = 0.5$ and $\alpha_{00} = 6.25$. Lines are drawn to guide the eye.

the divergence of the coarse-grained from the normal viscosities is larger. To equalize the viscosities, the friction coefficients γ_{11} and γ_{10} are used as tuning parameters, by multiplying them with the correction factors ζ_{11} and ζ_{10} respectively:

$$\gamma_{11}^* = \zeta_{11} \gamma_{11}, \quad (5.16)$$

$$\gamma_{10}^* = \zeta_{10} \gamma_{10}, \quad (5.17)$$

where the asterisks denote the effective friction coefficients. First the friction coefficient for the coarse-grained particles must be corrected. Requiring that $\gamma_{11}^* \geq \gamma_{00}$, the minimum value for the correction factor is $\zeta_{11} > 0.79$. For our base case system ($\rho = 6$, $k_B T = 0.5$, $\sigma_{00} = 4.5$ and $\alpha_{00} = 6.25$) we find $\zeta_{11} = 0.91$, which is in fact valid over a large range of parameters. With this value we can tune the mixed friction coefficient, that needs to be in the range $\gamma_{00} \leq \gamma_{10}^* \leq \gamma_{11}^*$. The corresponding correction factor has to be $0.98 < \zeta_{10} < 1.24 \zeta_{11}$. However, we find a value of $\zeta_{10} = 0.96$ for our system. In other words, the mixed friction coefficient γ_{10}^* should be smaller than the normal

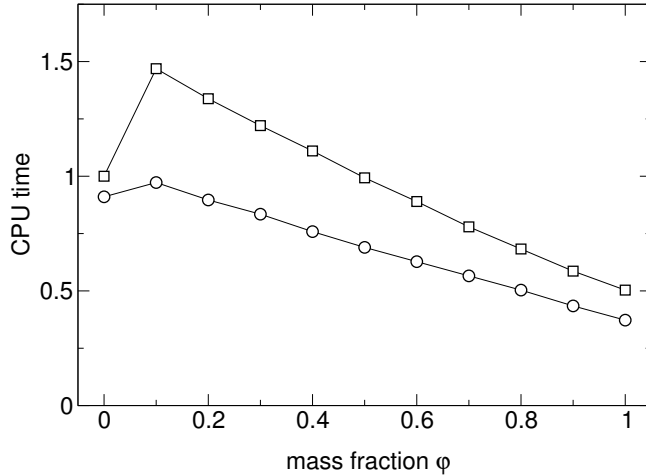


Figure 5.5: Normalized CPU time as a function of mass fraction ϕ of coarse-grained particles, using a cell list ($-\square-$) and a combination of cell and Verlet list ($-\circ-$).

friction coefficient γ_{00} to compensate for the viscosity difference. Figure 5.4 shows the uncorrected viscosities (based on γ_{10} and γ_{11}) and the corrected viscosities (based on γ_{10}^* and γ_{11}^*) for the base case parameters as a function of the mass fraction ϕ of the coarse-grained particles. The average values and standard deviations follow from 4 simulations of 200 time units at a time step of $\Delta t = 0.01$. As expected, the uncorrected viscosities increase linearly. The corrected viscosities agree to within 1% with the viscosity of the normal system (at $\phi = 0$). Therefore, we will use the low value of the mixed friction coefficient, although it is counterintuitive.

5.4.3 Performance

Having derived and validated all interaction parameters, we return to the original objective, the reduction of computational effort. A constant number of interactions per particle should lead to a computation time proportional to the number of particles. So, when we decrease the number of particles with a factor of two, the CPU time should also be reduced by half. However, this depends strongly on how the calculation of the interactions is implemented. Figure 5.5 shows the measured computation time as a function of the mass fraction of coarse-grained particles, using a cell list and a combination of cell and Verlet

list.

When we use a cell list, the cell size for the normal system ($\chi = 0$) is $r_{c,00}^3$ and for the fully coarse-grained system ($\chi = 1$) $r_{c,11}^3$. When we compare the values for these fractions in figure 5.5, the CPU time indeed scales with a factor of two, but the intermediate mass fractions do not show a monotonously decreasing line. The reason is that for a mixed system the cell size is dictated by the largest interaction radius possible, *i.e.* $r_{c,11}^3$. Even in a system containing mainly normal particles, the cells need to be large to include the few coarse-grained interactions. This means that many particles outside the interaction radius are considered. The fraction of these non-interacting particle pairs decreases with increasing χ , resulting in the non-linearity of the curve from $\chi = 0.1$ on. Only at a mass fraction higher than $\chi = 0.5$ the CPU time is smaller than for the normal system. As the coarse-graining procedure is intended for mixed systems, it is not computationally profitable when a cell list is used.

For the combination of cell and Verlet list, we need to determine the optimal values for the Verlet radius r_v . For the normal particles we find $r_{v,00} = 1.3 r_{c,00}$, for the coarse-grained particles $r_{v,11} = 1.2 r_{c,11}$ and for the mixed interactions $r_{v,10} = 1.2 r_{c,10}$. Note that the cells for the cell list must have the size of the largest r_v (which is not necessarily $r_{v,11}$, due to the system size effect). The performance of this combined implementation in figure 5.5 is much better than the cell list performance. When we compare the CPU time at mass fractions $\chi = 0$ and $\chi = 1$, it reduces by more than the expected factor of two. This is because the heavier coarse-grained particles have a smaller thermal velocity and are therefore slower to fulfill the update criterion. Due to the use of the cell list, the CPU requirement is not monotonously decreasing, but the effect is no longer counter-effective. Using the combined lists, the coarse-graining procedure achieves the desired reduction in computation time.

5.5 Conclusion

In this chapter we introduced a coarse-graining procedure for the DPD model to enhance computational efficiency. This scheme aims to keep the mass density, pressure, temperature and shear viscosity constant, while sacrificing the compressibility and diffusivity. For the particle mass, cut-off radius and phase repulsion straightforward relations are derived. The friction coefficient on the other hand, needs to be tuned to the shear viscosity. The corresponding noise

particles	$\bullet \bullet$	$\bullet \circ$	$\circ \circ$
mass	m_0		$m_1 = 2 m_0$
cut-off radius	$r_{c,00}$	$r_{c,10} = \sqrt[3]{3/2} r_{c,00}$	$r_{c,11} = \sqrt[3]{2} r_{c,00}$
phase repulsion	a_{00}	$a_{10} = \frac{15}{\pi} \frac{k_B T}{n r_{c,10}^4} + 2 a_{00} \frac{r_{c,00}^4}{r_{c,10}^4}$	$a_{11} = \frac{60}{\pi} \frac{k_B T}{n r_{c,11}^4} + 4 a_{00} \frac{r_{c,00}^4}{r_{c,11}^4}$
friction coefficient	γ_{00}	$\gamma_{10} = 2 \gamma_{00} \frac{r_{c,00}^5}{r_{c,10}^5}$	$\gamma_{11} = 4 \gamma_{00} \frac{r_{c,00}^5}{r_{c,11}^5}$
effective friction coeff.		$\gamma_{10}^* = \zeta_{10} \gamma_{10}$	$\gamma_{11}^* = \zeta_{11} \gamma_{11}$
noise amplitude	σ_{00}	$\sigma_{10}^* = \sqrt{2 \gamma_{10}^* k_B T}$	$\sigma_{11}^* = \sqrt{2 \gamma_{11}^* k_B T}$

Table 5.1: Parameters for the coarse-grained and mixed interactions in DPD model

amplitude is found through the dissipation-fluctuation theorem. The interaction parameters are recapitulated in table 5.5. The computational effort is roughly proportional to the number of particles, provided that a combination of a cell list and Verlet list is used in the simulations.

Combining length scales in Dissipative Particle Dynamics

6.1 Introduction

Particle models often involve computationally expensive simulations. Especially models that use relatively high particle densities, for instance Dissipative Particle Dynamics (DPD), suffer from long simulation times, compared to their continuous counterparts. Therefore, it can be beneficial to make a distinction between regions that require detailed simulation, and bulk areas that are of less interest. Such multiscale systems involving particle models¹³⁹ are discussed below.

One possible approach is to combine different models through direct interaction. A suitable framework is the fluid particle model, that gives a well defined meaning to the scale of a mesoscale particle.¹⁴⁰ This model describes a range of scales with a general algorithm. By changing the interaction terms one can arrive at molecular dynamics,¹⁰ dissipative particle dynamics^{1,2} or smoothed particle hydrodynamics.^{113,141} Dzwinel *et al.* used this algorithm to combine different models in a three-level model to simulate colloidal structures.⁷⁸ Stochastic rotational dynamics (SRD)¹¹⁴ also combines different scales. Identical mesoscale particles in a collision volume undergo a multiparticle collision that rotates their velocities by a random angle. The model conserves mass, momentum and energy and satisfies the Navier-Stokes equations. These mesoscale particles can, for instance, describe solvent particles, while the solute molecule is modeled with MD.^{142,143}

In another approach, models are coupled by an overlap region, where both models coexist. For instance, the combination of molecular dynamics with a continuum description^{144–147} bridges large scale differences. Flekkøy *et al.* base their hybrid model on the symmetric exchange of fluxes of conserved quantities.¹⁴⁷ At the particle side of the overlap region, the fluxes are measured and imposed on the continuum. Analogously, the continuum fluxes are imposed on the particles.

A third approach combines different scales in one mesoscopic fluid particle model.^{148,149} It is based on a Voronoi tessellation to coarse-grain the underlying system.¹³⁷ In this tessellation procedure the physical space is assigned to the nearest Voronoi centre. Each of these fluid particles is a thermodynamic system in itself, that exchanges mass, momentum and energy with its neighbours. By changing the resolution of the Lagrangian grid, a multiscale description emerges by construction.¹⁵⁰ The forces that govern the evolution over time are either based on ensemble averages of the microscopic level,¹⁴⁸ or on the stochastic continuum equations of fluctuating hydrodynamics.¹⁴⁹ Although derived on a different basis, the dynamics of the models are similar.¹⁵¹

In this chapter we propose a model to combine different length scales in DPD. As a particle analog to grid refinement in finite-element methods, its main objective is to increase computational efficiency. Similar to the Voronoi models, one set of model equations describes all scales, which makes simultaneous simulation straightforward. But instead of involving a range of length scales, here we restrict our model to contain only two different length scales. The first scale is the “classic” DPD description^{1,2,11} and the second scale is a coarse-grained version that was derived in the previous chapter. The different particle domains are coupled through an overlap region. In contrast to the particle–continuum models, they can exchange momentum through direct interaction, because both scales are treated within the DPD framework. We first describe schemes for the local refining and coarse-graining and examine the size of the overlap region. Next, we present the results of the multiscale model in equilibrium and under flow conditions.

6.2 Combination of length scales in one system

The normal and coarse-grained particles that were defined in the coarse-graining procedure are now combined into one system. They are placed in the simula-

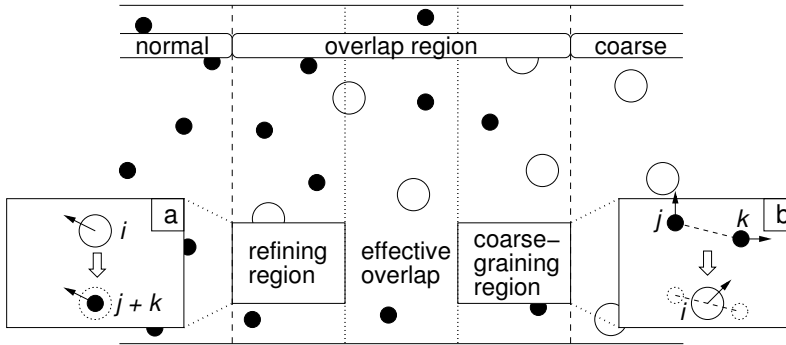


Figure 6.1: Schematic representation of simulation domain with overlap region; refining region for (a) local refining and coarse-graining region for (b) local coarse-graining.

tion domain, separated by an overlap region. This region is split into a refining region, a coarse-graining region and a remaining effective overlap region. Figure 6.1 shows this set up schematically. Coarse-grained particles that enter the refining region are locally refined to two normal particles. Similarly, two normal particles in the coarse-graining region are coarse-grained into one particle. We now turn to the refining and coarse-graining schemes and the size of the effective overlap region.

6.2.1 Local refining

Whenever a coarse-grained particle i enters the refining region, it is transformed into two normal particles, j and k (see fig. 6.1(a)). These take on the position \vec{r} and velocity \vec{v} of the original coarse-grained particle, half the mass m and hence half the momentum \vec{p} :

$$\begin{aligned}
 m_j &= m_k = \frac{1}{2} m_i, \\
 \vec{r}_j &= \vec{r}_k = \vec{r}_i, \\
 \vec{v}_j &= \vec{v}_k = \vec{v}_i, \\
 \vec{p}_j &= \vec{p}_k = \frac{1}{2} \vec{p}_i.
 \end{aligned}
 \tag{6.1}$$

The new particles are placed on top of each other, and the velocity they adopt is lower than the thermal velocity they should have. So, although it does conserve momentum, this scheme does not equilibrate nor thermalize the new particles. Fortunately, the fraction of refined particles is sufficiently small to neglect the influence on the system. Therefore, we leave it to the DPD thermostat to equilibrate and thermalize the new particles in the next time steps.

To avoid complications in the managing of Verlet lists, the local refinings only take place when the Verlet lists are updated. This makes the refining region essential to prevent coarse-grained particles from leaving the overlap region without being refined. The region should be wide enough to accommodate the displacement of the particles to be refined, *i.e.* the average time between Verlet list updates multiplied by their average velocity. Setting the width to $r_{c,00}$ amply satisfies this condition.

6.2.2 Local coarse-graining

When a normal particle j enters the coarse-graining region, another needs to be selected to join the coarse-graining (see fig. 6.1(b)). This coarse-graining “partner”, particle k , is not necessarily located in the coarse-graining region itself. The coarse-graining only takes place if particles j and k are less than the normal interaction radius apart, otherwise it is postponed to the next coarse-graining step. When the distance criterion is satisfied, particles j and k are coarse-grained into particle i , by averaging the positions \vec{r} and velocities \vec{v} and summing the masses m and momenta \vec{p} :

$$\left. \begin{aligned} m_i &= m_j + m_k \\ \vec{r}_i &= \frac{1}{2} (\vec{r}_j + \vec{r}_k) \\ \vec{v}_i &= \frac{1}{2} (\vec{v}_j + \vec{v}_k) \\ \vec{p}_i &= \vec{p}_j + \vec{p}_k \end{aligned} \right\} \text{if } |\vec{r}_j - \vec{r}_k| \leq r_{c,00}. \quad (6.2)$$

Again, this scheme conserves momentum, but leaves the thermalizing of the new coarse-grained particle to the DPD thermostat. The local coarse-graining also takes place when the Verlet lists are updated. This simplifies the Verlet list bookkeeping.

Particle k can be selected in different ways. One possibility is to introduce an extra Verlet list that stores only the normal particles around j . When the latter is locally coarse-grained it will select the nearest neighbour from the extra Verlet list. In another approach only the nearest normal neighbour is stored at the time of updating the Verlet list. In about 3% of the cases, this is not the nearest particle anymore when it is used for coarse-graining, just before the next update of the Verlet lists. However, the selection of the nearest neighbour is not a strict requirement; a “near” neighbour will do as well. Since it requires less memory, the latter approach is adopted.

If the distance criterion is not satisfied, both particles will continue unaltered until the next coarse-graining step. When a normal particle reaches the end of the overlap region, it will be reflected to prevent it from leaving. Changing its velocity would violate momentum conservation, therefore only the particle position is altered. The width of the coarse-graining region is also set to $r_{c,00}$. This is narrow enough to prevent the normal particles from drifting apart and wide enough to keep the number of reflections low.

6.2.3 Size of effective overlap region

The widths of the refining and coarse-graining regions are set to $r_{c,00}$, but the optimal size of the effective overlap region still needs to be determined. To this end the density profile is shown in figure 6.2 for different sizes of the effective overlap region. In this figure the normal particles are situated on the lefthand side and the coarse-grained particles on the righthand side. The space between the dotted lines indicates the overlap region. When the overlap width is small, the overall density is distorted from uniformity. This is because the interaction parameters between normal and coarse-grained particles are derived from a perfectly mixed system. In the overlap region, however, the normal and coarse-grained particles are physically separated, which unbalances the forces. As a result the coarse-grained particles experience a lower pressure, causing them to pile up on one side. On the other side the normal particles experience a higher pressure, resulting in a density decrease. This inequality becomes less pronounced when the change in partial densities is more gradual, *i.e.* at larger overlap. We will use an effective overlap region of width $4r_{c,00}$, where the deviation in the density profile is 1% at most.

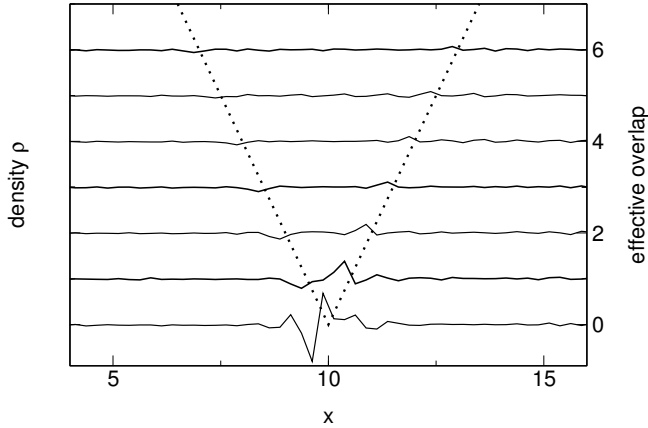


Figure 6.2: Density profiles at mass density $\rho = 6$ for different widths of overlap region, for clarity the density profiles are vertically shifted according to the effective overlap; the effective overlap is also indicated by the horizontal space between the dotted lines.

6.3 Computational details

To test the combined system, simulations are performed both in equilibrium and under different flow conditions. For both systems the mass density is $\rho = 6$ and the thermal energy is $k_B T = 0.5$. The base case parameters for the normal particles are particle mass $m_0 = 1$, cut-off radius $r_{c,00} = 1$, phase repulsion $\alpha_{00} = 6.25$ and noise amplitude $\sigma_{00} = 4.5$. The corresponding parameters for the coarse-grained particles are $m_1 = 2$, $r_{c,11} = 1.26$, $\alpha_{11} = 10.55$ and $\sigma_{11}^* = 4.8$, and the parameters for mixed interactions are $r_{c,10} = 1.14$, $\alpha_{10} = 7.51$ and $\sigma_{10}^* = 4.4$. The asterisks refer to the effective noise amplitudes (see previous chapter). The mass fraction of coarse-grained particles is $\varphi = 0.5$. The simulation domain is a cubic periodic box of size 20 in all directions. Two overlap regions are placed parallel to the yz -plane, around $x = 0$ and $x = 10$. They have an effective overlap of $4 r_{c,00}$, excluding a refining and coarse-graining region of width $r_{c,00}$ each. The simulation time is 200 time units at a time step of $\Delta t = 0.01$ after equilibration for 50 time units. For the simulations under shear flow the shear velocity is $V_{\text{shear}} = 0.5$, using Lees-Edwards boundary conditions.³¹ For the Poiseuille flow simulations the body force constant of $g = 0.02$ is applied in opposite directions to attain a periodic Poiseuille flow profile.¹³⁸

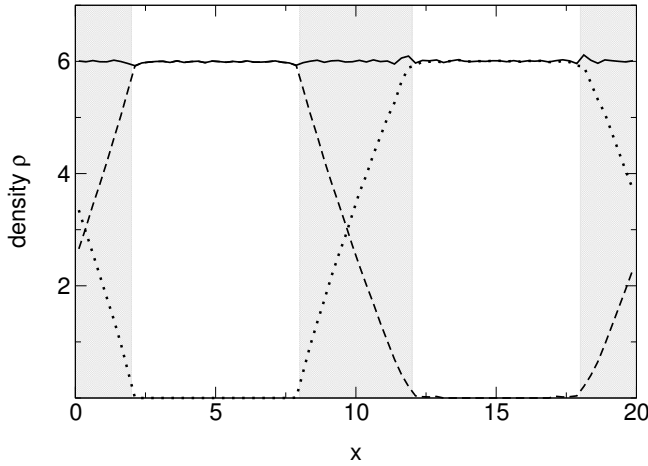


Figure 6.3: Density profile for combined system; overall mass density (—), consisting of mass density of normal particles (--) and mass density of coarse-grained particles (···). The shaded areas indicate the effective overlap regions.

The force calculations are implemented using a combination of a cell list and Verlet list. The optimal cut-off radii for the Verlet lists are $r_{v,00} = 1.3 r_{c,00}$, $r_{v,11} = 1.2 r_{c,11}$ and $r_{v,10} = 1.2 r_{c,10}$. The Verlet list is constructed from a cell list that consists of cells of size $r_{v,11}^3$. Updating takes place roughly every five time steps.

6.4 Results and discussion

In this section we will first examine the properties of the combined system in equilibrium. Next, the velocity profiles are shown for shear flow and Poiseuille flow, where the flow is directed either parallel or perpendicular to the overlap regions.

6.4.1 Combined system in equilibrium

Figure 6.3 shows the mass density profile for the combined system. The overall density shows anomalies at the edges of the overlap region, that were already discussed in section 6.2.3. In the coarse-graining regions ($12 \leq x \leq 13$ and

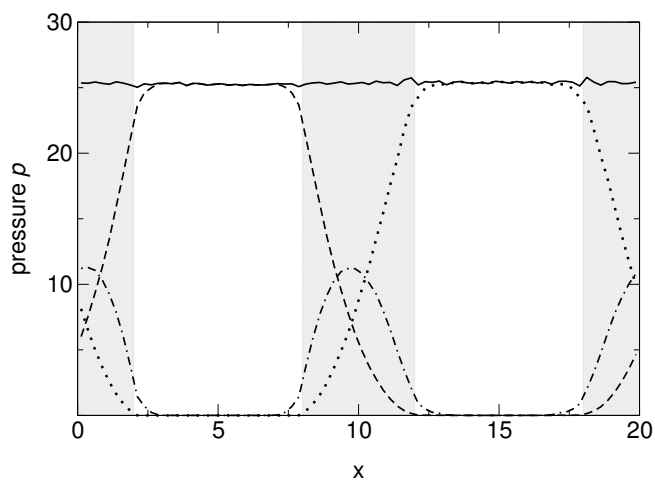


Figure 6.4: Pressure profile for combined system; overall pressure (—), consisting of ideal and virial pressure of normal particles (— —), ideal and virial pressure of coarse-grained particles (· · ·) and virial pressure of mixed particle interactions (— · · —). The shaded areas indicate the effective overlap regions.

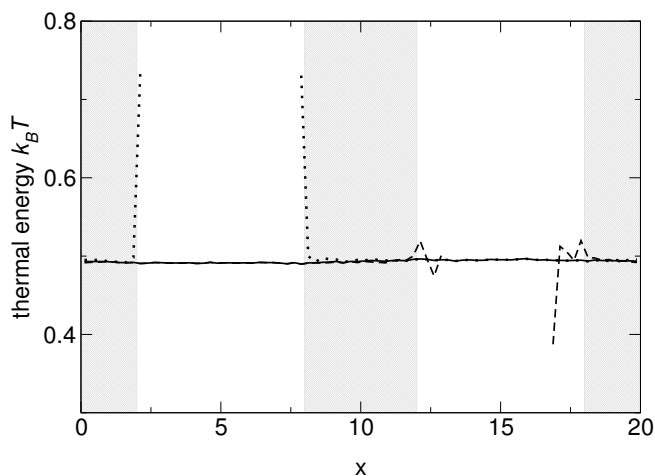


Figure 6.5: Temperature profile for combined system; overall temperature (—), temperature of normal particles (— —) and temperature of coarse-grained particles (· · ·). The shaded areas indicate the effective overlap regions.

$17 \leq x \leq 18$) a small number of normal particles is observed. These are the particles that did not instantly find a coarse-graining partner. Their fraction is too small to have any effect on the system. In the refining regions ($2 \leq x \leq 3$ and $7 \leq x \leq 8$) the presence of the coarse-grained particles that drifted inside between refining steps, is imperceptible on the scale of the figure.

In figure 6.4 the overall pressure is the sum of three partial pressures. The partial pressure of the normal particles consists of the ideal pressure of the normal particles and the virial term of normal interactions. The contribution of the coarse-grained particles is defined analogously. The third partial pressure only consists of the virial term of the mixed interactions, which is at a maximum in the middle of the overlap regions. The three partial pressures add up to a fairly constant overall pressure. The pressure is directly related to the density, causing the similarity between their overall profiles.

Finally, the temperature profile is shown in figure 6.5. The overall temperature is constant throughout the simulation domain. It is about 2% below the set temperature of $k_B T = 0.5$, which is a known artefact of the simple DPD algorithm.²² The temperatures of the separate particle types exhibit peaks at the edges of the overlap regions. These are caused by the small numbers of coarse-grained particles in the refining region, and normal particles in the coarse-graining region. In particular, the coarse-grained particles are excited by the fast moving normal particles around them, which is an artefact of the refining process. However, the fraction of particles with a deviating temperature is sufficiently small to leave the overall temperature unaffected.

6.4.2 Combined system under flow conditions

In the flow simulations we test two different flow types, Poiseuille flow and shear flow, on two different flow directions, parallel and perpendicular to the overlap regions. After equilibration, at time $t = 0$, the body force is switched on for the Poiseuille flow, or the Lees-Edwards boundaries are initiated for the shear flow. The resulting velocity profiles are compared to the profiles of a system containing only normal particles at time $t = 25$ and time $t = 75$, representing a nonstationary and nearly stationary situation. Figure 6.6 shows the velocity profiles of both systems at both times. In general, the velocities of the combined and the normal system coincide. The agreement of the nonstationary profiles (at time $t = 25$) demonstrates that the viscosity of the combined system is identical to the normal viscosity. The parallel flow profiles (fig. 6.6(a))

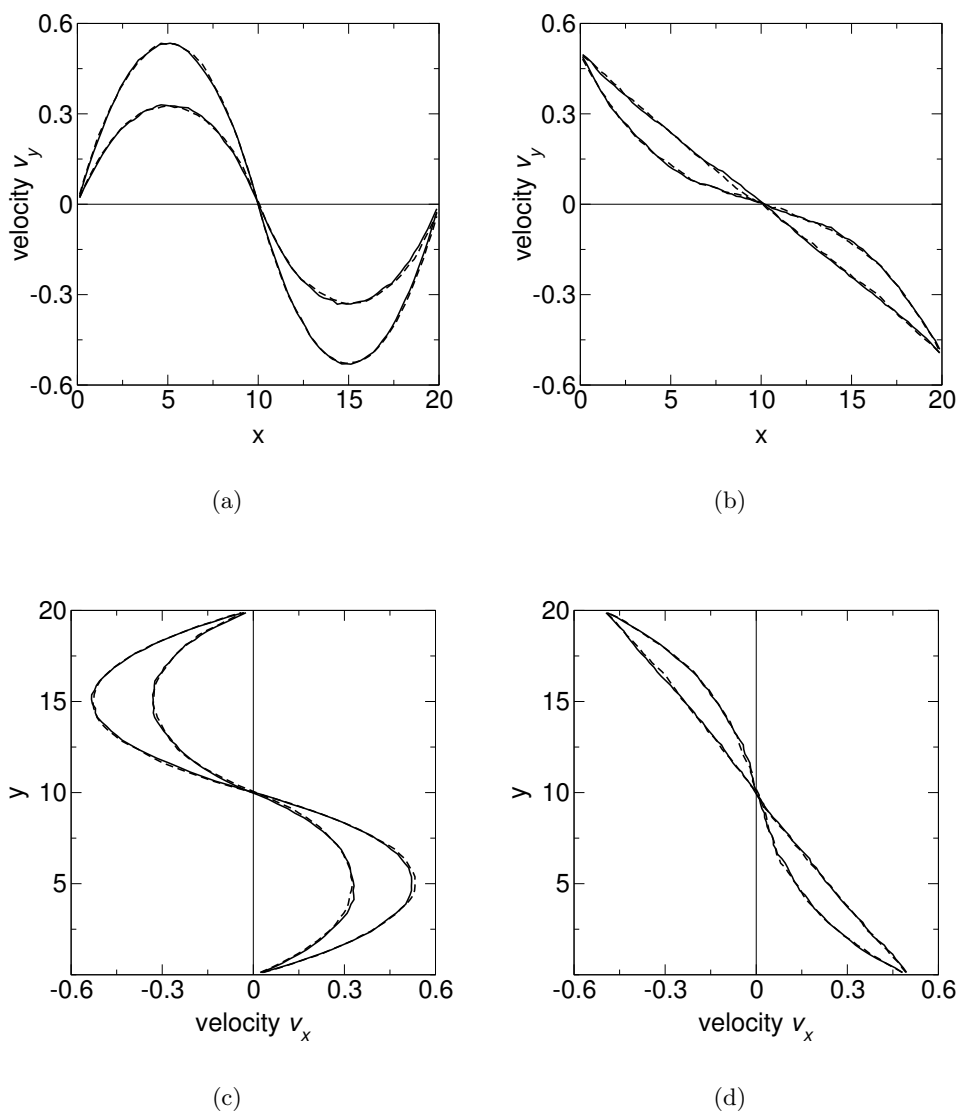


Figure 6.6: Velocity profiles of combined system (—) and normal system (---) at $t = 25$ and $t = 75$ for (a) parallel Poiseuille flow, (b) parallel shear flow, (c) perpendicular Poiseuille flow and (d) perpendicular shear flow.

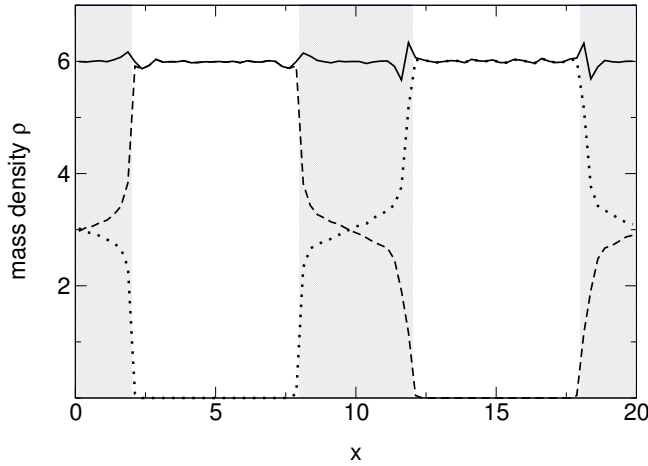


Figure 6.7: Density profile for combined system under perpendicular Poiseuille flow; overall mass density (—), consisting of mass density of normal particles (— —) and mass density of coarse-grained particles (· · ·). The shaded areas indicate the effective overlap regions.

and fig. 6.6(b)) do not show statistically significant distortions at the overlap regions. This means that the density deviations at the edges of the overlap (fig. 6.3) do not influence the local velocity. The symmetry of the profiles also indicates that the viscosity of the coarse-grained region (at the righthand side) is identical to the viscosity in the normal region (at the lefthand side).

The most critical tests for the combined system are the perpendicular flows. The maxima in the velocity profiles (fig. 6.6(c) and fig. 6.6(d)) are equal to the maximal velocities reached by the normal system. Thus, the local coarse-graining scheme is able to handle the large number of particles that enter the coarse-graining region at a relatively high velocity. However, as a secondary consequence of forcing the particles through the overlap region, the partial densities change as a step function rather than linearly (compare fig. 6.7 to fig. 6.3). This results in greater distortions in the density profile, as explained in section 6.2.3. But despite their magnitude, the density variations do not perturb the velocity profile.

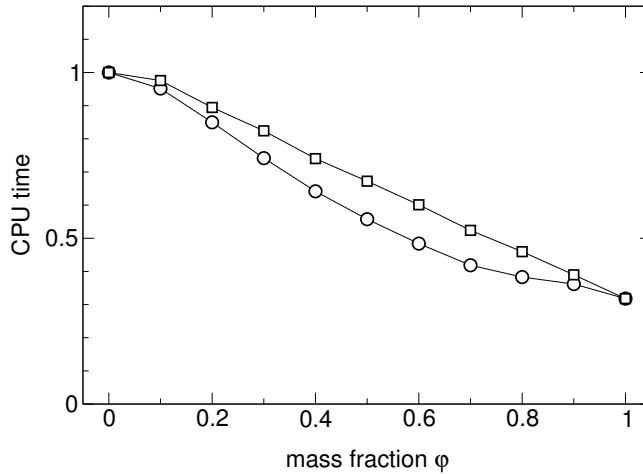


Figure 6.8: Computation time for the combined system with overlap region ($-\circ-$) and without overlap region ($-\square-$) as a function of the mass fraction of coarse-grained particles.

6.4.3 Performance

The previous sections showed that the combined system adequately reproduces the properties and behaviour of a normal system. Now we return to the original objective, to gain computational efficiency. To this end, the computation time is measured as a function of the mass fraction of coarse-grained particles. In figure 6.8 the results for the combined system with overlap, and the fully mixed system without overlap are shown. When we compare the CPU time at mass fractions $\phi = 0$ and $\phi = 1$, it reduces by a factor of three, instead of the expected factor of two. This is because the heavier coarse-grained particles have a smaller thermal velocity and are therefore slower to fulfill the Verlet list update criterion.

Introducing an overlap region influences the computation time in two ways. First, an overhead is created by the local refining and coarse-graining in the combined system. In equilibrium, the mass flux of locally coarse-grained particles is equal to the mass flux of locally refined particles. Under our simulation conditions this flux is in the order of 10^{-3} normal particles per unit area per time step. In the parallel flow simulations the fluxes are comparable to those in equilibrium, independent of the parallel velocity. It is obvious that the

perpendicular flows raise the coarse-graining and refining fluxes considerably, depending on the maximum velocity. For our flow parameters the flux is in the order of 10^{-2} normal particles per unit area per time step. This is still a small number that is not expected to create much overhead. The second effect of the overlap region, is that particles will mainly interact with their own kind. The scarcity of mixed interactions decreases the total number of interactions considerably. Thus, the effects of the extra overhead and the more efficient Verlet lists counteract. Although we cannot explicitly distinguish between the two, figure 6.8 clearly indicates the prevalence of the latter.

In practice, the mass fraction of coarse-grained particles will be around $\varphi = 0.5$, gaining almost a factor of two in computation time. One should realize that the approach is limited to large systems, because of the considerable amount of space required by the overlap regions. Furthermore, one should be interested in only a part of the system, while the remaining bulk area can be subjected to coarse-graining. When a system fulfills these requirements, the combination of length scales can be applied in various ways on various problems.

We have shown the combined system works properly for both parallel and perpendicular flows. There is no objection to combine the two and create a window of normal particles in the bulk of coarse-grained particles. This window can be free to move, for instance according to the centre of mass of the studied object. A possibility for very large systems, is to create a hierarchially coarse-grained system, by coarse-graining the coarse-grained domain. There is no theoretical limitation to the number of levels, as the coarse-graining procedure keeps the number of interactions per particle constant. In such a way larger length scale differences are bridged. The application of the combined system can be beneficial in a variety of systems. The study of complex molecules such as polymers or proteins under shear flow is a salient example. Also surface phenomena like Rayleigh-Taylor instability or the coalescence of droplets can be studied in a system with combined length scales.

6.5 Conclusion

In this chapter different length scales are combined in DPD to enhance computational efficiency. The normal and coarse-grained description that was developed in the previous chapter are combined into one system. They are coupled via an overlap region, where both types coexist. At the edges particles are

locally refined or coarse-grained. The overlap regions induce a slight perturbation of the density, but the pressure and temperature profiles are unaffected. The combined system reproduces the flow behaviour of a normal system, irrespective of the flow direction. Possible applications are numerous, provided that they involve large systems with large bulk areas. When half of such a system is coarse-grained, this study encouragingly shows that the computation time reduces by a factor of two.

Wormlike chain model

7.1 Introduction

Biological polymers have been extensively studied over recent years, experimentally as well as theoretically. For instance, DNA molecules are investigated in stretching experiments:¹⁵² while fixing one end of the molecule, a force is exerted on the other end and the extension is measured. They are typically inextensible single chains, that possess a degree of rigidity. The model of Kratky and Porod¹⁵³ (or wormlike chain model) is widely used as the basis for the modeling of these semi-flexible polymers. It describes the polymer as an inextensible continuous space curve, that includes a bending elasticity depending on the extent of bending. The wormlike chain is characterized by the persistence length λ . When the contour length L is much larger than this persistence length, $L \gg \lambda$, the polymer behaviour approximates a random walk (with step length 2λ), whereas when $L \ll \lambda$, the rigid rod model is a good approximation. However, many biological polymers possess a rigidity in the intermediate region, $L \approx \lambda$, where these simplified limits are not valid.

The Kratky-Porod model can be solved analytically, providing theoretical averages for the second and fourth moment of the end-to-end vector.¹⁵⁴ A large number of models (see for instance ref. 152, 155–160) have been proposed to complement it with the end-to-end distribution and/or the force-extension behaviour, but they are often limited to specific regimes. For instance, the Marko and Siggia model is only valid for $L \gg \lambda$.¹⁵² Ohm proposed a new model,¹⁶¹ that is based on similarities between the stiff and flexible limits. It provides a distribution function and force-extension relation between any two points on

the space curve. It reproduces the theoretical results accurately over the full range of parameters. When the wormlike chain is discretized by a number of beads, a bending potential needs to be explicitly included. To this end the elastic filament model¹⁶² is commonly used. In this chapter, we will briefly describe Ohm's model and improve the determination of the bending constant for the bending potential. We will test the model with Brownian dynamics simulations and a random generation scheme. The results are compared to the Kratky-Porod model.

7.2 Wormlike chain model

The second and fourth moments of the end-to-end distance r for the Kratky-Porod model are known analytically.¹⁵⁴ Introducing the dimensionless persistence length $q = \lambda/L$, they are given by:

$$\langle r^2 \rangle = 2L^2 \left[\left(e^{-1/q} - 1 \right) q^2 + q \right], \quad (7.1)$$

$$\langle r^4 \rangle = \frac{4}{27} L^4 \left[\left(2e^{-3/q} - 216e^{-1/q} + 214 \right) q^4 - \left(54e^{-1/q} + 156 \right) q^3 + 45q^2 \right]. \quad (7.2)$$

7.2.1 Stretching force

Ohm¹⁶¹ proposes the probability distribution $p(r)$ for the end-to-end distance r :

$$p(r) \propto \frac{q \left(\frac{r}{L} \right)^\alpha}{\left[q \left(1 - \left(\frac{r}{L} \right)^2 \right) \right]^{\frac{1}{2}(\alpha+3)}} \exp \left(-b \frac{r^2}{q(L^2 - r^2)} \right), \quad (7.3)$$

where the semi-empirical parameters α and b are given by:

$$\alpha = \frac{2 + 37 q^{5/4} + 276 q^{5/2}}{1 + 26 q^{5/4} + 60 q^{5/2}}, \quad (7.4)$$

$$b = \frac{(\alpha + 1) (5 q^{5/4} + 1)}{42 q^{5/4} + 4}. \quad (7.5)$$

From this distribution function the stretching force \vec{f}_S between the end points is derived:

$$\vec{f}_S = k_B T \left(\frac{\alpha - 2}{r^2} + \frac{\alpha + 3}{L^2 - r^2} - \frac{2b}{q} \frac{L^2}{(L^2 - r^2)^2} \right) \vec{r}. \quad (7.6)$$

7.2.2 Bending force

As the Kratky-Porod model was derived for a continuous space curve, the relations 7.1 – 7.6 are valid for any pair of points on the chain. This makes an extension to a discrete model straightforward. We construct a polymer of contour length L_{tot} and overall dimensionless persistence length q_{tot} by linking N beads. The contour length L and dimensionless persistence length q for the corresponding segments are given by:

$$L_{\text{tot}} = L (N - 1), \quad (7.7)$$

$$q = q_{\text{tot}} (N - 1). \quad (7.8)$$

When a discrete wormlike chain consists of more than two particles, the bending potential between the segments defines the overall stiffness of the chain, rather than the stretching potential of the individual segments. The elastic filament model¹⁶² defines the bending potential U_B as:

$$U_B(\theta_i) = A (1 - \cos \theta_i), \quad (7.9)$$

where A is the bending constant and θ_i is the angle between segment $(\vec{r}_{i-1} - \vec{r}_i)$ and $(\vec{r}_i - \vec{r}_{i+1})$, when i refers to the consecutive particle labels (see fig. 7.1). This potential pushes the segments apart, but they are also affected by thermal

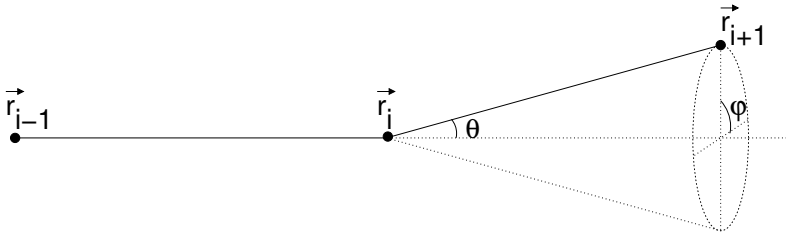


Figure 7.1: Schematic representation of two connected segments with link angle θ and torsion angle φ .

fluctuations. Therefore, the Boltzmann factor, $\exp(-U_B(\theta)/k_B T)$, determines the distribution p_θ :

$$p_\theta(\cos \theta) = 1 - \frac{\alpha \exp(-\alpha(1 - \cos \theta))}{1 - \exp(-2\alpha)}, \quad (7.10)$$

where $\alpha = A/k_B T$. The average of this distribution is $\langle \cos \theta \rangle$:

$$\langle \cos \theta \rangle = 1 - \frac{1}{1 - \exp(-2\alpha)} \left(\frac{1}{\alpha} - \exp(-2\alpha) \left(1 + \frac{1}{\alpha} \right) \right). \quad (7.11)$$

The next step is to relate this average cosine of the link angle to the dimensionless persistence length of the segment. Previous authors have proposed expressions, such as $\langle \cos \theta \rangle = \exp(-1/q)^{163}$ and $\langle \cos \theta \rangle = 1 - 1/q$,¹⁶⁴ for the wormlike chain. However, although they are valid in the limit of large n , they perform poorly at intermediate values ($n \approx 10$). Here, we apply the freely rotating model¹⁶⁵ to relate the average cosine to the dimensionless persistence length. Each rigid link in the model is connected to the previous link with a fixed angle and can rotate freely around it. The average end-to-end distance is known analytically, depending on the angle and the link length. Both are fixed in the freely rotating model. However, in the wormlike chain model they are not constant, so we approximate them with their average values instead:

$$\langle r^2 \rangle_{\text{tot}} = \langle r^2 \rangle_{\text{seg}} \sum_{i=1}^n \sum_{j=1}^n \langle \cos \theta \rangle^{|i-j|} = \langle r^2 \rangle_{\text{seg}} \left(n + 2 \sum_{i=1}^n \sum_{j>i}^n \langle \cos \theta \rangle^{|i-j|} \right). \quad (7.12)$$

This is a polynomial equation in $\langle \cos \theta \rangle$ of the order $(n - 1)$. Note that for two segments ($n = 2$) it reduces to the “cosine rule”. The average size of the total chain $\langle r^2 \rangle_{\text{tot}}$ is determined by the theoretical expression (eq. 7.1) for q_{tot} and L_{tot} . The same expression for q and L gives the average segment size $\langle r^2 \rangle_{\text{seg}}$. Our approach is to solve equation 7.12 numerically for $\langle \cos \theta \rangle$. Subsequently, equation 7.11 is solved for A , which defines the bending potential (eq. 7.9) for the discrete wormlike chain model.

7.3 Computational details

To test the wormlike chain model, we make a distinction between the stretching and bending potential. First we isolate the stretching potential, by examining polymers of two beads at different dimensionless persistence lengths, ranging from $0.01 \leq q \leq 10$. Next, polymers of more beads are studied to assess the effect of the bending potential. The number of beads ranges from $3 \leq N \leq 32$, while the overall dimensionless persistence length is $q_{\text{tot}} = 0.1, 1$ or 10 .

The two-bead polymers are simulated with Brownian dynamics. This will show how well the stretching force (eq. 7.6) approximates the distribution function (eq. 7.3) and whether this distribution in turn produces the theoretical moments (eq. 7.1 and 7.2). The simulations are performed at a thermal energy of $k_B T = 1$, a bead mass of $m = 1$ and a friction coefficient of $\gamma = 10$. These (arbitrary) parameters only influence the dynamics of the system, while we will focus on the static properties, that are independent of the temperature. As the stretching force increases for stiffer polymers, the time step is limited by the dimensionless persistence length q of the segments. The time step should be small enough to prevent the distance between the two beads from exceeding the contour length L (where the stretching potential is undefined). We found this condition satisfied, if the time step obeys $\Delta t = 0.001/q$, with a maximum of $\Delta t_{\text{max}} = 0.01$. After equilibrating the polymer for 10^3 time steps, the end-to-end distance is monitored for 10^9 time steps and averaged.

As will be shown in the two-bead simulations, the force-extension relation accurately reproduces the distribution function. Therefore, it is sufficient (and faster) to use only the latter for the many-bead polymers. We will use a random generation scheme, that constructs a large number of independent wormlike chains. This scheme “grows” polymers by subsequently connecting links with the appropriate parameters. The probability functions of equation 7.3 and 7.10 provide the length of a link and the link angle, while the torsion angle is a uniform random variable $\varphi \in [0, 2\pi)$ (see fig. 7.1). To pick a length or angle for the next segment, we start with a random uniform variable. A uniform deviate $x \in [0, 1)$ transforms to a random deviate y from a known probability distribution $p(y)$, by inverting the integral:¹⁶⁶

$$x = \int_0^y p(y) dy. \quad (7.13)$$

Unfortunately, the integral of the link length probability distribution (eq. 7.3) is not known analytically. Therefore, we discretize the integral and invert it numerically. The transformed random deviates are stored in an array. Picking a random angle, $\cos \theta$, from the angle probability distribution p_θ (eq. 7.10), is much easier, because the integral is known and invertible:

$$\cos \theta = 1 + \frac{1}{\alpha} \ln [1 - x (1 - \exp(-2\alpha))]. \quad (7.14)$$

As an example of finding the value of α , we will consider a polymer of $N = 8$ beads with an overall persistence length of $q_{\text{tot}} = 1$. Let us fix the contour length of a single segment at $L = 1$. Then the overall length $L_{\text{tot}} = (N - 1)L = 7$ (eq. 7.7) and the dimensionless persistence length of a segment is $q = (N - 1)q_{\text{tot}} = 7$ (eq. 7.8). Equation 7.1 provides the average segment distance of $\langle r^2 \rangle_{\text{seg}} = 0.954$ for the segment values $q = 7$ and $L = 1$ and average overall distance of $\langle r^2 \rangle_{\text{tot}} = 36.1$ for the overall values $q_{\text{tot}} = 1$ and $L_{\text{tot}} = 7$. From the freely jointed model one obtains an average cosine of the link angle of $\langle \cos \theta \rangle = 0.885$ (eq. 7.12). As a comparison, the “exponential” relation gives $\langle \cos \theta \rangle = \exp(-1/q) = 0.867$ and the “inverse” relation $\langle \cos \theta \rangle = 1 - 1/q = 0.857$, which are both considerably lower than our value. Finally, solving equation 7.11 gives the value for $\alpha = 8.68$. Although this procedure is quite time-consuming, as it includes two iteration sequences, it only needs to be executed once, at the start of a simulation.

For three different values of the overall persistence length $q_{\text{tot}} = 0.1, 1$ and 10 and the number of beads ranging from $3 \leq N \leq 32$, 10^7 polymers are generated. For an overall persistence length of $q_{\text{tot}} = 1$, we compare our method for finding the bending constant of the bending potential to the “exponential”¹⁶³ and “inverse”¹⁶⁴ methods. For each polymer the second and fourth moments, the end-to-end distribution and the static structure factor is measured. In an isotropic medium the static structure factor $S(\mathbf{k})$ is calculated by:¹⁶⁵

$$S(\mathbf{k}) = \frac{1}{N} \left\langle \left(\sum_{i=0}^{N-1} \cos(\mathbf{k}\vec{r}_i) \right)^2 + \left(\sum_{i=0}^{N-1} \sin(\mathbf{k}\vec{r}_i) \right)^2 \right\rangle. \quad (7.15)$$

This quantity is measured experimentally in scattering experiments, that characterize the microscopic structure of a polymer (see for instance ref. 167). It describes the response function of the polymer for density perturbations.¹⁶⁸

7.4 Results and discussion

7.4.1 Wormlike chain of two beads

We will first examine the stretching force between two beads. Figure 7.2 shows the measured probability distribution of the end-to-end distance, as well as the theoretical distribution from equation 7.3. For all plotted values of the dimensionless persistence length the two distributions coincide exactly. This means that the derived force-extension relation (eq. 7.6) indeed produces the theoretical distribution and that it is valid in both the flexible and stiff limit. Next, we will compare the second and fourth moments of the end-to-end distributions to the Kratky-Porod model (eq. 7.1 and 7.2). The moments are shown in figure 7.3(a) as a function of the dimensionless persistence length q . The measured moments correspond very well with the theory. A closer examination of the deviation between measurements and theory in figure 7.3(b) shows that the second moment is within 1% and the fourth moment within 2% (with the exception of $q_{\text{tot}} = 0.01$) from the theoretical values. This good agreement demonstrates that this force-extension relation can be used to model a wormlike chain, over the whole range between the flexible and stiff limits. The peculiar trends in figure 7.3(b) are the result of the semi-empirical character of the distribution. Note that for $q_{\text{tot}} \geq 0.2$ both moments are smaller than one, which will affect the many-bead polymers.

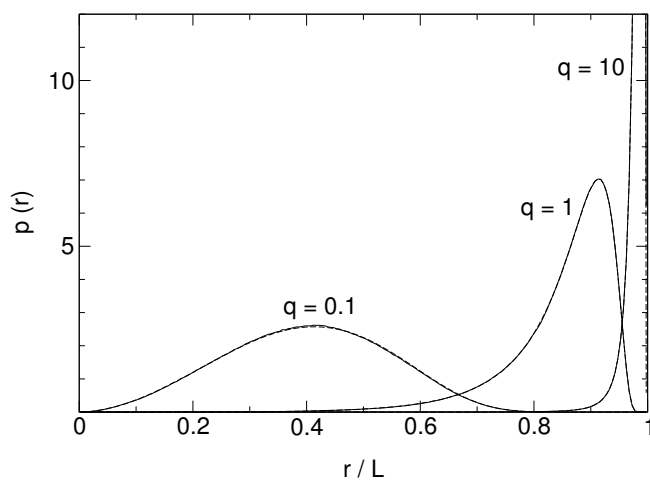


Figure 7.2: Measured (—) and theoretical (---) end-to-end distance distribution for different values of the dimensionless persistence length q . For clarity only the lower part of the $q = 10$ distribution is shown, see figure 7.4(e) for the complete distribution.

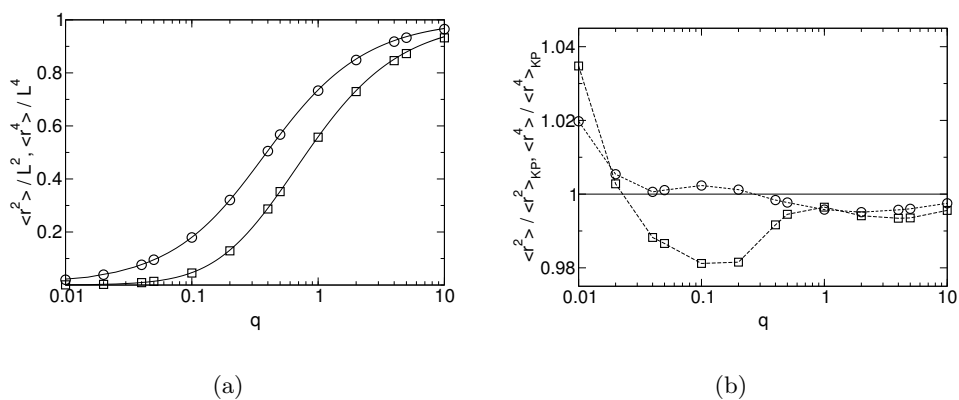


Figure 7.3: (a) Measured second (\circ) and fourth (\square) moment of the end-to-end distance distribution and Kratky-Porod model (—) and (b) deviation from theory, as a function of the dimensionless persistence length q .

7.4.2 Wormlike chain of N beads

The stretching force between two beads in the previous section accurately reproduces the theoretical properties. As the many-bead polymers in fact consist of a string of two-bead polymers, the average segment length will also be correct. So, we will only consider the overall properties of the many-bead polymers.

In the left panels of figure 7.4 the probability distribution is shown for the shortest ($N = 3$) and the longest ($N = 32$) wormlike chain, as well as the theoretical distribution. Note that the scales of the x - and y -axes differ for each value of the overall persistence length. In each figure the distributions of both the shortest and longest chain are close to the theoretical distribution, although slightly left-shifted and differently shaped. These differences are attributed to the choice of the bending potential and to the fact that the average segment lengths are slightly smaller than theory (as shown in the previous paragraph). Their combination apparently leads to a systematic underestimation of the average end-to-end distance.

A more significant measure than the end-to-end distance, is the static structure factor. The latter includes all correlations between all beads, rather than only correlating the two end-points. It gives information about the mass distribution around the centre of mass of the polymer and its response to perturbations. The static structure factor is shown in the right panels of figure 7.4 for polymers consisting of $N = 4, 8, 16$ and 32 beads. For large wavelengths (small k -values) the static structure factor coincides for all polymer sizes. This means that small polymers capture the static properties of large polymers, until the static structure factor breaks down when the dimensionless wave vector $kL_{\text{tot}}/2\pi$ becomes larger than the size of the polymer.

Next, we examine the ratio of the second and fourth moments of the many-bead polymers to the predictions of the Kratky-Porod model in figure 7.5. For comparison the results for two beads are shown as well. It is obvious that all measured moments are smaller than the theoretical moments. The reason is the combination of the bending potential and the segment lengths that are smaller than theory, as explained above. Despite this deviation, the measured second moment is within 2% and the fourth moment within 6% from theory. When the overall persistence length or the number of beads increases, the deviation from theory decreases. The polymers contain increasingly stiffer segments, which improves the application of the bending potential. So, although

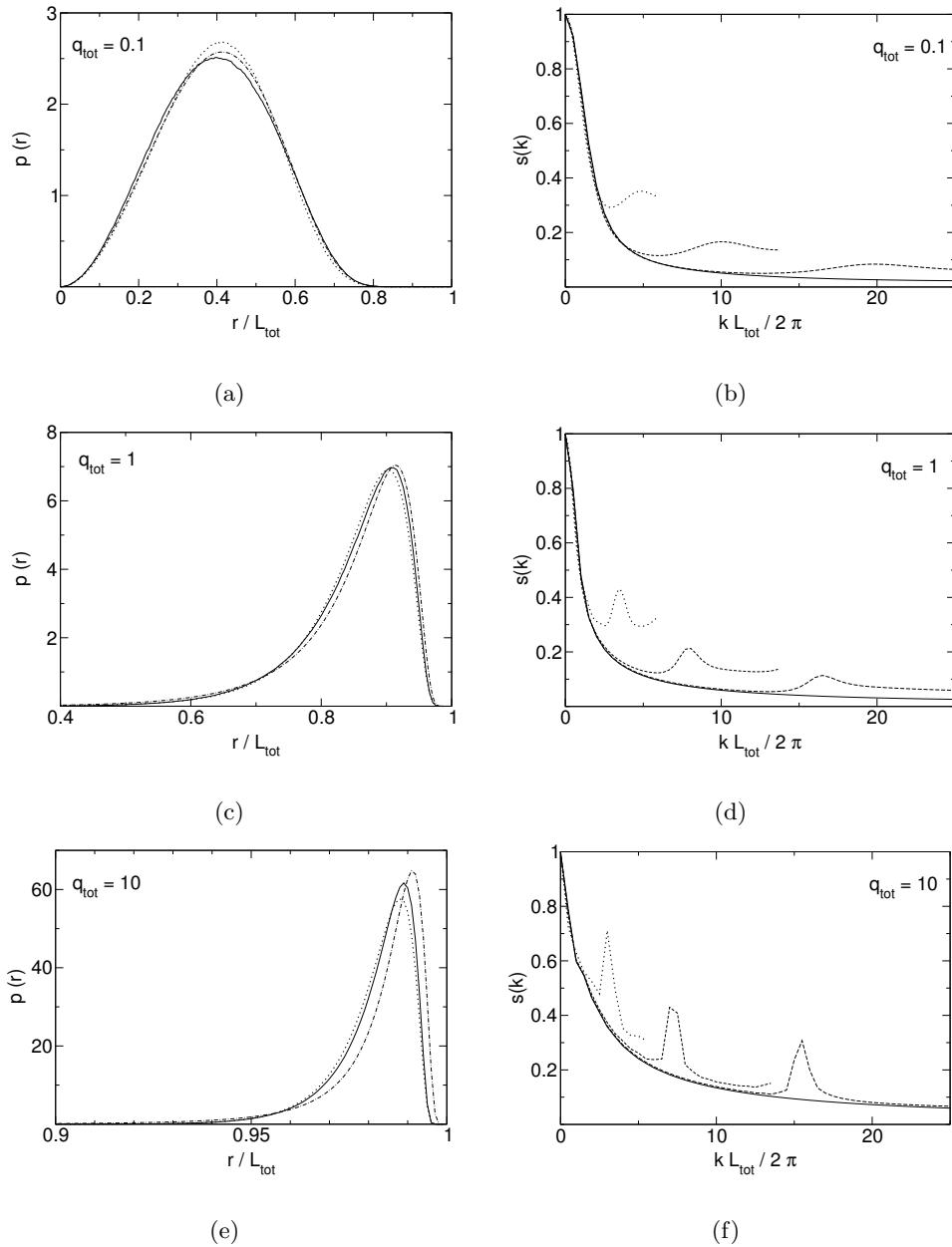


Figure 7.4: Left panels: end to end distribution functions of 3-bead polymer (\cdots), 32-bead polymer ($—$) and theoretical distribution of equation 7.3 ($- \cdot - \cdot$). Right panels: static structure factor of 4-bead polymer (\cdots), 8-bead polymer ($- -$), 16-bead polymer ($— —$) and 32-bead polymer ($—$). Overall dimensionless persistence lengths are (a) and (b) $q_{\text{tot}} = 0.1$; (c) and (d) $q_{\text{tot}} = 1$; (e) and (f) $q_{\text{tot}} = 10$.

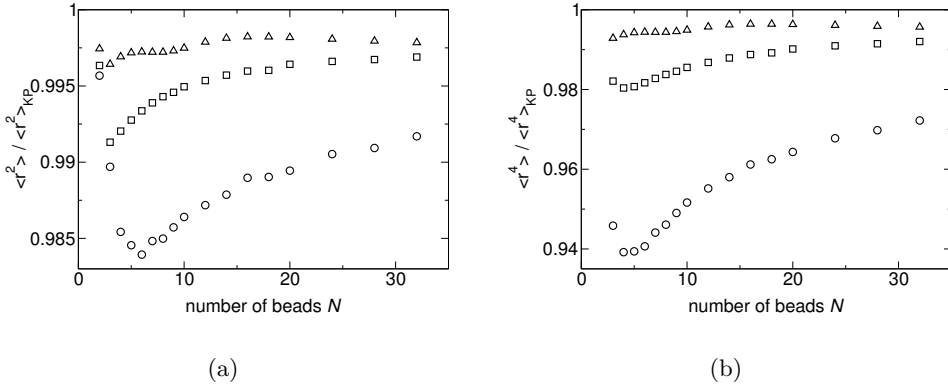


Figure 7.5: Measured (a) second and (b) fourth moment of the end-to-end distance distribution, normalized on the theoretical value of the Kratky-Porod model, for the dimensionless overall persistence length of $q_{tot} = 0.1$ (\circ), $q_{tot} = 1$ (\square) and $q_{tot} = 10$ (\triangle).

our discretized wormlike chain model works for a large range of rigidities and lengths, it performs best for rather long and stiff chains.

Finally, we compare our method using the freely rotating model to other methods to determine the bending constant for the bending potential. These are the “exponential” method ($\langle \cos \theta \rangle = \exp(-1/q)$) and the “inverse” method ($\langle \cos \theta \rangle = 1 - 1/q$). In figure 7.6 the normalized second and fourth moments are shown for a wormlike chain of overall persistence length $q_{tot} = 1$ of increasing number of beads N . It is clear that all methods cause the moments to approximate the theoretical values for large N . However, over the whole range the measured moments using our method are closer to theory, especially at low and intermediate values. The static structure factors show the same trend; the structure factor for a $N = 32$ polymer coincides for all methods, while it shows deviations for a $N = 4$ polymer.

7.5 Conclusion

In this chapter we have introduced a discrete wormlike chain model that incorporates a stretching potential between beads and a bending potential between links. The stretching force is based on the probability distribution proposed

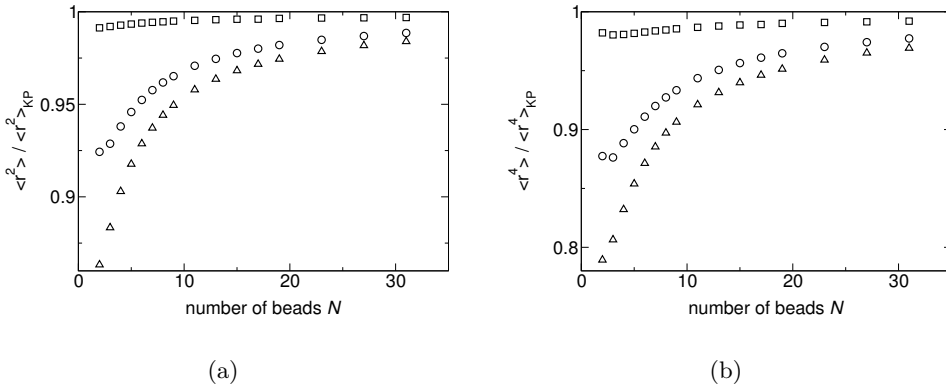


Figure 7.6: Measured (a) second and (b) fourth moment of the end-to-end distance distribution, normalized on the theoretical value of the Kratky-Porod model for $q_{\text{tot}} = 1$ using the freely rotating method (\square), the “exponential” method (\circ) and the “inverse” method (\triangle) for determining the bending constant of the bending potential.

by Ohm,¹⁶¹ and the bending force is taken from the elastic filament model.¹⁶² Its bending constant is determined using the freely rotating model, which is significantly better than previous methods. The result is a discrete particle model for wormlike chains, that accurately reproduces the large-wavelength behaviour and the second and fourth moments of the Kratky-Porod model (to approximately 5%). The model is applicable over the range from the flexible to stiff limit.

Multiple time step algorithm for explicit solvent simulations

8.1 Introduction

In nature, biological polymers are usually found in aqueous surroundings. Under dilute and semidilute conditions the hydrodynamic interactions are not negligible, but can have a strong effect on the dynamic behaviour of the polymers. These effects can be modeled by the Oseen tensor,¹⁶⁵ but in complex geometries or under semidilute conditions, its application becomes computationally complicated. In such cases, one can resort to an explicit solvent. By taking the interactions of the solvent and solute particles into account, the hydrodynamic effects are resolved by construction. Often, the solvent particles possess a soft potential or no potential at all, while biological polymers are modelled by potentials that can approach infinity. This limits the maximal time step for the simulation, even though the solvent particles could be modelled with considerably larger time steps.¹⁶⁹ For such systems, a multiple time step algorithm can bridge the gap in time scales. One of the first known multiple time step algorithms is the reference system propagator algorithm (*RESPA*), that was introduced by Tuckerman *et al.* for systems with disparate masses¹⁷⁰ and for systems with long range forces.¹⁷¹ Its reversible variant *r-RESPA*¹⁷² and the similar *Verlet-I* algorithm¹⁷³ are most commonly used. It is a Verlet algorithm with an inner loop for the small time steps, while the outer loop handles the large time steps. This approach enables for instance the simulation of a stiff oscillator in a soft fluid,¹⁷⁴ where the oscillator is simulated with a small time step and the solvent with a large time step. A similar

algorithm has been proposed by Symeonidis *et al.*, that combines a polymer with a DPD solvent.¹⁷⁵ As all solvent particles are only simulated during the large time step, this approach does not reproduce the hydrodynamic effects on the polymer.

We will simulate the polymer and the solvent particles directly surrounding it, with a small time step. The bulk of solvent particles farther away, will be updated with a large time step. At each large time step, solvent particles can switch type, depending on the distance to the polymer. Our approach requires an algorithm from the Verlet family. However, the modified velocity Verlet algorithm that is commonly used to integrate the DPD equations, is not consistent due to the velocity dependence of the dissipative force. As the DPD thermostat depends intrinsically on the time step, detailed balance - which is crucial for the thermostat to work properly - cannot be obtained with the multiple time step algorithm for DPD. Instead, we use the Lowe-Andersen thermostat¹⁸ to control the temperature. In a bath collision, the relative velocity of a particle pair is changed according to the Maxwell-Boltzmann distribution. Using our multiple time step algorithm, all particles still experience the same number of bath collisions per time unit, thus ensuring identical dynamic properties, such as the viscosity.

First, we will apply the multiple time step algorithm on a simple fluid, consisting of identical particles, and investigate its properties. Obviously, the algorithm is not required in this case. Next, we will study a solution with a single wormlike chain, described by the model that was introduced in the previous chapter. Due to the finite extensibility of the wormlike chain, the potentials in the model are very steep, limiting the time step for the polymer. In this case the multiple time step algorithm is crucial to save computation time. Using the algorithm we will compare the internal dynamics of model biological polymers with and without hydrodynamic interactions.

8.2 Multiple time step model

8.2.1 Multiple time step algorithm

Starting from the velocity Verlet algorithm,¹⁰ we will modify it to incorporate multiple time stepping, without violating the original algorithm. We define two types of particles. Type 0 particles take a small time step $\Delta\tau$, while type

```

algorithm mts {
  check particle types
   $\vec{r}_1(t + \Delta t) = \vec{r}_1(t) + \Delta t \vec{v}_1(t) + \frac{1}{2} \Delta t^2 (\vec{f}_{11}(t) + \vec{f}_{10}(t))$ 
   $\vec{v}_1(t + \frac{1}{2} \Delta t) = \vec{v}_1(t) + \frac{1}{2} \Delta t (\vec{f}_{11}(t) + \vec{f}_{10}(t))$ 
  if ( $\Delta_1^{\max} \geq \Delta^{\text{crit}}$ ) update verletlist
  for (i = 0; i < (int)( $\Delta t / \Delta \tau$ ); i++) {
     $\vec{r}_0(t + (i + 1) \Delta \tau) = \vec{r}_0(t + i \Delta \tau) + \Delta \tau \vec{v}_0(t + i \Delta \tau) +$ 
       $\frac{1}{2} \Delta \tau^2 (\vec{f}_{00}(t + i \Delta \tau) + \vec{f}_{01}(t + i \Delta \tau))$ 
     $\vec{v}_0(t + (i + \frac{1}{2}) \Delta \tau) = \vec{v}_0(t + i \Delta \tau) +$ 
       $\frac{1}{2} \Delta \tau (\vec{f}_{00}(t + i \Delta \tau) + \vec{f}_{01}(t + i \Delta \tau))$ 
    if ( $\Delta_0^{\max} \geq \Delta^{\text{crit}}$ ) update verletlist
     $\vec{f}_{00}(t + (i + 1) \Delta \tau) = g [\vec{r}_0(t + (i + 1) \Delta \tau)]$ 
     $\vec{f}_{10}(t + (i + 1) \Delta \tau) = \vec{f}_{01}(t + (i + 1) \Delta \tau)$ 
       $= g [\vec{r}_0(t + (i + 1) \Delta \tau), \vec{r}_1(t + \Delta t)]$ 
     $\vec{v}_0(t + (i + 1) \Delta \tau) = \vec{v}_0(t + (i + \frac{1}{2}) \Delta \tau) +$ 
       $\frac{1}{2} \Delta \tau (\vec{f}_{00}(t + (i + 1) \Delta \tau) + \vec{f}_{01}(t + (i + 1) \Delta \tau))$ 
    LA thermostat ( $\Delta \vec{v}_{00}, \Gamma \Delta \tau$ )
  }
   $\vec{f}_{11}(t + \Delta t) = g [\vec{r}_1(t + \Delta t)]$ 
   $\vec{v}_1(t + \Delta t) = \vec{v}_1(t + \frac{1}{2} \Delta t) + \frac{1}{2} \Delta t (\vec{f}_{11}(t + \Delta t) + \vec{f}_{10}(t + \Delta t))$ 
  LA thermostat ( $\Delta \vec{v}_{01} = \Delta \vec{v}_{10}, \Gamma \Delta t$ )
  LA thermostat ( $\Delta \vec{v}_{11}, \Gamma \Delta t$ )
}

```

Figure 8.1: Multiple time step algorithm

1 particles take a larger time step Δt . Obviously, the positions and velocities of the type 0 particles need to be updated $\Delta t/\Delta\tau$ times in one large time step. Figure 8.1 shows a pseudo-code for one large time step with the multiple time step algorithm. The subscripts 0 and 1 refer to the corresponding particle types, while double subscripts denote interaction terms between like and unlike particle pairs.

At the beginning of the large time step, particles can switch type according to a preset criterion, such as the position in the simulation domain or the distance to a polymer. In the latter case the current Verlet list can be used for this purpose. The positions of type 1 particles at time t are advanced over a time step Δt . Their velocities take a $\frac{1}{2}\Delta t$ step, following the velocity Verlet algorithm. For both evaluations the force exerted by other type 1 particles, $\tilde{\mathbf{f}}_{11}$ is needed, as well as the force experienced from type 0 particles, $\tilde{\mathbf{f}}_{10}$, at time t . If the maximum displacement of a type 1 particle exceeds the Verlet list criterion, the list needs updating. Next, the inner loop is executed $\Delta t/\Delta\tau$ times. The positions and velocities of the type 0 particles are updated in the same way as the type 1 particles, albeit with a time step of $\Delta\tau$. The force terms $\tilde{\mathbf{f}}_{00}$ and $\tilde{\mathbf{f}}_{01}$ are available at the beginning of each small time step $\Delta\tau$. Again the Verlet lists are checked, but now for the maximum displacement of the type 0 particles. Next, the forces between type 0 particles pairs and between mixed particle pairs are evaluated, according to a potential g based on the particle positions. When we are simulating a polymer in solution, this potential consists of the spring forces between polymer particles and the solvent potential. Note that the mixed force term $\tilde{\mathbf{f}}_{01} = \tilde{\mathbf{f}}_{10}$ is based on the current type 0 positions, but also on the type 1 positions at the end of the large time step $t + \Delta t$, because these were updated before. With these force terms the velocities of the type 0 particles can advance for $\frac{1}{2}\Delta\tau$. The last step of the inner loop comprises the thermostating. With a probability of $\Gamma\Delta\tau$ new relative velocities are drawn from a Gaussian distribution.¹⁸ Unlike the force calculation, this step only alters the velocities of type 0 particles. After all runs through the inner loop, the mixed force terms have progressed to their value at $t + \Delta t$. The type 1 force term $\tilde{\mathbf{f}}_{11}$ is calculated from the positions of the type 1 particles. Together these two force terms determine the last velocity update for the type 1 particles. Finally, the relative velocities of the 11 and mixed interaction pairs are thermostatted, with a probability of $\Gamma\Delta t$.

Some remarks on this algorithm need to be taken into account. The particle types are determined only at the beginning of a large time step. During this time step the polymer could drift too close to a type 1 particle, with which

it should not have interaction. This could happen when the Verlet lists are updated more than once in a large time step. Therefore the cut-off radius for the Verlet lists should be chosen such that updates take place less frequently. The updates themselves can take place in both the outer and inner loop, even when particles switch type in between updates. Because the displacements are monitored for each individual particle and the update criterion is identical for both types, the updates are independent of the particle type switches. The thermostating steps depend on the time steps Δt or $\Delta \tau$ and the bath collision frequency Γ . In a large time step the type 1 particles that are completely surrounded by like particles, will be thermostatted with a probability of $\Gamma \Delta t$ per interaction. Or, they experience $N_{\text{int}} \Gamma \Delta t$ bath collisions per large time step, where $N_{\text{int}} = 4\pi n r_c^3 / 3$ is the number of particles that surround a particle. Similarly, the type 0 particles undergo $N_{\text{int}} \Gamma \Delta \tau$ bath collisions during a small time step. As this takes place $\Delta t / \Delta \tau$ times, the total number of bath collisions in a large time step is $N_{\text{int}} \Gamma \Delta t$, identical to the type 1 particles. Particles that interact with both types of particles, experience bath collisions in both the small and large time step. These will also add up to $N_{\text{int}} \Gamma \Delta t$, provided that the fraction of surrounding type 0 and 1 particles remain more or less constant in one large time step. This results in identical dynamic properties for all particles.

In our polymer simulations we will use an ideal solvent, which simplifies the algorithm significantly. Almost all force terms are zero, except for the forces between the polymer particles. The velocity updates of the solvent particles is redundant as well; their velocity is only altered by the thermostat.

8.2.2 Computational efficiency

The multiple time step algorithm is designed to save computation time. However, it also introduces the distinction between type 0 and type 1 particles, which causes an extra overhead in most subroutines. Neglecting this fact, we expect the computation time to be proportional to the fraction x_0 of type 0 particles and the ratio $\Delta t / \Delta \tau$:

$$T_{\text{comp}} \propto \left(\frac{\Delta t}{\Delta \tau} x_0 + (1 - x_0) \right) n N_{\text{tot}}, \quad (8.1)$$

where n is the number density and N_{tot} the total number of particles. When

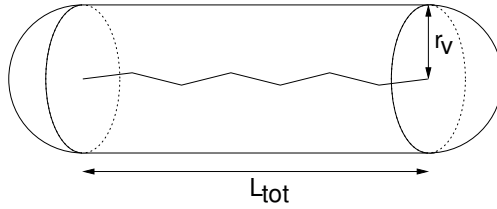


Figure 8.2: Schematic representation of the volume surrounding a wormlike chain.

we measure the computation time as a function of fraction χ_0 , this expression predicts the slope to be $(\Delta t/\Delta\tau - 1)$. Similarly, the slope of the computation time versus the multiple time step ratio $\Delta t/\Delta\tau$ is expected to be χ_0 . This can easily be verified in simulations, when one variable is varied, while the other is fixed. However, in the wormlike chain simulations both variables depend on the number of polymer particles N . We can estimate the number of type 0 particles in the vicinity of the polymer by imagining a straight tube around the polymer (see fig. 8.2). The tube's length is the polymer length L_{tot} and the radius the Verlet list radius r_v . In the previous chapter, the total length was defined as $L_{\text{tot}} = L(N - 1)$, where L is the distance between two beads. The volume of the tube and two semi-spheres of radius r_v at either end, contains all particles N_0 that should be flagged as type 0:

$$N_0 \simeq n \left(\pi r_v^2 L(N - 1) + \frac{4}{3} \pi r_v^3 \right). \quad (8.2)$$

It should be noted that this approximation is only valid for straight polymers. Wormlike chains with an overall dimensionless persistence length of one or higher, satisfy this requirement to a good approximation. The number of beads also dictates the time step ratio. In the previous chapter, we found the small time step $\Delta\tau$ to be reciprocally dependent on the persistence length of the polymer segment q :

$$\Delta\tau = 0.001/q, \quad (8.3)$$

which in turn depends on the number of beads and the overall persistence length:

$$\mathbf{q} = \mathbf{q}_{\text{tot}}(\mathbf{N} - 1). \quad (8.4)$$

So, when we discretize a wormlike chain with constant \mathbf{q}_{tot} , by a growing number of polymer beads, the segments get stiffer, and the allowed time step gets smaller. How all this affects the total computation time in equation 8.1 depends on the simulation setup, *e.g.* how the total number of particles varies with the number of polymer beads.

8.3 Computational details

The multiple time step algorithm aims to facilitate computations that suffer from disparate time scales. However, first a series of simulations of identical particles are performed. These will provide an insight into the effect of the algorithm on the basic properties, such as density, pressure and velocity. Once the multiple time step algorithm is satisfactorily validated in this simple case, we focus on a system where different time steps are required. A solution of a discrete wormlike chain, that was introduced in the previous chapter, fits this requirement perfectly. The steep potentials drastically limit the time step for the polymer, while a large number of solvent particles are required to incorporate hydrodynamic interactions accurately.

8.3.1 Simulations without polymer

For the simulations with identical simple particles, we define a base case for which the parameters are number density $n = 6$, thermal energy $k_B T = 1$, thermal collision frequency $\Gamma = 10$ and particle mass $m = 1$. Here we consider a system with conservative forces because it is the simplest non-trivial test of the methodology. The conservative force is a soft potential ($f_C = a(1 - r/r_c)$) with a phase repulsion $a = 10$. The cut-off for the interaction radius is $r_c = 1$ and for the Verlet radius $r_v = 1.3$. The number of particles is $N = 6000$, in a cubic simulation domain of volume $(10r_c)^3$. The large time step is always $\Delta t = 0.01$, while the corresponding small time step depends on the ratio used.

For the generation of profiles, the fraction of type 1 particles is $x_1 = 0.5$. The simulations are run for 1000 time units, after an equilibration of 100 time units. A body force of $g = 0.5$ is applied in opposite directions.¹³⁸ To test the

density profile, time step ratios $\Delta t/\Delta\tau$ of 1, 3, 10, 30 and 100 are studied. For the velocity profiles different regions for the type 0 particles are examined, at a ratio of $\Delta t/\Delta\tau = 10$. The type 0 particles either occupy the left part or the middle part of the simulation domain, or they are perfectly mixed with the type 1 particles. This allows us to compare the dynamic behaviour and thus the viscosity of the different types.

To validate the scaling of the computation time (eq. 8.1), short simulations using the base case parameters are run for 20 time units, after an equilibration of one time unit. The CPU time is measured as a function of the fraction x_0 of type 0 particles and the ratio $\Delta t/\Delta\tau$.

8.3.2 Simulations with polymer

For the polymer simulations a single wormlike chain is dissolved in an ideal fluid (*i.e.* without conservative forces), which simplifies the multiple time step algorithm significantly. The fluid has a number density $n = 3$, thermal energy $k_B T = 1$, thermal collision frequency $\Gamma = 10$ and particle mass $m = 1$. We use a large time step $\Delta t = 0.01$.

First, we investigate a wormlike chain that is discretized by $N = 8$ beads, with an overall dimensionless persistence length of $q_{\text{tot}} = 1$. The dimensionless persistence length of a segment is $q = (N - 1)q_{\text{tot}} = 7$. In the previous chapter, we derived the time step for the wormlike chain model, as $\Delta\tau = 0.001/q = 1.43 \cdot 10^{-4}$. Consequently, the multiple time step ratio is $\Delta t/\Delta\tau = 70$. The segment length between two beads is taken as the average distance between two solvent particles: $L = n^{-1/3} = 0.69$. The overall length is $L_{\text{tot}} = L(N - 1) = 4.85$, while the simulation domain is a cubic box of volume 10^3 . Four simulations - with and without the multiple time step algorithm - are run for 5000 time units, excluding an equilibration time of 10 time units. The results of the short (with multiple time stepping) and long runs (without multiple time stepping) are compared to validate the algorithm for the wormlike chain in solution. The remainder of the simulations are only run using multiple time stepping.

For a discrete wormlike chain with an overall dimensionless persistence length of $q_{\text{tot}} = 1$, the number of beads is varied, from $N = 4, 8, 12, 16, 20$ to 24. The corresponding polymer parameters are shown in table 8.1. For larger polymer lengths, the distance between two polymer beads cannot be restrained to the contour length L , which is the maximum allowed. This is due to the

N	L_{tot}	q	$\Delta\tau$	$\Delta t/\Delta\tau$	ζ_{BD}
4	2.08	3	$3.33 \cdot 10^{-4}$	30	10.9
8	4.85	7	$1.43 \cdot 10^{-4}$	70	8.29
12	7.63	11	$9.09 \cdot 10^{-5}$	110	7.06
16	10.4	15	$6.67 \cdot 10^{-5}$	150	7.03
20	13.2	19	$5.26 \cdot 10^{-5}$	190	6.38
24	15.9	23	$4.35 \cdot 10^{-5}$	230	6.27

Table 8.1: Parameters for discrete wormlike chain simulations with $q_{\text{tot}} = 1$, $L = 0.69$ and varying number of beads N .

combination of extremely stiff segments and large time step ratio. We can avoid this by decreasing the large time step Δt for the type 1 particles.

The solvent parameters are the same as above ($n = 3$, $k_B T = 1$, $\Gamma = 10$, $m = 1$). To avoid a varying finite size effect, we need to change the system size according to the polymer size. As a wormlike chain of $q_{\text{tot}} = 1$ is more or less stretched out, we can use the contour length $L_{\text{tot}} \propto N$ as typical length scale. By choosing the simulation domain as N^3 , we actually simulate identical “periodic” semi-dilute solutions.

The same simulations are performed with Brownian dynamics, to assess the effect of hydrodynamic interactions. We choose the Brownian parameters such, as to obtain an identical centre of mass diffusion. The centre of mass of the polymer is characterized by its position \vec{r}_{cm} and velocity \vec{v}_{cm} :

$$\vec{r}_{\text{cm}} = \frac{1}{N} \sum_{i=0}^{N-1} \vec{r}_i, \quad (8.5)$$

$$\vec{v}_{\text{cm}} = \frac{1}{N} \sum_{i=0}^{N-1} \vec{v}_i. \quad (8.6)$$

The translational diffusion of the centre of mass \mathbb{D}_{cm} is measured via the velocity autocorrelation function, using the Green-Kubo relation:¹⁹

$$\mathbb{D}_{\text{cm}} = \frac{1}{3} \int_0^\infty \langle \vec{v}_{\text{cm}}(t) \vec{v}_{\text{cm}}(0) \rangle dt, \quad (8.7)$$

or the mean squared displacement:¹⁹

$$\mathbb{D}_{\text{cm}} = \frac{1}{6} \lim_{t \rightarrow \infty} \frac{d \langle (\vec{r}_{\text{cm}}(t) - \vec{r}_{\text{cm}}(0))^2 \rangle}{dt}. \quad (8.8)$$

Both methods give comparable results. We measure the diffusion of the worm-like chain in solution and use the values to tune the friction coefficient ζ_{BD} for the Brownian simulations¹⁶⁵(see table 8.1):

$$\zeta_{\text{BD}} = \frac{k_{\text{B}}T}{N\mathbb{D}_{\text{cm}}} \quad (8.9)$$

By equalizing this translational diffusion, the effect of the hydrodynamic interactions on the internal dynamics of the polymer is isolated. This becomes apparent when we compare the Rouse modes of the polymer with and without solvent. The Rouse modes are a measure of dynamic properties, such as the rotational diffusion. The mode \vec{X}_p is a normal coordinate of the polymer, with an integer wavenumber p :¹⁶⁵

$$\vec{X}_p = \sqrt{\frac{2}{N}} \sum_{i=0}^{N-1} \vec{r}_i \cos\left(\frac{p\pi(i + \frac{1}{2})}{N}\right), \quad p = 0, 1, 2, 3... \quad (8.10)$$

The autocorrelation function of the Rouse modes decay exponentially, with a time constant τ_p that is characteristic for the mode:

$$\langle \vec{X}_p(t) \vec{X}_p(0) \rangle = \langle X_p^2 \rangle \exp\left(-\frac{t}{\tau_p}\right). \quad (8.11)$$

We will compare these relaxation times (which should not be confused with the small time step $\Delta\tau$), for the polymers with and without hydrodynamic interactions.

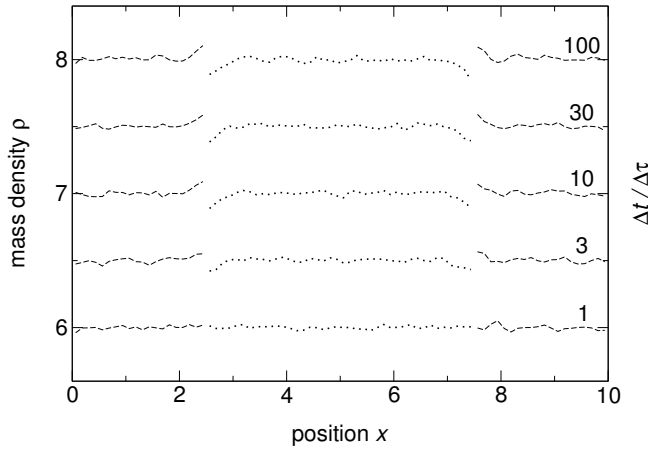


Figure 8.3: Overall density profiles consisting of type 0 particles (\cdots) and type 1 particles ($--$), with $n = 6$, $k_B T = 1$, $\Gamma = 10$, $\alpha = 10$, shifted for various time step ratios $\Delta t / \Delta \tau$.

8.4 Results and discussion

8.4.1 Validation multiple time step algorithm

To verify the multiple time step algorithm, we investigate its effects on simulations with identical particles. Depending on the location in the simulation domain, particles are either simulated with a small or large time step. The resulting profiles show whether the different domains behave identically and whether anomalies at the boundaries occur.

The first profile to examine is the density profile, shown in figure 8.3. The overall density is shown for different values of the time step ratio, shifted vertically for clarity. Obviously, the density profile with a ratio of one, corresponds to a simulation without the multiple time step algorithm. Therefore, the overall density is equal in both domains and it does not show any fluctuations other than stochastic noise. For larger ratios, the density in both domains is also equal, but the profile is perturbed at the boundaries. Fortunately, the deviation is not very large (less than 2%) and independent of the time step ratio (when $\Delta t / \Delta \tau > 1$). An alternative is to introduce an overlap region, in which both types of particles coexist. An example is shown in figure 8.4.

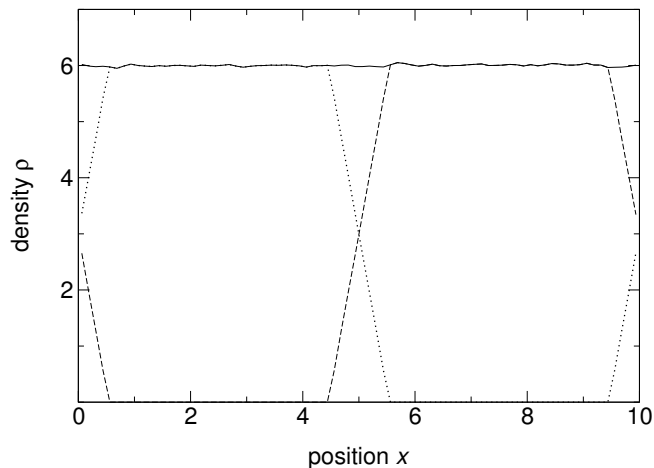


Figure 8.4: Overall density profile (—) with overlap region, consisting of type 0 particles (···) and type 1 particles (---), with $n = 6$, $k_B T = 1$, $\Gamma = 10$, $a = 10$ and $\Delta t/\Delta\tau = 10$.

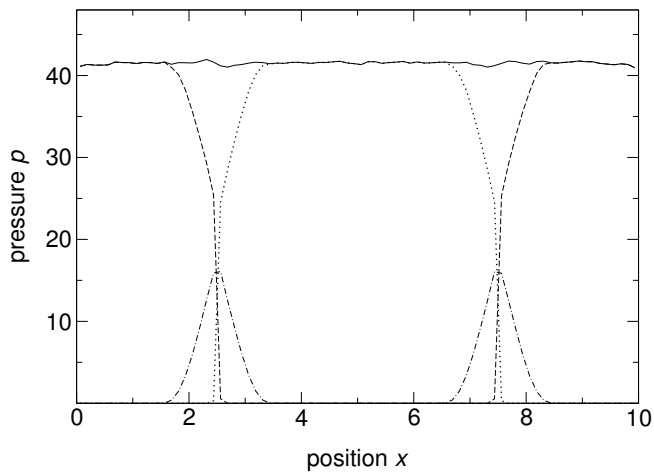


Figure 8.5: Overall pressure profile (—), consisting of type 0 ideal and excess pressure (···), type 1 ideal and excess pressure (---) and mixed excess pressure (·-·-), with $n = 6$, $k_B T = 1$, $\Gamma = 10$, $a = 10$ and $\Delta t/\Delta\tau = 10$.

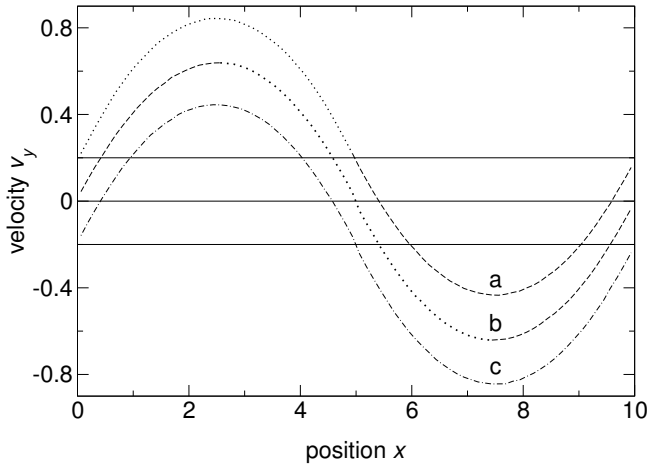


Figure 8.6: Velocity profiles of type 0 particles (\dots) and type 1 particles ($- -$) with $n = 6$, $k_B T = 1$, $\Gamma = 10$, $\alpha = 10$, $g = 0.5$ and $\Delta t / \Delta \tau = 10$; (a) left: type 0, right: type 1 (shifted $+0.2$), (b) inside: type 0, outside: type 1, (c) type 0 and 1 mixed (shifted -0.2).

Connected to the density profile, is the pressure profile, shown in figure 8.5. Again the pressure in both domains is identical, and thus balanced. The deviations at the domain boundaries are less pronounced than in the density profile.

Finally, figure 8.6 shows the velocity profiles for a periodic Poiseuille flow. Three different setups are considered. For clarity the resulting profiles are shifted vertically, as they fall on top of each other. In profile (a) the left part of the simulation domain takes small time steps, while the right part is updated at large time steps. The maximum values of the velocity in both domains are identical opposites, confirming that the viscosities are equal. The velocity at the domain boundaries is zero, as expected, but it is also the location where the body force switches sign. To isolate the effect of the different time steps, we turn to profile (b). Here the type 0 particles are located in the middle of the simulation domain, while the type 1 particles occupy the outer parts. Again the maximum velocities are equal opposites and the transition is smooth, despite the fluctuations in density. Finally, the different types of particles are mixed. Profile (c) shows the same velocities, and hence viscosity, as the separated profiles. Thus, the multiple time step algorithm does not affect dynamic behaviour and its application is not limited to specific setups.

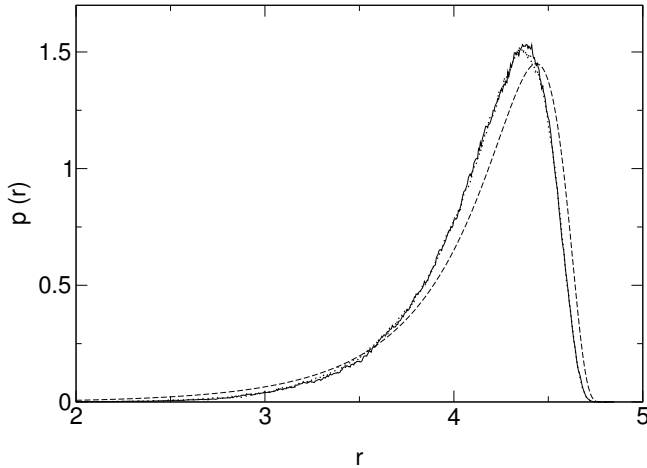


Figure 8.7: End-to-end distribution of wormlike chain with (—) and without (···) multiple time step algorithm and theoretical distribution (---)

8.4.2 Wormlike chain in solution

Before we apply the multiple time step algorithm to wormlike chains of variable length, we compare the results for one polymer with and without multiple time stepping. This chain consists of $N = 8$ beads and is dissolved in an ideal fluid. First we need to check whether the algorithm alters the static properties of the polymer. Figure 8.7 shows the average end-to-end distributions of the polymers with and without multiple time stepping, together with the theoretical distribution (see previous chapter). The two measured distributions agree well; the difference at the peak is not statistically significant. Both distributions however, deviate from the theoretical, as expected from the results in the previous chapter.

The translational diffusion coefficient of the centre of mass is measured for the polymer with (+mts) and without (-mts) multiple time stepping. The measurements are consistent:

- $\mathbb{D}_{\text{cm}}(+\text{mts}) = 0.0165 \pm 0.0004$
- $\mathbb{D}_{\text{cm}}(-\text{mts}) = 0.0170 \pm 0.0007$

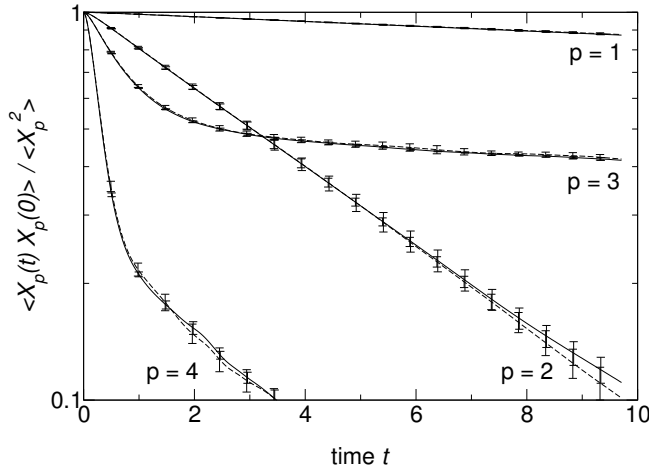


Figure 8.8: Autocorrelation function of Rouse mode X_p with mode p , for wormlike chain with (—) and without (---) multiple time step algorithm.

This means the algorithm does not alter this dynamic property. For a more thorough examination of the dynamic properties, we consider the Rouse modes of the wormlike chains. Only the first four modes ($p \in \{1, 2, 3, 4\}$) are plotted in figure 8.8. Again the autocorrelation function decay coincides for both chains; deviations are within the stochastic error. However, the decay should be exponential according to theory (*i.e.* a straight line in the log-plot of fig. 8.8), but only the first two modes comply. Zatozsky and Lisy showed that the Rouse mode decay is a fractional power rather than exponential, when the time dependent Navier-Stokes equation is explicitly taken into account.¹⁷⁶ However, this is probably not the cause here, as the Brownian dynamics simulations show the same trend. For the subsequent polymers, we will only compare the relaxation times of the first two Rouse modes.

From the simulations with and without multiple time step algorithm, we conclude that the algorithm does not introduce any artefacts. For the remainder of the simulations we will use multiple time stepping. Wormlike chains with an overall dimensionless persistence length of $q_{\text{tot}} = 1$, are discretized in $N = 4, 8, 12, 16, 20$ and 24 beads. The measured centre of mass diffusion for the wormlike chains in solution is shown as circles in figure 8.9. From these measurements we determine the friction coefficient for the Brownian dynamics simulations, using equation 8.9. The used friction coefficients ζ_{BD} are shown in table 8.1. The squares in figure 8.9 denote the resulting diffusion coefficients for the Brown-

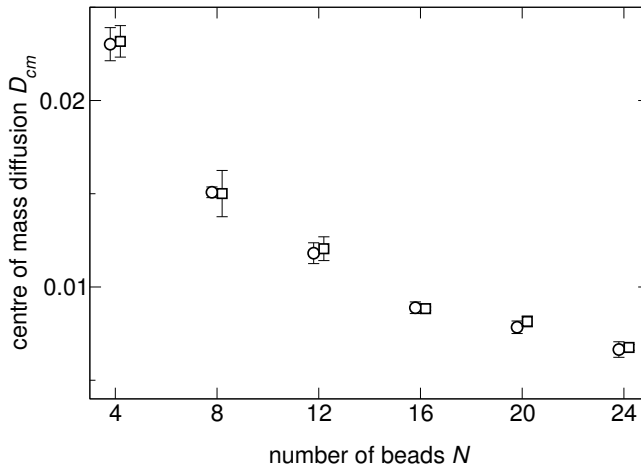


Figure 8.9: Centre of mass translational diffusion coefficient of wormlike chain with $q_{\text{tot}} = 1$ as a function of number of polymer beads N ; with (\circ) and without (\square) hydrodynamic interactions; symbols are shifted horizontally for clarity.

ian dynamics simulations. Indeed, the centre of mass diffusion of the wormlike chains in solution is reproduced.

Now that the translational diffusion is identical for the polymers in solution and in “vacuum”, we focus on the internal dynamics. Figure 8.10 shows the first four Rouse modes for all polymer lengths. The odd behaviour of the $p = 3$ mode, that was already observed in figure 8.8, is apparent, but as the number of polymer beads increases, the effect weakens on the considered time scales. The relaxation times of the first two Rouse modes are shown in figure 8.11, for the wormlike chain in solution and in vacuum. Obviously, the relaxation times of the chains in solution (circles) are larger than those simulated by Brownian dynamics (squares). In other words, polymer conformations stay correlated for a longer time, when the polymer experiences hydrodynamic interactions. Although their absolute values differ, the trend of both relaxation times as a function of the number of beads is similar. For the first Rouse mode we find roughly $\tau_1 \propto N^{2.7}$ and for the second $\tau_2 \propto N^{2.5}$. We cannot relate these numbers to Rouse or Zimm behaviour,^{165,177} because the stretching and bending potentials of the wormlike chain are far from ideal. Furthermore, the Zimm model is derived for an infinite dilution and infinitely long chains; both of which requirements are not satisfied in our simulations. So, we can only conclude that the hydrodynamic interactions slow the relaxation of the internal

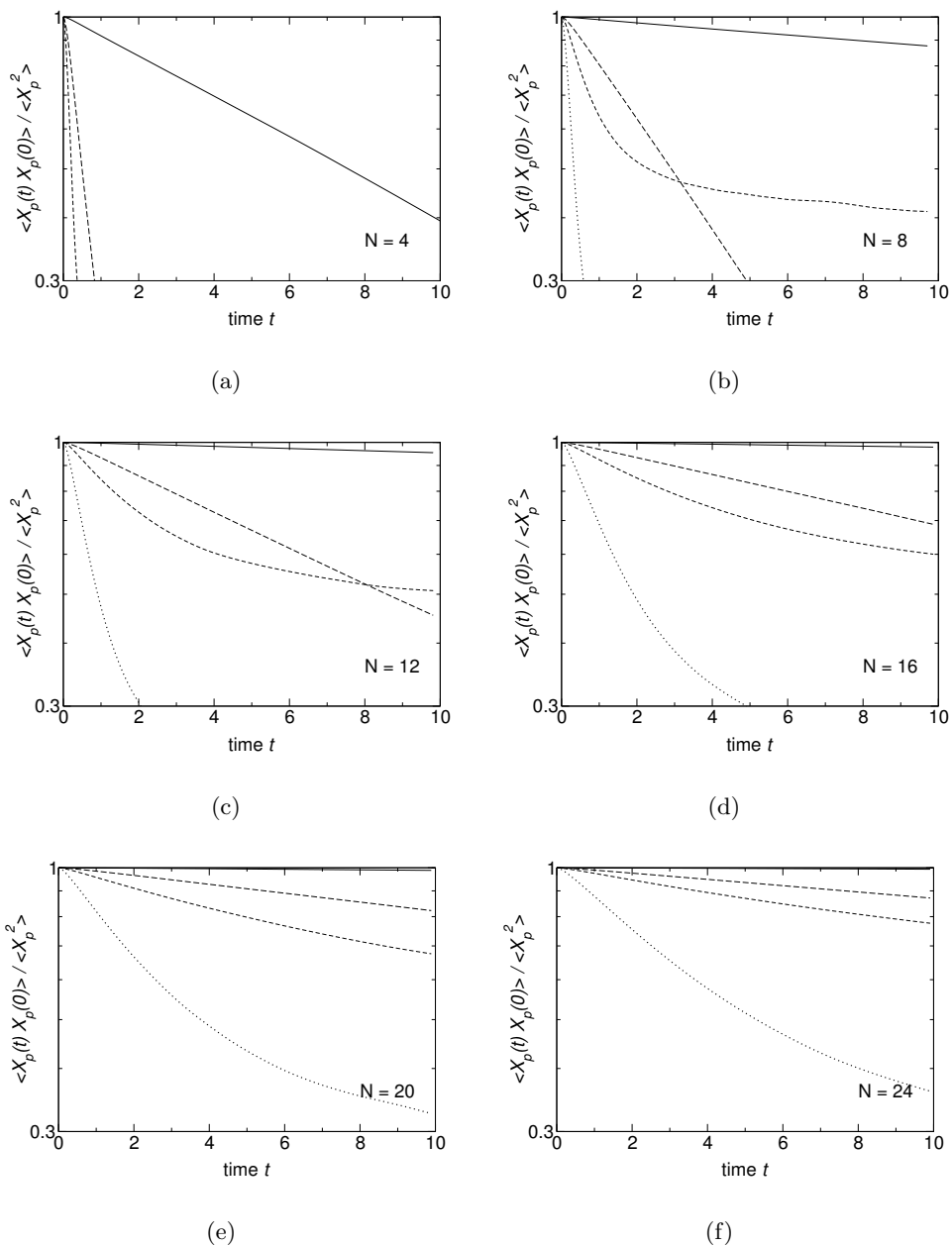


Figure 8.10: Autocorrelation function of Rouse mode X_p with mode $p = 1$ (—), $p = 2$ (---), $p = 3$ (- · -) and $p = 4$ (· · ·), for (a) $N = 4$, (b) $N = 8$, (c) $N = 12$, (d) $N = 16$, (e) $N = 20$ and (f) $N = 24$.

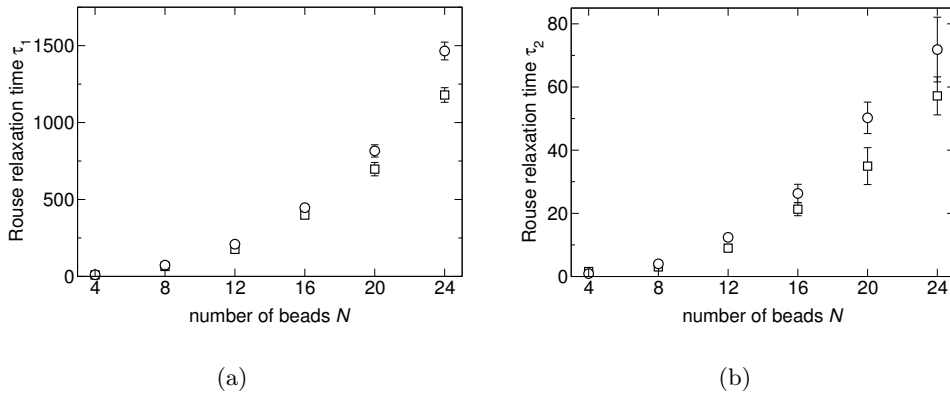


Figure 8.11: Relaxation times of (a) first Rouse mode τ_1 and (b) second Rouse mode τ_2 of wormlike chain with $q_{\text{tot}} = 1$ as a function of number of polymer beads N ; with (\odot) and without (\square) hydrodynamic interactions.

dynamics. The extent of this effect is comparable for all polymer lengths, that were studied.

8.4.3 Performance

Now that the multiple time step algorithm is validated and applied, we turn to its original purpose, the increase of computational efficiency. We will study this for the simulations with identical particles, as well as those with wormlike chains.

Equation 8.1 describes the expected computation time for the multiple time step algorithm. It predicts it to be linearly dependent on the fraction of type 0 (or 1) particles and on the time step ratio. The circles in figure 8.12(a) show the computation time is indeed a linear function of the fraction, although the slope differs from the prediction. When the fractions $x_0 = 0$ (all particles are simulated with $\Delta t = 0.01$) and $x_0 = 1$ (all particles are simulated with $\Delta \tau = 0.001$) are compared, the gain in computation time is not the expected factor of ten, but rather a factor of seven. However, when the fraction x_0 is either 0 or 1, the multiple time step algorithm and its overhead is redundant. The squares in figure 8.12(a) denote the computation time needed for these fractions when the algorithm is not implemented (*i.e.* no distinction between

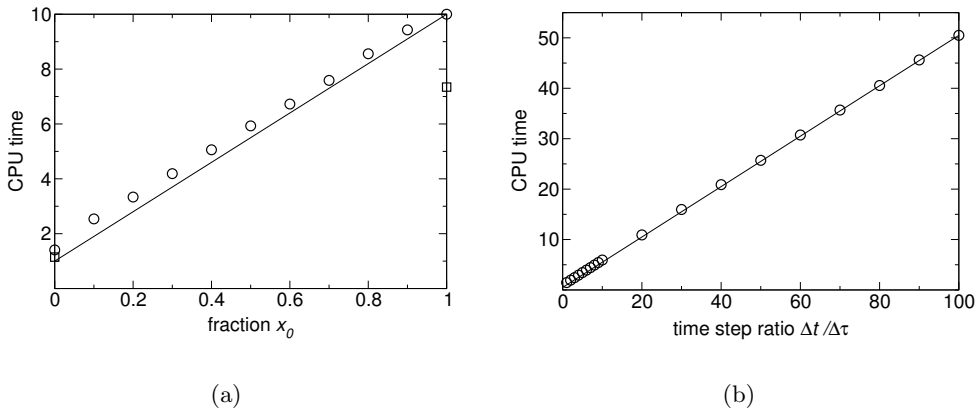


Figure 8.12: Measured (○, □) and theoretical (—) computation time as a function of (a) fraction x_1 of type 1 particles (while $\Delta t / \Delta \tau = 10$) and (b) time step ratio $\Delta t / \Delta \tau$ (while $x_1 = 0.5$).

type 0 and 1 particles). Obviously, these require less time; only when the fraction x_0 is not too large ($x_0 < 0.7$), the multiple time step algorithm is profitable. Similarly, the computation time increases linearly with the time step ratio in figure 8.12(b), but it is higher than predicted. The computation time with a time step ratio of $\Delta t / \Delta \tau = 100$ differs by a factor of 35 instead of 50.5, compared to the computation time without multiple time stepping ($\Delta t / \Delta \tau = 1$). Over the whole range of parameters though, the performance of the multiple time step algorithm promises to reduce the computation time considerably.

Figure 8.13 shows a snapshot of the wormlike chain of $N = 8$ beads. It has the typical “cylinder with caps” shape, as was schematically shown in figure 8.2. The validity of this shape assumption is also demonstrated by comparing measurements to the prediction derived from it. Equation 8.2 predicts 104 type 0 particles surrounding the polymer, while measurements give an average of 105. For a total of $N_{\text{tot}} = 3000$ particles, the fraction of type 0 particles is only $x_0 = 0.03$. With this value and the time step ratio of $\Delta t / \Delta \tau = 70$, equation 8.1 predicts a difference of factor 21 between the simulation time with and without multiple time stepping. The short simulations took on average 2.5h, while 38h were required for the long simulations; a gain of factor 15 in computation time, which is consistent with the result of figure 8.12(a).

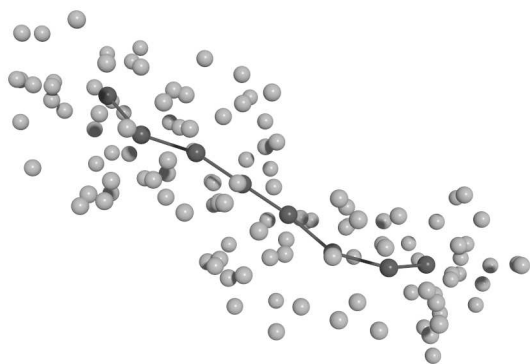


Figure 8.13: Snapshot of wormlike chain with $q_{\text{tot}} = 1$ and $N = 8$ (dark interconnected spheres) with surrounding type 0 particles (light spheres).

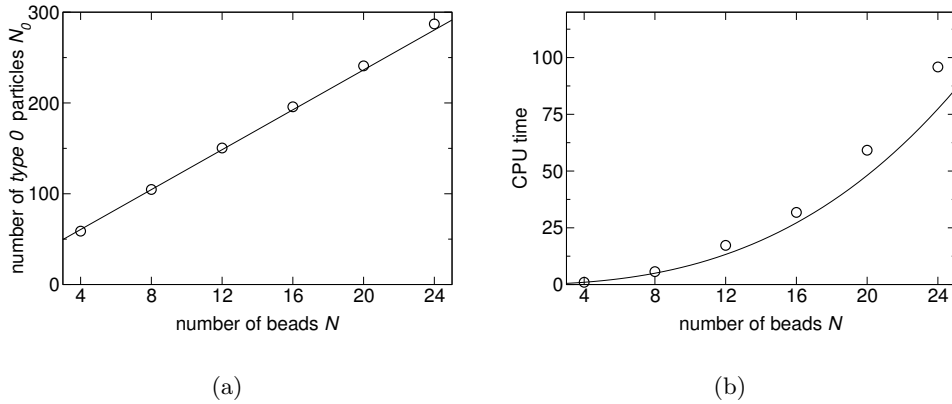


Figure 8.14: Wormlike chain of $q_{\text{tot}} = 1$ of variable number of beads N ; (a) measured number of type 0 particles around polymer (\circ) and prediction by equation 8.2 ($—$); (b) measured computation time (\circ) and prediction by equations 8.2 and 8.12 ($—$).

For the wormlike chains of variable length, we measured the average number of type 0 particles surrounding a polymer. Figure 8.14(a) shows that equation 8.2 predicts this number very accurately. When we use the relations for the small time step (eq. 8.3) and the segment stiffness (eq. 8.4), as well as the simulation setup $N_{\text{tot}} = nN^3$, equation 8.1 for the total computation time becomes:

$$T_{\text{comp}} \propto \left(\frac{\Delta t}{0.001} q_{\text{tot}} (N - 1) N_0 + nN^3 - N_0 \right) n. \quad (8.12)$$

Combined with equation 8.2, this predicted computation time is shown in figure 8.14(b). As expected, the simulations perform worse than predicted. Still, these computation times are a small fraction of the time a simulation would take without the multiple time step algorithm.

8.5 Conclusion

Simulations on systems with disparate time scales are computationally inefficient when the bulk of particles is simulated with an unnecessarily small time step. To avoid this, we introduced a multiple time step algorithm. Where re-

quired, particles are simulated with a small time step, while a larger time step is allowed for the remainder of the particles. We modified the velocity Verlet algorithm to incorporate multiple time stepping in a particle model with Lowe-Andersen thermostat. Simulations with similar particles that only differ in the time step used, showed that the multiple time step algorithm slightly disturbs the density profile, while the velocity profile is not affected. Applied to the wormlike chain model, the multiple time step algorithm does not affect static nor dynamic properties of the polymer. Compared to the results of Brownian dynamics simulations, we were able to assess the effect of hydrodynamic interactions on the polymer. These were found to slow the relaxation of the internal dynamics. The extent of impediment is similar for different polymer lengths. The computation time gained by the multiple time step algorithm depends on the time step ratio required and the fraction of small time step particles. Provided that this fraction is not too large, the multiple time stepping is very profitable. The substantial increase of the computational efficiency promises to access larger time and length scales in mesoscale simulations.

Bibliography

- [1] Hoogerbrugge, P. J.; Koelman, J. M. V. A. *Europhys. Lett.* **1992**, *19*, 155–160.
- [2] Español, P.; Warren, P. B. *Europhys. Lett.* **1995**, *30*, 191–196.
- [3] Español, P. *Phys. Rev. E* **1995**, *52*, 1734–1742.
- [4] Marsh, C. A.; Backx, G.; Ernst, M. H. *Phys. Rev. E* **1997**, *56*, 1676–1691.
- [5] Marsh, C. A.; Backx, G.; Ernst, M. H. *Europhys. Lett.* **1997**, *38*, 411–415.
- [6] Evans, G. T. *J. Chem. Phys.* **1999**, *110*, 1338–1342.
- [7] Ripoll, M.; Ernst, M. H.; Español, P. *J. Chem. Phys.* **2001**, *115*, 7271–7284.
- [8] Serrano, M.; Español, P.; Zúñiga, I. *Comp. Phys. Comm.* **1999**, *121*, 306–308.
- [9] Español, P.; Serrano, M. *Phys. Rev. E* **1999**, *59*, 6340–6347.
- [10] Allen, M. P.; Tildesley, D. J., in *Computer Simulations of Liquids* Oxford Science Publications; Oxford, 1987.
- [11] Groot, R. D.; Warren, P. B. *J. Chem. Phys.* **1997**, *107*, 4423–4435.
- [12] Groot, R. D.; Rabone, K. L. *Biophys. J.* **2001**, *81*, 725–736.
- [13] Kranenburg, M.; Nicolas, J.-P.; Smit, B. *Phys. Chem. Chem. Phys.* **2004**, *6*, 4142–4151.
- [14] Pagonabarraga, I.; Frenkel, D. *Mol. Simulat.* **2000**, *25*, 167–175.
- [15] Pagonabarraga, I.; Frenkel, D. *J. Chem. Phys.* **2001**, *115*, 5015–5026.
- [16] Trofimov, S. Y.; Nies, E. L. F.; Michels, M. A. J. *J. Chem. Phys.* **2002**, *117*, 9383–9394.
- [17] Warren, P. B. *Phys. Rev. E* **2003**, *68*, 066702.
- [18] Lowe, C. P. *Europhys. Lett.* **1999**, *47*, 145–151.
- [19] Frenkel, D.; Smit, B., in *Understanding Molecular Simulation* Academic Press; London, 1996.
- [20] Stoyanov, S. D.; Groot, R. D. *J. Chem. Phys.* **2005**, *122*, 114112.
- [21] Marsh, C. A.; Yeomans, J. M. *Europhys. Lett.* **1997**, *37*, 511–516.
- [22] Pagonabarraga, I.; Hagen, M. H. J.; Frenkel, D. *Europhys. Lett.* **1998**, *42*, 377–382.

- [23] Otter, W. K.den ; Clarke, J. H. R. *Int. J. Mod. Phys. C* **2000**, *11*, 1179–1193.
- [24] Peters, E. A. J. F. *Europhys. Lett.* **2004**, *66*, 311–317.
- [25] Otter, W. K.den ; Clarke, J. H. R. *Europhys. Lett.* **2001**, *53*, 426–431.
- [26] Shardlow, T. *Siam J. Sci. Comput.* **2003**, *24*, 1267–1282.
- [27] Novik, K. E.; Coveney, P. V. *J. Chem. Phys.* **1998**, *109*, 7667–7677.
- [28] Besold, G.; Vattulainen, I.; Karttunen, M.; Polson, J. M. *Phys. Rev. E* **2000**, *62*, 7611–7614.
- [29] Vattulainen, I.; Karttunen, M.; Besold, G.; Polson, J. M. *J. Chem. Phys.* **2002**, *116*, 3967–3979.
- [30] Nikunen, P.; Karttunen, M.; Vattulainen, I. *Comp. Phys. Comm.* **2003**, *153*, 407–423.
- [31] Lees, A. W.; Edwards, S. F. *J. Phys. C* **1972**, *5*, 1921–1929.
- [32] Kong, Y.; Manke, C. W.; Madden, W. G.; Schlijper, A. G. *Int. J. Thermophys.* **1994**, *15*, 1093–1101.
- [33] Revenga, M.; Zúñiga, I.; Español, P.; Pagonabarraga, I. *Int. J. Mod. Phys. C* **1998**, *9*, 1319–1328.
- [34] Revenga, M.; Zúñiga, I.; Español, P. *Comp. Phys. Comm.* **1999**, *121*, 309–311.
- [35] Duong-Hong, D.; Phan-Thien, N.; Fan, X. *Comput. Mech.* **2004**, *35*, 24–29.
- [36] Pivkin, I. V.; Karniadakis, G. E. *J. Comput. Phys.* **2005**, *207*, 114–128.
- [37] Willemsen, S. M.; Hoefsloot, H. C. J.; Iedema, P. D. *Int. J. Mod. Phys. C* **2000**, *11*, 881–890.
- [38] Visser, D. C.; Hoefsloot, H. C. J.; Iedema, P. D. *J. Comput. Phys.* **2005**, *205*, 626–639.
- [39] Bonet Avalos, J.; Mackie, D. *Europhys. Lett.* **1997**, *40*, 141–146.
- [40] Español, P. *Europhys. Lett.* **1997**, *40*, 631–636.
- [41] Ripoll, M.; Español, P. *Int. J. Mod. Phys. C* **1998**, *9*, 1329–1338.
- [42] Willemsen, S. M.; Hoefsloot, H. C. J.; Visser, D. C.; Hamersma, P. J.; Iedema, P. D. *J. Comput. Phys.* **2000**, *162*, 385–394.
- [43] Español, P.; Serrano, M.; Öttinger, H. C. *Phys. Rev. Lett.* **1999**, *83*, 4542–4545.
- [44] Cotter, C. J.; Reich, S. *Europhys. Lett.* **2003**, *64*, 723–729.
- [45] Willemsen, S. M.; Vlugt, T. J. H.; Hoefsloot, H. C. J.; Smit, B. *J. Comput. Phys.* **1998**, *147*, 507–517.

- [46] Wijmans, C. M.; Smit, B.; Groot, R. D. *J. Chem. Phys.* **2001**, *114*, 7644–7654.
- [47] Español, P.; Revenga, M. *Phys. Rev. E* **2003**, *67*, 026705.
- [48] Ripoll, M.; Ernst, M. H. *Phys. Rev. E* **2004**, *71*, 041104.
- [49] Groot, R. D. *J. Chem. Phys.* **2003**, *118*, 11265–11277.
- [50] Novik, K. E.; Coveney, P. V. *Phys. Rev. E* **2000**, *61*, 435–448.
- [51] Visser, D. C.; Hoefsloot, H. C. J.; Iedema, P. D. *J. Comput. Phys.* **2005**, *211*, (in press).
- [52] Trofimov, S. Y.; Nies, E. L. F.; Michels, M. A. J. *J. Chem. Phys.* **2005**, *123*, 144102.
- [53] Jakobsen, A. F. *J. Chem. Phys.* **2005**, *122*, 124901.
- [54] Dzwinel, W.; Yuen, D. A. *Int. J. Mod. Phys. C* **2000**, *11*, 1–25.
- [55] Reith, D.; Meyer, H.; Müller-Plathe, F. *Macromolecules* **2001**, *34*, 2335–2345.
- [56] Guerrault, X.; Rousseau, B.; Farago, J. *J. Chem. Phys.* **2004**, *121*, 6538–6546.
- [57] Maiti, A.; McGrother, S. *J. Chem. Phys.* **2004**, *120*, 1594–1601.
- [58] Schlijper, A. G.; Hoogerbrugge, P. J.; Manke, C. W. *J. Rheol.* **1995**, *39*, 567–579.
- [59] Kong, Y.; Manke, C. W.; Madden, W. G.; Schlijper, A. G. *J. Chem. Phys.* **1997**, *107*, 592–602.
- [60] Groot, R. D. *Langmuir* **2000**, *16*, 7493–7502.
- [61] Vliet, R. E. van ; Hoefsloot, H. C. J.; Hamersma, P. J.; Iedema, P. D. *Macromol. Theory Simul.* **2000**, *9*, 698–702.
- [62] Vliet, R. E. van ; Dreischor, M. W.; Hoefsloot, H. C. J.; Hamersma, P. J.; Iedema, P. D. *Fluid Phase Equilib.* **2002**, *201*, 67–78.
- [63] Vliet, R. E. van ; Hoefsloot, H. C. J.; Hamersma, P. J.; Iedema, P. D. *Polymer* **2003**, *44*, 1757–1763.
- [64] Malfreyt, P.; Tildesley, D. J. *Langmuir* **2000**, *16*, 4732–4740.
- [65] Goujon, F.; Malfreyt, P.; Tildesley, D. J. *ChemPhysChem* **2004**, *5*, 457–464.
- [66] Wijmans, C. M.; Smit, B. *Macromolecules* **2002**, *35*, 7138–7148.
- [67] Irfachsyad, D.; Tildesley, D. J.; Malfreyt, P. *Phys. Chem. Chem. Phys.* **2002**, *4*, 3008–3015.
- [68] Spenley, N. A. *Europhys. Lett.* **2000**, *49*, 534–540.
- [69] Groot, R. D.; Madden, T. J. *J. Chem. Phys.* **1998**, *108*, 8713–8724.

- [70] Groot, R. D.; Madden, T. J.; Tildesley, D. J. *J. Chem. Phys.* **1999**, *110*, 9739–9749.
- [71] Qian, H.-J.; Lu, Z.-Y.; Chen, L.-J.; Li, Z.-S.; Sun, C.-C. *J. Chem. Phys.* **2005**, *122*, 184907.
- [72] AlSunaidi, A.; Otter, W. K.; Clarke, J. H. R. *Phil. Trans. R. Soc. Lond. A* **2004**, *362*, 1773–1781.
- [73] Levine, Y. K.; Gomes, A. E.; Martins, A. S.; Polimeno, A. *J. Chem. Phys.* **2005**, *122*, 144902.
- [74] Pan, G.; Manke, C. W. *Int. J. Mod. Phys. B* **2003**, *17*, 231–235.
- [75] Padding, J. T.; Briels, W. J. *J. Chem. Phys.* **2001**, *115*, 2846–2859.
- [76] Boek, E. S.; Coveney, P. V.; Lekkerkerker, H. N. W.; Schoot, P. van der *Phys. Rev. E* **1997**, *55*, 3124–3133.
- [77] Dzwinel, W.; Yuen, D. A. *J. Colloid Interf. Sci.* **2000**, *225*, 179–190.
- [78] Dzwinel, W.; Yuen, D. A.; Boryczko, K. *J. Mol. Model.* **2002**, *8*, 33–43.
- [79] Martys, N. S. *J. Rheol.* **2005**, *49*, 401–424.
- [80] Coveney, P. V.; Novik, K. E. *Phys. Rev. E* **1996**, *54*, 5134–5141.
- [81] Novik, K. E.; Coveney, P. V. *Int. J. Mod. Phys. C* **1997**, *8*, 909–918.
- [82] Rekvig, L.; Kranenburg, M.; Hafskjold, B.; Smit, B. *Europhys. Lett.* **2003**, *63*, 902–907.
- [83] Rekvig, L.; Kranenburg, M.; Vreede, J.; Hafskjold, B.; Smit, B. *Langmuir* **2003**, *19*, 8195–8205.
- [84] Rekvig, L.; Hafskjold, B.; Smit, B. *J. Chem. Phys.* **2004**, *120*, 4897–4905.
- [85] Rekvig, L.; Hafskjold, B.; Smit, B. *Phys. Rev. Lett.* **2004**, *92*, 116101.
- [86] Rekvig, L.; Hafskjold, B.; Smit, B. *Langmuir* **2004**, *20*, 11583–11593.
- [87] Jury, S.; Bladon, P.; Cates, M.; Krishna, S.; Hagen, M.; Ruddock, N.; Warren, P. *Phys. Chem. Chem. Phys.* **1999**, *1*, 2051–2056.
- [88] Prinsen, P.; Warren, P. B.; Michels, M. A. J. *Phys. Rev. Lett.* **2002**, *89*, 148302.
- [89] Li, D.-W.; Liu, X. Y.; Feng, Y. P. *J. Phys. Chem. B* **2004**, *108*, 11206–11213.
- [90] Yamamoto, S.; Hyodo, S. *J. Chem. Phys.* **2005**, *122*, 204907.
- [91] Yamamoto, S.; Maruyama, Y.; Hyodo, S. *J. Chem. Phys.* **2002**, *116*, 5842–5849.
- [92] Laradji, M.; Kumar, P. B. Sunil *Phys. Rev. Lett.* **2004**, *93*, 198105.
- [93] Ortiz, V.; Nielsen, S. O.; Discher, D. E.; Klein, M. L.; Lipowsky, R.; Shillcock, J. *J. Phys. Chem. B* **2005**, *109*, 17708–17714.

- [94] Ayton, G.; Voth, G. A. *Biophys. J.* **2002**, *83*, 3357–3370.
- [95] Kranenburg, M.; Venturoli, M.; Smit, B. *J. Phys. Chem. B* **2003**, *107*, 11491–11501.
- [96] Shillcock, J. C.; Lipowsky, R. *J. Chem. Phys.* **2002**, *117*, 5048–5061.
- [97] Kranenburg, M.; Venturoli, M.; Smit, B. *Phys. Rev. E* **2003**, *67*, 060901.
- [98] Illya, G.; Lipowsky, R.; Shillcock, J. C. *J. Chem. Phys.* **2005**, *122*, 244901.
- [99] Kranenburg, M.; Laforge, C.; Smit, B. *Phys. Chem. Chem. Phys.* **2004**, *6*, 4531–4534.
- [100] Kranenburg, M.; Smit, B. *J. Phys. Chem. B* **2005**, *109*, 6553–6563.
- [101] Kranenburg, M.; Vlaar, M.; Smit, B. *Biophys. J.* **2004**, *87*, 1596–1605.
- [102] Li, D.-W.; Liu, X. Y. *J. Chem. Phys.* **2005**, *122*, 174909.
- [103] Jakobsen, A. F.; Mouritsen, O. G.; Besold, G. *J. Chem. Phys.* **2005**, *122*, 204901.
- [104] Dzwinel, W.; Yuen, D. A. *Mol. Simulat.* **1999**, *22*, 369–395.
- [105] Clark, A. T.; Lal, M.; Ruddock, J. N.; Warren, P. B. *Langmuir* **2000**, *16*, 6342–6350.
- [106] Willemsen, S. M.; Hoefsloot, H. C. J.; Iedema, P. D. *J. Stat. Phys.* **2002**, *107*, 53–65.
- [107] Chen, S.; Phan-Thien, N.; Fan, X.-J.; Khoo, B. C. *J. Non-Newton. Fluid Mech.* **2004**, *118*, 65–81.
- [108] Symeonidis, V.; Karniadakis, G. E.; Caswell, B. *Phys. Rev. Lett.* **2005**, *95*, 076001.
- [109] Kim, J. M.; Philips, R. J. *Chem. Eng. Sci.* **2004**, *59*, 4155–4168.
- [110] Keaveny, E. E.; Pivkin, I. V.; Maxey, M.; Karniadakis, G. E. *J. Chem. Phys.* **2005**, *123*, 104107.
- [111] Moore, G. E. *Electronics* **1965**, *38*, 19 April.
- [112] Bekker, H.; Dijkstra, E. J.; Renardus, M. K. R.; Berendsen, H. J. C. *Mol. Sim.* **1995**, *14*, 137–151.
- [113] Monaghan, J. J. *Annu. Rev. Astron. Astrophys.* **1992**, *30*, 543–574.
- [114] Malevanets, A.; Kapral, R. *J. Chem. Phys.* **1999**, *110*, 8605–8613.
- [115] Palmer, B. J. *Phys. Rev. E* **1994**, *49*, 359–366.
- [116] Heemels, M. W.; Bakker, A. F.; Lowe, C. P. *Prog. Colloid Polym. Sci.* **1998**, *110*, 150.
- [117] Hess, B. *J. Chem. Phys.* **2002**, *116*, 209–217.
- [118] Ciccotti, G.; Jacucci, G.; McDonald, I. R. *J. Stat. Phys.* **1979**, *21*, 1–22.

- [119] Arya, G.; Maginn, E. J.; Chang, H.-C. *J. Chem. Phys.* **2000**, *113*, 2079–2087.
- [120] Müller-Plathe, F. *Phys. Rev. E* **1999**, *59*, 4894–4898.
- [121] Trozzi, C.; Ciccotti, G. *Phys. Rev. A* **1984**, *29*, 916–925.
- [122] Kikuchi, N.; Pooley, C. M.; Ryder, J. F.; Yeomans, J. M. *J. Chem. Phys.* **2003**, *119*, 6388–6395.
- [123] Varnik, F.; Binder, K. *J. Chem. Phys.* **2002**, *117*, 6336–6349.
- [124] Allahyarov, E.; Gompper, G. *Phys. Rev. E* **2002**, *66*, 036702.
- [125] Kauzlarić, D.; Greiner, A.; Korvink, J. G. *Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show* **2004**, *2*, 454–457.
- [126] Bird, R. B.; Stewart, W. E.; Lightfoot, E. N., in *Transport Phenomena* Wiley & Sons; New York, 1960.
- [127] Soddemann, T.; Dünweg, B.; Kremer, K. *Phys. Rev. E* **2003**, *68*, 046702.
- [128] Meier, K.; Laesecke, A.; Kabelac, S. *J. Chem. Phys.* **2004**, *121*, 3671–3687.
- [129] Nielsen, S. O.; Lopez, C. F.; Srivinas, G.; Klein, M. L. *J. Phys.: Condens. Matter* **2004**, *16*, R481–R512.
- [130] Anderson, H. *J. Comput. Phys.* **1983**, *52*, 24.
- [131] McCoy, J. D.; Curro, J. G. *Macromolecules* **1998**, *31*, 9362–9368.
- [132] Akkermans, R. L. C.; Briels, W. J. *J. Chem. Phys.* **2000**, *113*, 6409–6422.
- [133] Fukunaga, H.; Takimoto, Y.; Doi, M. *J. Chem. Phys.* **2002**, *116*, 8183–8190.
- [134] Müller-Plathe, F. *Chem. Phys. Chem* **2002**, *3*, 754–769.
- [135] Shelley, J. C.; Shelley, M. Y.; Reeder, R. C.; Bandyopadhyay, S.; Klein, M. L. *J. Phys. Chem. B* **2001**, *105*, 4464–4470.
- [136] Liwo, A.; Oldziej, S.; Pincus, M. R.; Wawak, R. J.; Rackovsky, S.; Scheraga, H. A. *J. Comput. Chem.* **1997**, *18*, 849–873.
- [137] Flekkøy, E. G.; Coveney, P. V. *Phys. Rev. Lett.* **1999**, *83*, 1775–1778.
- [138] Backer, J. A.; Lowe, C. P.; Hoefsloot, H. C. J.; Iedema, P. D. *J. Chem. Phys.* **2005**, *122*, 154503.
- [139] Koumoutsakos, P. *Annu. Rev. Fluid Mech.* **2005**, *37*, 457–487.
- [140] Español, P. *Phys. Rev. E* **1998**, *57*, 2930–2948.
- [141] Lucy, L. *Astron. J.* **1977**, *82*, 1013.
- [142] Malevanets, A.; Kapral, R. *J. Chem. Phys.* **2000**, *112*, 7260–7269.
- [143] Malevanets, A.; Yeomans, J. M. *Europhys. Lett.* **2000**, *52*, 231–237.
- [144] O’Connell, S. T.; Thompson, P. A. *Phys. Rev. E* **1995**, *52*, 5792–5795.

- [145] Hadjiconstantinou, N. G.; Patera, A. T. *Int. J. Mod. Phys. C* **1997**, *8*, 967–976.
- [146] Garcia, A. L.; Bell, J. B.; Crutchfield, W. Y.; Alder, B. J. *J. Comp. Phys.* **1999**, *154*, 134–155.
- [147] Flekkøy, E. G.; Wagner, F.; Feder, J. *Europhys. Lett.* **2000**, *52*, 271–276.
- [148] Flekkøy, E. G.; Coveney, P. V.; De Fabritiis, G. *Phys. Rev. E* **2000**, *62*, 2140–2157.
- [149] Serrano, M.; Español, P. *Phys. Rev. E* **2001**, *64*, 046115.
- [150] De Fabritiis, G.; Coveney, P. V. *Comput. Phys. Commun.* **2003**, *153*, 209–226.
- [151] Serrano, M.; De Fabritiis, G.; Español, P.; Flekkøy, E. G.; Coveney, P. V. *J. Phys. A: Math. Gen.* **2002**, *35*, 1605–1625.
- [152] Marko, J. F.; Siggia, E. D. *Macromolecules* **1995**, *28*, 8759–8770.
- [153] Kratky, O.; Porod, G. *Recl. Trav. Chim. Pays-Bas* **1949**, *68*, 1106–1122.
- [154] Yamakawa, H., in *Modern theory of polymer solutions* Harper & Row; New York, 1971.
- [155] Fixman, M.; Kovac, J. *J. Chem. Phys.* **1973**, *58*, 1564–1568.
- [156] Bustamante, C.; Marko, J. F.; Siggia, E. D.; Smith, S. *Science* **1994**, *265*, 1599–1600.
- [157] Ha, B.-Y.; Thirumalai, D. *J. Chem. Phys.* **1997**, *106*, 4243–4247.
- [158] Samuel, J.; Sinha, S. *Phys. Rev. E* **2002**, *66*, 050801.
- [159] Dhar, A.; Chaudhuri, D. *Phys. Rev. Lett.* **2002**, *89*, 065502.
- [160] Winkler, R. G. *J. Chem. Phys.* **2003**, *118*, 2919–2928.
- [161] Ohm, V. *to be published* , .
- [162] Lowe, C. P. *Fut. Gen. Comp. Sys.* **2001**, *17*, 853–862.
- [163] Rubinstein, M.; Colby, R. H., in *Polymer Physics* Oxford University Press; Oxford, 2003.
- [164] Fynewever, H.; Yethiraj, A. *J. Chem. Phys.* **1998**, *108*, 1636–1644.
- [165] Doi, M.; Edwards, S. F., in *The Theory of Polymer Dynamics* Oxford Science Publications; Oxford, 1986.
- [166] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P., in *Numerical Recipes in C*, 2nd ed. Cambridge University Press; Cambridge, 1992.
- [167] Harnau, L.; Winkler, R. G.; Reineker, P. *J. Chem. Phys.* **1996**, *104*, 6355–6368.
- [168] Hansen, J. P.; McDonald, I. R., in *Theory of Simple Liquids* Academic Press; London, 1986.

-
- [169] Schlick, T.; Barth, E.; Mandziuk, M. *Annu. Rev. Biophys. Biomol. Struct.* **1997**, *26*, 181–222.
- [170] Tuckerman, M.; Berne, B. J.; Rossi, A. *J. Chem. Phys.* **1991**, *94*, 1465–1469.
- [171] Tuckerman, M.; Berne, B. J.; Martyna, G. J. *J. Chem. Phys.* **1991**, *94*, 6811–6815.
- [172] Tuckerman, M.; Berne, B. J.; Martyna, G. J. *J. Chem. Phys.* **1992**, *97*, 1990–2001.
- [173] Grubmüller, H.; Heller, H.; Windemuth, A.; Schulten, K. *Mol. Simul.* **1991**, *6*, 121–142.
- [174] Tuckerman, M.; Berne, B. J. *J. Chem. Phys.* **1993**, *98*, 7301–7318.
- [175] Symeonidis, V.; Karniadakis, G. E.; Caswell, B. *Comput. Sci. Eng.* **2005**, *7*, 39–46.
- [176] Zatorovsky, A. V.; Lisy, V. *J. Mol. Liq.* **2003**, *105*, 289–294.
- [177] Attig, N.; Binder, K.; Grubmüller, H.; Kremer, K., in *Computational Soft Matter: From Synthetic Polymers to Proteins* Graphische Betriebe; Forschungszentrum Jülich, 2004.

Summary

The research presented in this thesis involves the combination of different time and length scales in Dissipative Particle Dynamics (DPD). This model in itself already bridges the gap between microscale and macroscale processes. In a liquid for instance, molecules behave chaotically on a microscale, whereas at a macroscale hydrodynamic behaviour appears. However, many complex phenomena occur on a level between these extremes, the mesoscale. On this level of description, macroscale models provide too little detail and microscale models are computationally too expensive. DPD is designed for precisely these mesoscale simulations. It is a particle model that can describe processes on a rather large scale without losing the chaotic character of the underlying molecules. Still, the mesoscale simulations that require a large number of particles, can be computationally demanding. Therefore, it would be beneficial to make a distinction between regions that need detailed simulation, and bulk areas that are of lesser interest. In the first region, a small scale model provides the desired information, while in the other a large scale model simulates the remainder of the system in less detail. In this thesis, we combine these differently scaled models within the framework of DPD. Bridging the time and length scale differences in such a way, will increase the computational efficiency of DPD.

Chapter 2 gives a general overview of Dissipative Particle Dynamics (DPD), as well as its proposed improvements and reported applications. Introduced only fourteen years ago, it is still a relatively young modelling technique. As a particle model, it describes a fluid as a collection of point masses, each of which behaves according to Newton's laws of motion. A particle represents a fluid element, that interacts with its neighbouring fluid elements. Each particle pair within a specified cut-off radius, experiences three forces. Firstly, a conservative force acts on the particles, which is usually soft repulsive. This means particles always repel each other, but the repulsion never goes to infinity. This allows for larger time steps, compared to microscale models. Secondly, a dissipative force acts on the particles that reduces their relative velocity. Due to

this friction, a fast particle is slowed down by a slower particle, which speeds up itself. Thirdly, a random force “kicks” the particle pair, while conserving the total momentum. The dissipative force drains energy from the system, whereas the random force supplies it. These counteracting forces balance the energy in the system, and together they act as a thermostat. It can be demonstrated that the DPD equations of motion agree with the Navier-Stokes equation, thus connecting DPD to the macroscale.

As we will focus on computational efficiency throughout this thesis, we will first examine in chapter 3 how the DPD model can be efficiently implemented. Three different implementations are considered, *viz.* the cell list, the Verlet list and the combination of both. They differ in the way they select particle pairs from all possible combinations, which accounts for a large part of the computation time. For each implementation, we derive an expression for the total computation time. Neglecting the smaller contributions, we reduce the expressions to simplified theoretical values, that are only dependent on the input parameters. With these values one can accurately predict the optimal value for the Verlet radius and reasonably identify the fastest implementation. In general, DPD simulations involve large numbers of particles in the liquid regime (where the diffusive motion of the particles is limited). Under these circumstances, the implementation that combines the cell and Verlet list, is usually the most efficient from a computational point of view.

Not only the implementation of the model itself, but also accurate measurements of the system properties can considerably lengthen the computation time. The viscosity is in this respect notorious, as it is determined from the time-consuming measurement of shear stresses. In chapter 4 we develop a new technique for viscosity measurements that is based on counter-flowing Poiseuille flows. This “periodic Poiseuille flow method” does not require shear stress measurements, as the system average of the velocity in the flow direction directly provides the viscosity of the fluid. Moreover, the application of opposing external forces enables the use of ordinary periodic boundary conditions. We compare this method to the shear flow method and the stress autocorrelation method, at different parameter settings. All methods give comparable values for the average of the measured viscosity, but the results of the periodic Poiseuille flow method are more accurate than those obtained by both stress methods. Thus, to achieve a desired accuracy, our method requires a shorter computation time. In the work presented in this thesis we use this new method for viscosity measurements.

Now we turn to the combination of length scales within the DPD model. For this purpose the basic model equations need to be complemented with a coarse-grained version, that describes the same system with less particles. In chapter 5 we develop this coarse-graining procedure. It aims to keep the mass density, pressure, temperature and shear viscosity constant, while sacrificing the compressibility and diffusivity. For the particle mass, cut-off radius and phase repulsion we derive straightforward relations. The friction coefficient on the other hand, needs to be tuned to the shear viscosity. In chapter 6 these descriptions with different length scales are combined into one system to enable simultaneous simulation. They are coupled via an overlap region, where both types of particles coexist. At the edges particles are locally refined or coarse-grained, thus allowing for mass and momentum transfer between the two different scales. The overlap regions induce a slight perturbation of the density, but the pressure and temperature profiles are unaffected. The combined system reproduces the flow behaviour of a normal system, irrespective of the flow direction. Regarding the computational effort, the combined system is more efficient than a system without coarse-grained region. A note of concern though, is that due to the considerable size of the overlap regions, only large systems with large bulk areas are suitable for this combination technique.

Just as the combination of length scales, we would like to combine different time scales in one system. Now the small scale region is not characterized by a larger number of particles, but by smaller time steps. Before developing such a multiple time step algorithm, we first describe a discrete wormlike chain model in chapter 7, that will serve as a relevant problem to approach with different time steps. The continuous wormlike chain model (or Kratky-Porod model) describes a single semi-flexible polymer, that has a finite extensibility. A discrete wormlike chain model approximates its continuous counterpart, by discretizing the chain in a number of particles and defining a stretching potential between beads and a bending potential between links. We determined the bending constant by applying the freely rotating model, which proved to be significantly better than previous methods. The resultant model accurately reproduces the large-wavelength behaviour and the second and fourth moment of the continuous model, while its application ranges from the flexible to stiff limit. Such a wormlike chain in solution is a very appropriate problem to treat with different time steps. As the potential of the wormlike chain model approaches infinity at maximal extension, this greatly limits the maximal time step for the simulation, whereas many solvent particles could be simulated with considerably larger time steps. A system that combines the different time steps would circumvent this constraint. Unfortunately, this proves to be impossible

in DPD, because the time step is intrinsically intertwined with the thermostat equations. Instead, we use the Lowe-Andersen thermostat for the development of the multiple time step algorithm in chapter 8. It simulates the polymer and surrounding solvent particles with a small time step, while a larger time step is allowed for the bulk of the solvent. Using multiple timestepping does not affect the static nor dynamic properties of the polymer. With the new algorithm we were able to assess the effect of hydrodynamic interactions on the polymer, by comparison to the results of Brownian dynamics simulations (without hydrodynamic interactions). These interactions were found to slow the relaxation of the internal dynamics, independent of the polymer length. How much computation time is gained by the multiple time step algorithm, depends on the time step ratio required and the fraction of small time step particles. Provided that this fraction is not too large, multiple time stepping substantially increases the computational efficiency.

Finally, it would be interesting to combine different length scales as well as different time scales in one system, thus uniting both approaches presented in this thesis. Although one needs to abandon the DPD thermostat to do so (and use the Lowe-Andersen thermostat instead), the same techniques apply. This would increase the computational efficiency of mesoscale particle models even more and would allow access to ever larger time and length scales in simulations. Hopefully, the work in this thesis will have contributed to this.

Samenvatting

Het onderzoek in dit proefschrift beschrijft de combinatie van verschillende tijd- en lengteschalen in *Dissipative Particle Dynamics* (DPD). Zelf overbrugt dit model ook al het verschil tussen processen die zich op micro- en macroschaal afspelen. In een vloeistof bijvoorbeeld, gedragen moleculen zich op microschaal chaotisch, terwijl hydrodynamisch gedrag zich pas op macroschaal openbaart. Echter, veel complexe processen manifesteren zich in een gebied tussen deze twee uitersten, de mesoschaal. Op dit niveau verschaffen macroschaalmodellen te weinig informatie, terwijl microschaalmodellen teveel rekentijd vergen. Juist voor deze mesoschaalsimulaties is DPD ontwikkeld. Het is een deeltjesmodel dat processen op redelijk grote schaal kan beschrijven zonder het chaotische karakter van de moleculen uit het oog te verliezen. Niettemin kan voor mesoschaalsimulaties met grote aantallen deeltjes veel rekentijd nodig zijn. Daarom is het nuttig om een onderscheid te maken tussen gebieden die gedetailleerde simulatie behoeven, en bulkgebieden die minder interessant zijn. In het eerste gebied zorgt een kleinschalig model voor de gedetailleerde informatie, terwijl een grootschalig model het andere gebied voor zijn rekening neemt. In dit proefschrift voegen we deze modellen van verschillende schaal samen in DPD. Het overbruggen van verschillen in tijd- en lengteschaal zal zo de computationele efficiëntie van DPD ten goede komen.

Hoofdstuk 2 geeft een algemeen overzicht van DPD, evenals de gepubliceerde verbeteringen en toepassingen. DPD is slechts veertien jaar geleden ontwikkeld, en is daarmee relatief jong op modelgebied. Het is een deeltjesmodel, dat bestaat uit een verzameling puntmassa's die zich voortbewegen volgens de wetten van Newton. Een deeltje stelt een vloeistofelement voor, dat beïnvloed wordt door zijn buurelementen. Elk deeltjespaar dat zich binnen een bepaalde *cut-off* radius bevindt, ondervindt drie krachten. Ten eerste werkt een conservatieve kracht op de deeltjes, die meestal *soft repulsive* is. Dit betekent dat deeltjes elkaar altijd afstoten, maar die afstoting wordt nooit oneindig groot. Hierdoor kan de simulatie grotere tijdstappen maken, dan met microschaalmodellen. Ten tweede werkt een dissipatieve kracht op de deeltjes,

die hun relatieve snelheidsverschil vermindert. Door deze wrijving zal een snel deeltje worden afgeremd door een langzamer deeltje, dat zelf wordt versneld. Ten derde “schopt” een randomkracht het deeltjespaar, onder behoud van impuls. De dissipatieve kracht onttrekt energie aan het systeem, terwijl de randomkracht juist energie levert. Samen houden ze de energie van het systeem constant en functioneren ze als thermostaat. Het is aangetoond dat de modelvergelijkingen van DPD overeenstemmen met de Navier-Stokes-vergelijking, zodat DPD gerelateerd kan worden aan de macroschaal.

Daar de computationele efficiëntie als een rode draad door dit proefschrift loopt, onderzoeken we in hoofdstuk 3 eerst hoe het DPD-model efficiënt geïmplementeerd kan worden. Drie verschillende implementaties komen aan bod, *nl.* de *cell list*, de *Verlet list* en een combinatie van de twee. Ze verschillen in de manier waarop ze deeltjesparen kiezen uit alle mogelijke combinaties, wat een groot deel van de rekentijd vraagt. Voor elke implementatie leiden we een uitdrukking af voor de totale rekentijd. Als we de kleinere bijdragen verwaarlozen, versimpelen ze tot theoretische waardes, die alleen afhangen van de inputparameters. Met deze theorie kan de optimale waarde van de Verlet-radius nauwkeurig voorspeld worden en kan de snelste implementatie redelijk goed bepaald worden. Over het algemeen omvatten DPD-simulaties grote aantallen deeltjes die zich in het vloeistofregime bevinden (waar diffusie een beperkte rol speelt, vergeleken met gassen). Onder deze omstandigheden is meestal de implementatie die de *cell* en *Verlet list* combineert, het meest efficiënt vanuit computationeel oogpunt.

Niet alleen de modelimplementatie zelf, maar ook het nauwkeurig meten van systeemeigenschappen, kan de rekentijd behoorlijk doen oplopen. De viscositeit is een berucht voorbeeld, omdat deze eigenschap bepaald wordt uit de tijdrovende meting van schuifspanningen. In hoofdstuk 4 ontwikkelen we een nieuwe techniek om viscositeit te meten, die is gebaseerd op twee Poiseuillestromen in tegengestelde richting. Deze *periodic Poiseuille flow* methode omzeilt de schuifspanningsmetingen, doordat uit het gemiddelde van de snelheid in de stromingsrichting direct de viscositeit bepaald kan worden. Bovendien volstaan gewone periodieke randvoorwaarden dankzij de tegengestelde krachten. We vergelijken deze methode met de *shear flow* methode en de *stress autocorrelation* methode, bij verschillende parameters. Alle methodes geven vergelijkbare gemiddeldes voor de gemeten viscositeit, maar de resultaten van de *periodic Poiseuille flow* methode zijn nauwkeuriger dan die van de andere methodes. Kortom, om een bepaalde nauwkeurigheid te bereiken, heeft onze methode minder rekentijd nodig. In dit proefschrift gebruiken we deze nieuwe methode

om de viscositeit te meten.

Vervolgens richten we de aandacht op het combineren van lengteschalen in het DPD-model. Hiervoor moeten de basismodelvergelijkingen aangevuld worden met een grootschaligere variant, die hetzelfde systeem beschrijft met minder deeltjes. In hoofdstuk 5 ontwikkelen we deze *coarse-graining* procedure. Het doel ervan is de massadichtheid, druk, temperatuur en viscositeit gelijk te houden, terwijl eisen aan de compressibiliteit en diffusie losgelaten worden. Voor de deeltjesmassa, *cut-off* radius en afstotingsparameter vinden we eenvoudige uitdrukkingen. De frictiecoëfficiënt daarentegen moet via “trial and error” aangepast worden om de juiste viscositeit te leveren. Het klein- en grootschalige DPD-model worden in hoofdstuk 6 samengevoegd tot één systeem om gelijktijdige simulatie mogelijk te maken. Ze worden gekoppeld via een overlapgebied, waarin beide type deeltjes naast elkaar bestaan. Aan de randen worden lokaal ofwel kleinschalige deeltjes samengevoegd tot grotere, ofwel grootschalige deeltjes gesplitst in kleinere. Op deze manier wordt massa- en impulstransport van de ene naar de andere schaal mogelijk gemaakt. De overlapgebieden verstoren de dichtheid enigszins, maar de druk- en temperatuurprofielen worden niet beïnvloed. Het gecombineerde systeem vertoont hetzelfde hydrodynamisch gedrag als een normaal (kleinschalig) systeem, onafhankelijk van de stromingsrichting. Computationeel gezien is het gecombineerde systeem echter efficiënter dan een systeem zonder grootschalig gebied. Men dient echter terdege te beseffen, dat door de aanzienlijke afmetingen van de overlapgebieden, alleen grote systemen met grote bulkgebieden zich lenen voor deze combinatietechniek.

Net zoals de combinatie van lengteschalen, zouden we ook tijdschalen in één systeem willen combineren. Dan wordt het kleinschalige gebied niet gekenmerkt door een groter aantal deeltjes, maar door kleinere tijdstappen. Voor dat we een dergelijk *multiple time step* algoritme ontwikkelen, behandelen we eerst een discreet *wormlike chain* model in hoofdstuk 7, dat later als relevant voorbeeld zal dienen om met verschillende tijdstappen aan te pakken. Het continue *wormlike chain* model (of Kratky-Porod-model) beschrijft een enkele semiflexibele polymeerstreng, die slechts beperkt uitgerekt kan worden. Het discrete model benadert de continue tegenhanger door de streng op te delen in een aantal deeltjes. Tussen die deeltjes wordt een potentiaal gedefinieerd die de strekking en buiging bepaalt. De buigconstante volgt uit de toepassing van het *freely rotating* model en is significant beter dan die uit andere methodes. Het gedrag bij grote golfengtes en het tweede en vierde moment van het continue model worden namelijk nauwkeurig gereproduceerd door ons model. Bovendien

is het model geldig in het complete gebied tussen de flexibele en rigide limiet. Een oplossing van deze *womlike chain* in andere deeltjes is uitermate geschikt om te benaderen met verschillende tijdstappen. Omdat de potentiaal van het model naar oneindig gaat bij maximale uitrekking, kan de simulatie slechts kleine tijdstappen maken, terwijl voor de meeste oplosdeeltjes een veel grotere tijdstap mogelijk zou zijn. Een systeem dat deze verschillende tijdschalen combineert, zou deze beperking omzeilen. Dit is echter niet mogelijk met DPD, omdat de tijdstap onlosmakelijk verstrengeld is in de thermostaatvergelijkingen. In plaats daarvan gebruiken we de Lowe-Andersen-thermostaat voor het ontwikkelen van het *multiple time step* algoritme in hoofdstuk 8. Hiermee worden de polymeer en de dichtstbijzijnde oplosdeeltjes met een kleine tijdstap gesimuleerd, terwijl een grotere tijdstap gebruikt wordt voor de bulk van het oplosmiddel. Het algoritme heeft geen enkele invloed op de statische of dynamische eigenschappen van het polymeer. Met behulp van het algoritme onderzoeken we het effect van hydrodynamische interacties op het polymeer, door de resultaten te vergelijken met die van *Brownian dynamics* simulaties (zonder hydrodynamische interacties). Deze interacties vertragen de relaxatie van de interne dynamica, onafhankelijk van de lengte van het polymeer. Hoeveel rekentijd het nieuwe algoritme uitspaart, hangt af van de verhouding tussen de kleine en grote tijdstap en de verhouding tussen de aantallen kleine en grote deeltjes. Mits deze verhoudingen niet te groot zijn, verhoogt het *multiple time step* algoritme de computationele efficiëntie in hoge mate.

Ten slotte zou het natuurlijk boeiend zijn om beide technieken uit dit proefschrift samen toe te passen, door zowel verschillende lengteschalen als verschillende tijdschalen in één systeem te combineren. Hoewel we dan moeten afstappen van de DPD-thermostaat (en de Lowe-Andersen-thermostaat moeten gebruiken), zou dit niets veranderen aan de aanpak. Het zou de computationele efficiëntie nog meer ten goede komen en steeds grotere tijd- en lengteschalen bereikbaar maken voor mesoschaalsimulaties. Hopelijk heeft het werk in dit proefschrift daaraan bijgedragen.

Dankwoord

Woei! Het is af! Het is onmogelijk onder woorden te brengen hoe blij ik daarmee ben. Maar gelukkig kan ik er wel heel veel mensen voor bedanken. Allereerst natuurlijk mijn promotor Piet, die altijd geïnteresseerd was en mij de vrijheid gaf om mijn eigen pad te kiezen. Huub wil ik bedanken voor zijn tomeloze motivatie, verkapt in zijn dagelijkse kamerbezoek (alhoewel er weinig verkapt is aan “Hee Backer, is dat proefschrift nou eens af?”). Zijn praktische instelling - alleen hij kon de onvolprezen bottebijlmethode verzinnen - was vaak een verademing tussen het theoretische geneuzel. On those more fundamental issues, Christopher has been of great help during the last part of my PhD. Voor de begeleiding in de begintijd, ben ik veel dank verschuldigd aan Alfons Hoekstra en Peter Sloot.

Dat Marieke en Dirk mijn paranimfen zijn, is geen willekeurige keus. Zij zijn mijn zonder-jullie-zou-dit-proefschrift-er-niet-zijn-geweest-vrienden. Marieke omdat zij me altijd tijdens een ochtendbakkie een hart onder de riem wist te steken en planningsen voor me opstelde als ik het zelf niet deed. En Dirk omdat ik met hem vijf jaar lang over DPD en andere zaken kon praten, en omdat in hetzelfde (Marker) schuitje zitten een stuk leuker is dan alleen. Evert heeft me erg geholpen met ingewikkelde theoretische kwesties, die in zijn uitleg opeens heel simpel bleken te zijn. Arjen, Dirk, Elske en Evert bedankt voor de nuttige discussies en het goede commentaar op de tekst. Voor de mooie omslag wil ik Ronald bedanken, maar misschien nog wel meer voor de vele vakantie-uitvluchten (“van mij mogen we vertrekken, ik ben er klaar voor”, na vijf uur wachten op het busstation). En dan is er nog die ene collega, die mij heeft geleerd dat buffers in C “buffy” horen te heten.

Maar leuker dan de promotie zelf, is de promotietijd. De dag beginnen met vers gezette koffie, het woord van de dag en de itsaio comics. Lunchen met broodjes van bakker Ed in het grote park, en vrijwel aansluitend om drie uur ijs in het kleine park. De laatste films bespreken aan het begin van het werkoverleg met Sander en Roland. Zo lang frozen bubbelen dat er zelfs in je dromen bolletjes

ploffen. De stekjes van Nicole in ruil voor het verpotten van haar planten. Gerookte zalm met dille, of palingsalade, of een tóósti tijdens de luns. Er pas heel laat achter komen dat mijn cd-laatje steeds open en dicht gaat omdat Jochem er een knop voor gemaakt heeft op itsradio (en oplossen met plakband). De strikte koffieroutine van kamer B6.23: Dirk zet 's ochtends, ik na de lunch, Evert liever niet en Menno is altijd net te laat. Vlak voor de Roetertoeter nog de planten water geven. In het zonnetje aan de kade IJwit drinken. Na vijf aioweekends kunnen zeggen dat het tweede toch echt het leukst was. Alle reclames voor de film al kennen omdat je ze de dag ervoor ook al zag. Met je rug tegen de verwarming aan de grote tafel bij de Wildeman, turen naar het bord met de bieren op tap (mmm, hoppig). Hiervoor wil ik alle collega's door de jaren heen heel erg bedanken.

Het vaste steunpunt waren Frank en Jopine; zij hoorden mijn grieven aan en respecteerden mijn keuzes. Bij Eduard zijn van al mijn promotie mumbo jumbo alleen de fuzzy bolletjes blijven hangen, maar het logeren vond ik kaasgaaf (zowel in Rotterdam als in Accra). En de belangrijkste persoon om te bedanken voor alles wat niet onder woorden te brengen is, Jasper. Want je kunt niets zeker weten, en alles gaat voorbij.

