Developing dual-scale models for structured liquids and polymeric materials

A thesis submitted to The University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Science

2016

Richard J. Gowers

School of Chemical Engineering and Analytical Science

Contents

Word count: 32,690

List of Figures

List of Tables

Abstract

The University of Manchester

Abstract of thesis submitted by Richard J Gowers for the degree of Doctor of Philosophy and entitled "Developing dual–scale models for structured liquids and polymeric materials" in the year 2016.

Computer simulation techniques for exploring the microscopic world are quickly gaining popularity as a tool to complement theoretical and experimental approaches. Molecular dynamics (MD) simulations allow the motion of an N–body soft matter system to be solved using a classical mechanics description. The scope of these simulations are however limited by the available computational power, requiring the development of multiscale methods to make better use of available resources.

Dual scale models are a novel form of molecular model which simultaneously feature particles at two levels of resolution. This allows a combination of atomistic and coarse-grained (CG) force fields to be used to describe the interactions between particles. By using this approach, targeted details in a molecule can be described at high resolution while other areas are treated with fewer degrees of freedom. This approach aims to allow for simulating the key features of a system at a reduced computational cost. In this thesis, two generations of a methodology for constructing dual scale models are presented and applied to various materials including polyamide, polyethene, polystyrene and octanol. Alongside a variety of well known atomistic force fields, these models all use iterative Boltzmann inversion (IBI) force fields to describe the CG interactions. In addition the algorithms and data structures for implementing dual scale MD are detailed, and expanded to include a multiple time step (MTS) scheme for optimising its peformance.

Overall the IBI and atomistic force fields were compatible with each other and able to correctly reproduce the expected structural results. The first generation methodology featured bonds directly between atoms and beads, however these did not produce the correct structures. The second generation used only atomistic resolution bonds and this improved the intramolecular structures greatly for a relatively minor cost. In both the polyamide and octanol systems studied, the models were also able to properly describe the hydrogen bonding. For the CG half of the force field, it was possible to either use preexisting force field parameters or develop new parameters in situ. The resulting dynamical behaviour of the models was unpredictable and remains an open question both for CG and dual scale models.

The theoretical performance of these models is faster than the atomistic counterpart because of the reduced number of pairwise interactions that must be calculated and this scaling was seen with the proposed reference implementation. The MTS scheme was successful in improving the performance with no effects on the quality of results. In summary this work has shown that dual scale models are able to correctly reproduce the structural behaviour of atomistic models at a reduced computational cost. With further steps towards making these models more accessible, they will become an exciting new option for many types of simulation.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

A huge thank you to my supervisor Dr. Paola Carbone, without whom this would not have been possible. Thanks also to my cosupervisor and advisor Dr Flor Siperstein and Carlos Avendaño.

This work was funded by BBSRC grant BB/J014478/1.

Chapter 1

Introduction

1.1 Background and motivation

Our understanding of the natural world is the best tool we have for solving the current and future problems that face its inhabitants. Through this understanding we can explain and predict natural phenomena, allowing us to engineer solutions and begin to orchestrate the world around us to our benefit. Much of our knowledge of the natural world is derived from what we know of the behaviour of some of its smallest constituent parts. We can build an understanding of large industrial processes or complex biological systems by working methodically from the atoms upwards.

With the introduction of computers into the arsenal of tools available to scientists, the amount of raw computation possible has increased by orders of magnitude. Using this computational capacity we can perform virtual experiments at the microscopic length scale that had hitherto had only been theorised. Computer simulation has grown to become a valuable tool which is complimentary to both theoretical and experimental approaches. Results and ideas can flow in both directions between these three approaches as they all attempt to describe the natural world, as shown in Figure 1.1

One of the many branches of simulation is molecular dynamics (MD), which allows the modelling of many particles under a classical mechanics description[2]. The motion of particles is iteratively calculated by calculating all pairwise forces and then applying Newton's second law, a task which was impossible before computers due to the shear number of calculations required. Unfortunately, while the power of modern computers is indeed staggering, it is still dwarfed by Avagadro's number which stands at approximately 6×10^{23} and quantifies the number of individual atoms found in just a few grams of a given material. In the face of numbers of this magnitude it is immediately obvious that brute force approaches to simulating

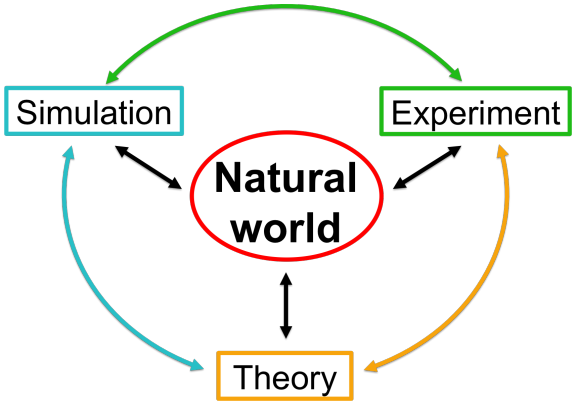


Figure 1.1: Computational modelling as a complimentary approach for both theoretical and experimental approaches to problem solving, Adapted from Ref [1].

all these atoms will progress slowly. Some thought is therefore required in how we choose to apply computational modelling to problem solving.

An obvious approach to increasing the scope of simulation is to make each particle in the model represent many atoms creating coarse-grained (CG) models, where the particles are referred to as beads. By reducing the degrees of freedom in the model, these models are able to operate much faster and have been successful in helping expand the potential of MD simulation to many real world applications[3]. Unfortunately for many applications it is necessary to retain certain specific details, both for the model to correctly reproduce behaviour and also because the desired results are these same small details. One of the best examples of such small scale details are hydrogen bonds. Despite involving the smallest atom in the periodic table, these interactions are some of the strongest between different molecules and their influence cannot be overlooked, or averaged away as in CG approaches.

Therefore more recently attention has turned to multiscale approaches which try and mix the benefits of large and small scale approaches[4, 5]. One example of a multiscale approach are dual scale or hybrid models which feature a mix of atoms and beads simulataneously within a simulation. This is relatively new method in MD, first proposed in 2006 by Christen and van Gunsteren[6]. These dual scale models make use of virtual sites (VS), which are imaginary representations which allow atoms to masquerade as CG particles[7]. By using a model which selectively retains details computational power can be applied more efficiently.

This Thesis aims to develop novel algorithms for using dual scale models for MD. Various approaches to creating and operating these models will be trialled against many test systems. One of the key benchmarks for the dual scale models that will be examined in this work is the ability to accurately recreate hydrogen bonds. Another recurring theme is the modelling of polymers. Polymers present a difficult challenge in modelling as they exhibit long characteristic length and time scales but they often feature details that require fine detail such as intramolecular

hydrogen bonding or interactions with surfaces or interfaces. Through the development of these algorithms it is hoped that new possibilities in multiscale modelling are unlocked, allowing larger systems to be simulated in a more timely manner.

1.2 Layout of the thesis

This thesis is split into two halves, starting with Part I giving an introduction to the nomenclature, algorithms and background that will be used throughout this thesis. This starts in Chapter 2 which is an overview of the basic theory of computational modelling and molecular dynamics that will underpin the rest of the work. Chapters 3 and 4 begin to delve into the more specific theory related to this thesis with reviews of multiscale modelling and in particular dual scale models.

In Part II we arrive at the novel research generated for this Thesis, presented as a series of journal publications and technical chapters. Chapter 5 features an application of a dual scale model to the polymer polyamide, which presents a difficult problem in modelling as it features a combination of polymer physics which are large scale and slow with hydrogen bonding which is very small and fast. Chapter 6 then tries to expand this same dual scale model methodology by experimenting with novel algorithms for how the system is evolved in dual scale MD simulations. The two different scales in a dual scale model can be sampled at different rates to maximise the computational efficiency and this is applied to systems of polystyrene and polyethene. The technical details of the implementation of the algorithms used in the previous two chapters are presented in Chapter 7 while Chapter 8 gives a review of parallel computation for MD and its application in this software used in this research. In Chapter 9, a revised dual scale methodology is presented, based on the author's experience to date. This is accompanied by an automated workflow to produce these models and is applied to a system of octanol, whose behaviour is dictated by their hydrogen bonds. Chapter 10 concludes this work with a discussion of the overarching themes and findings in this work, and a discussion of future directions that could be pursued.

1.3 A note on the format

This thesis is presented as an Alternative Format thesis, this was chosen as it is the author's belief that this format most accurately captures the contribution to the field made by this work. Peer reviewed scientific publications have long been the medium of exchange in scientific research, and these have become instantly available globally with the advent of the internet, thereby increasing the pace of scientific discourse.

This is reflected in the bibliography for each chapter, in total there are a handful of theses cited in contrast to hundreds of papers.

Chapters 5 and 6 are reproductions of peer reviewed articles Chapter 9 is in preparation for a peer reviewed journal, whlie Chapter 3 is an invited contribution to an online encyclopedia. All others chapters are formed of work created solely for this thesis. There is a small amount of overlap of material between Chapters 3 and 4 the former gives a wider overview of multiscale methods, while the latter is more targeted towards the work presented in this thesis.

Bibliography

[1] D. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*. New York, NY, USA: Cambridge University Press, 2005.

[2] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*. New York, NY, USA: Clarendon Press, 1989.

[3] J. J. de Pablo, "Coarse-grained simulations of macromolecules: From DNA to nanocomposites," *Annual Review of Physical Chemistry*, vol. 62, no. 1, pp. 555–574, 2011.

[4] C. Peter and K. Kremer, "Multiscale simulation of soft matter systems," *Faraday Discuss.*, vol. 144, pp. 9–24, 2010.

[5] M. Praprotnik, L. Delle Site, and K. Kremer, "Multiscale simulation of soft matter: From scale bridging to adaptive resolution," *Annual Review of Physical Chemistry*, vol. 59, no. 1, pp. 545–571, 2008.

[6] M. Christen and W. F. van Gunsteren, "Multigraining: An algorithm for simultaneous fine-grained and coarse-grained simulation of molecular systems," *The Journal of Chemical Physics*, vol. 124, no. 15, p. 154106, 2006.

[7] A. J. Rzepiela, M. Louhivuori, C. Peter, and S. J. Marrink, "Hybrid simulations: combining atomistic and coarse-grained force fields using virtual sites," *Phys. Chem. Chem. Phys.*, vol. 13, pp. 10437–10448, 2011.

Part I

Methodology

Chapter 2

Molecular Dynamics

2.1 Introduction

It is well known that the macroscopic behaviour of materials is a direct result of the microscopic motion of the particles from which they are made[1]. Therefore if we could accurately measure and perform experiments at this microscopic scale we could understand and predict a wide variety of complex behaviours. With the advent of modern computer power, we are now able to simulate the motion of particles, allowing us to perform these experiments at these length scales in a virtual environment.

Molecular dynamics (MD) is a computational method for predicting the motion of many particles under the laws of classical mechanics. The cornerstone of this is Newton's second law of motion, which relates the force on a body to its change in position over time. This can be applied repeatedly to an N body system to numerically solve the equations of motion for each particle. The first examples of using MD, in the 50s and 60s[2, 3, 4], were modest in size containing only around 1,000 particles, however this capacity has rapidly grown. Modern MD simulations are capable of modelling 10^6s of particles, representing tens of nanometers of space and milliseconds of real time, making it a potent tool for exploring the microscopic domain. This expanding potential of MD has been recognised with Karplus, Levitt and Warshel being awarded the Nobel Prize for Chemistry in 2013 for their work in developing new multiscale models for complex chemical systems[5, 6].

This chapter therefore aims to provide an overview of the fundamental algorithms used in MD as follows: Section 2.2 outlines the meaning of the term model, Section 2.3 aims to briefly describe how the force on each particle is calculated, while Section 2.4 describes how this force is applied to set the virtual experiments in motion. Finally, Section 2.5 describes the various techniques used to turn the finite simulation of particles into a full statistical ensemble, allowing us to tap into

the rich field of statistical mechanics to analyse our results.

2.2 Models

2.2.1 What is a model?

In very general terms, a scientific model is a simplified representation of a system of interest that is designed to allow us to answer questions using the information that we have placed within the model. Various approximations are made in the construction of the model, both because of unknown quantities in the system, and also to reduce the complexity of the system to something more tractable.

Good scientific models are often much simpler than their real life counterparts, but are still able to accurately describe complex phenomena, bringing with them a form of mathematical beauty. Models are able to both augment our understanding of a known process, allowing us to understand what has happened between two known states, and also allow us to predict what might happen in a given set of circumstances, i.e. given A and B, what will C be?

Despite the power of models, it is worth remembering that they are often limited in their use outside of their intended application. As a consequence of the initial choices made in constructing the model, they are often quite one dimensional in their strength. This is not an argument against using such models, but instead a reminder that care must be taken in how models are used. The best approach to comprehensively understanding a system is therefore to employ a variety of different models.

2.2.2 Molecular models

In the field of MD, molecular models are used to represent the microscopic world as a series of connected particles in a manner similar to the ball and stick representation of chemicals. These models exist in a digital form in the memory of computers, and allow us to probe microscopic lengths in ways which are often difficult or impossible to access using experimental techniques.

The molecular models used in MD belong to the classical mechanics branch of physics any quantum mechanical effects are not explicitly considered. The role of electrons is simplified to partial point charges on particles which represent an averaging of their behaviour over time. Charges in the system are then handled using methods based on Coulomb's law. This assumption imposes a lower size limit on the phenomena that can be studied with MD, and does not allow for the changing of any chemical bonds such as happens during chemical reactions. More detailed

approaches to modelling the role of electrons such as DFT[7] and ab initio MD[8] exist, however this level of detail comes at a huge computational cost.

Molecular models have been created to describe a wide variety of length and time scales. The simplest models to conceive are atomistic models, where each particle represents a single atom, and connections between these particles are simply chemical bonds. Perhaps more abstract are coarse-grained (CG) models, where particles in a model represent groups of atoms, and connections between these particles represent that atoms within these particles are themselves connected. The models in this thesis belong to the two categories described above, and the work presented aims to explore the space between these two types of model.

2.3 Forcefields

With our system of interested defined using molecular models, we can now begin to set our virtual experiment into motion. The motion of the particles in our models is caused by the net force on them, which acts to minimise their potential energy.

$$\mathbf{F}\left(\{\mathbf{r}\}\right) = -\frac{dV\left(\{\mathbf{r}\}\right)}{d\mathbf{r}} \tag{2.1}$$

Where $\mathbf{F}$ is the force, V is the potential energy and $\mathbf{r}$ is the position of the particle.

The potential energy is a consequence of the interactions between particles, and the set of functional forms and parameters which define this is collectively known as the force field. Popular force fields for atomistic modelling include AMBER[9], CHARMM[10] and GROMOS[11]. These forcefields are parametrised for modelling different types of systems, but are broadly the same, with the potential energy being split between contributions from bonded and nonbonded contributions.

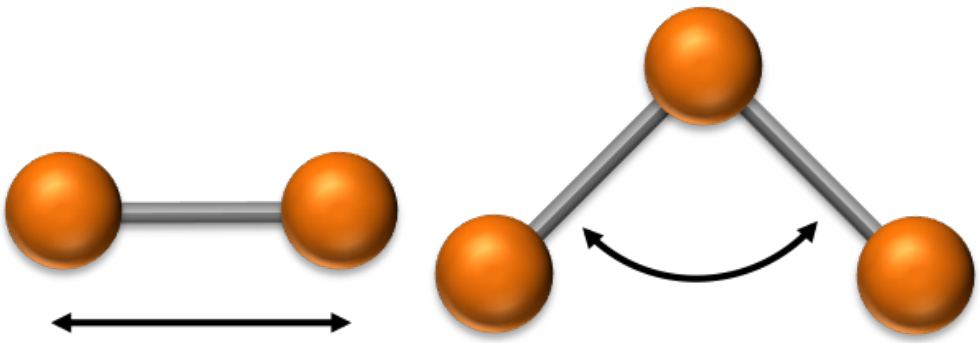
$$V(\mathbf{r}) = V_{bonded} + V_{nonbonded} \tag{2.2}$$

These will now be described in turn.

2.3.1 Bonded forces

The bonded forces on a particle are used to model the restrictions to motion caused by chemical bonds in molecules. These take the form of a sum of many potentials between 2, 3 or 4 particles.

$$V_{bonded}\left(\mathbf{r}\right) = \sum V_{bond} + \sum V_{angle} + \sum V_{dihedral} + \sum V_{improper} \tag{2.3}$$



(a) A bond between two particles (b) An angle between three particles

Figure 2.1: Visualising a bond and an angle

Bonds

For chemical bonds between two particles i and j, it is often sufficient for this to follow a simple harmonic potential which restricts the length of the bond similar to a Hookean spring.

$$V_{bond}(\mathbf{r}_{ij}) = k_{bond}\left(|\mathbf{r}_{ij}| - r_0\right)^2 \tag{2.4}$$

Where r_0 refers to the rest length of the bond, and k_{bond} gives the stiffness of the bond.

This potential is one of the strongest potentials in a forcefield, which is understandable as it is what holds a molecule together, however this also has the consequence that it has the highest frequency causing it to be the limiting factor in deciding the time step, (Δt in Section 2.4).

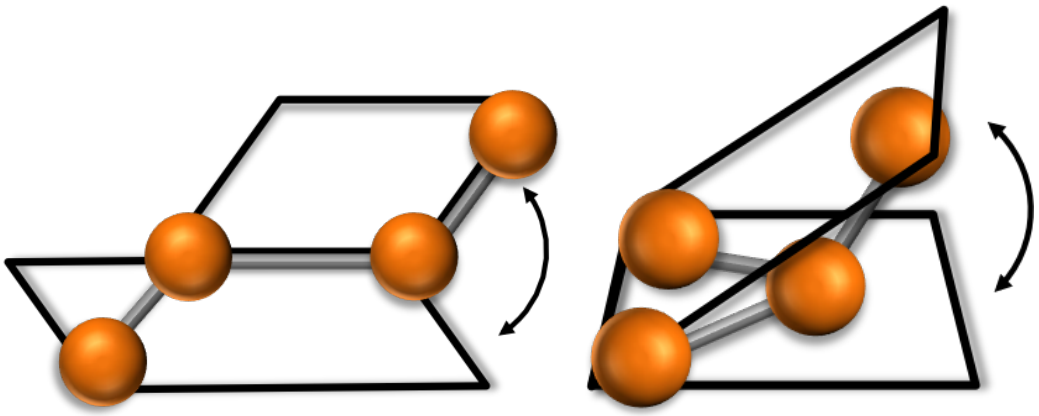
For this reason, it is often chosen to fix or constrain bonds at a prescribed length so that the time step can be increased, usually by a factor of 2–3. There are numerous constraint algorithms including SHAKE[12], RATTLE[13], SETTLE[14] and LINCS[15].

Angles

Angular potentials are employed to maintain the angle subtended between three particles, shown in Figure 2.1b. These can often also be modelled using a harmonic potential

$$V_{angle}(\theta_{ijk}) = k_\theta(\theta_{ijk} - \theta_0)^2 \tag{2.5}$$

Where k_θ is a measure of stiffness and θ_0 a reference angle.



(a) Dihedral (b) Improper dihedral

Figure 2.2: The dihedral angle between 4 particles

Dihedrals and improper dihedrals

Regular dihedral, or torsional, potentials are used to model the energy penalty for bonds rotating. The potential is applied to 4 sequential atoms, with an angle being measured between the intersection of the planes formed by the first three and last three atoms, shown in Figure 2.2a. The functional form for this typically take the shape of a cosine function, with multiple energy minima, as shown in Equation 2.6.

$$V_{dihedral} = k_\phi \left(1 + \cos\left(n\phi - \phi_0\right)\right) \tag{2.6}$$

Where k_ϕ defines the resistance to rotation, n is the multiplicity of the dihedral, ϕ is the angle between the two planes and ϕ_0 is the reference angle.

Finally, improper or out–of–planes torsions are used to control the planarity of four particles, the angle is again measured as the intersection of two planes, but the atoms are no longer sequentially bonded, shown in Figure 2.2b. This is used to keep a section planar, such as in ring molecules, or to keep a pyramidal shape, such as in ammonia. These again can use a stiff harmonic potential, as any bending is unnatural.

2.3.2 Nonbonded forces

Nonbonded forces model the attractions and repulsions collisions between particles as they move throughout the system volume. With the exception of particles with which they have bonded interactions, particles interact with all other particles in their vicinity through a pairwise potential, as shown in Equation 2.7. This has two contributions, which will now be explained in turn.

$$V_{nonbonded} = \sum_j V_{dispersive}(r_{ij}) + \sum_j V_{qq}(r_{ij}) \tag{2.7}$$

Dispersive forces

The dispersive force models the weak attraction between particles caused by van der Waals forces and the strong repulsion at short distances cause by their physical size. The dispersive potentials between particles can be categorised as either hard or soft. Hard potentials increase exponentially as the distance between particles closes, ultimately causing the particles to repel each other with enough force to never overlap.

The most common hard potential between atoms is the Lennard–Jones (LJ) potential[16], shown in Equation 2.8. This is parameterised by ε_{ij} which defines the total attraction between the two particles and σ_{ij} which defines their effective radius.

$$V_{lj} = 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{2.8}$$

There are many potentials which are variants of the LJ potential, such as the Mie potential[17] which uses different exponents than the standard 12–6 and the Weeks-Chandler-Anderson (WCA) potential[18] which uses only the repulsive half of the LJ potential

Softer potentials still increase as particles come together, but reach a finite maximum value. This has the consequence that particles can pass through each other if they collide with enough momentum. While this seems strange at first, it is perfectly reasonable for CG models where the particle represents a group of atoms. As these particles begin to overlap the individual atoms navigate around each other, allowing the centers of the two particles to occupy the same location. Some CG force fields use analytical forms, however others use a numerical form for the potential is used[19]. For even larger particles, sometimes just a simple linear ramp can be used between particles, such as is used in dissipative particle dynamics[20, 21].

To avoid problems with the finite size of the system, and to reduce the number of interactions that must be calculated, a maximum range is put on nonbonded interactions, generally called the cutoff radius or r_{cut}. For dispersive forces the value at longer ranges is generally very weak and so can be truncated beyond r_{cut}.

$$V_{ij} = \begin{cases} V_{dispersive}(r_{ij}) & r_{ij} \leq r_{cut} \\ 0 & r_{ij} > r_{cut} \end{cases} \tag{2.9}$$

Inevitably however, this will lead to a discontinuity in the potential at the cut-

off. Equation 2.8 can be modified so that both the potential and its derivative go continuously to zero at the cutoff[22] as shown in Equation 2.10.

$$V_{lj,shifted}(r_{ij}) = V_{lj}(r_{ij}) + 4\varepsilon_{ij}\left\{\left[6\left(\frac{\sigma_{ij}}{r_{cut}}\right)^{12} - 3\left(\frac{\sigma_{ij}}{r_{cut}}\right)^{6}\right]\left(\frac{r_{ij}}{r_{cut}}\right)^{2} - \left[7\left(\frac{\sigma_{ij}}{r_{cut}}\right)^{12} - 4\left(\frac{\sigma_{ij}}{r_{cut}}\right)^{6}\right]\right\} \tag{2.10}$$

It should also be noted that these long range contributions can have a large effect on the calculated pressure of the system, which will be defined in Equation 2.30.

Electrostatics

The electrostatic force between two particles models the force experienced because of net charge on the particles. The potential between two point charges in a vacuum is defined according to Coulomb's law as

$$V_{qq}(r_{ij}) = \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}} \tag{2.11}$$

Where q is the charge on the particle, and ε_0 is the permittivity of free space.

As this potential decays at a rate of r^{-1}, simply truncating the potential at the same r_{cut} as the dispersive potentials would cause a noticeable jump in the potential. However increasing r_{cut} far enough so the potential has decayed to a small enough amount would require a very large distance, and hence the calculation of an extreme amount of pairs. Methods therefore exist to approximate the electrostatic potential at longer distances.

The first option available to deal with long range electrostatics is the reaction field (RF) method[23, 24]. This treats all interactions within r_{cut} explicitly, and outside of this range treats the system as a dielectric continuum. To do this, a modified version of Equation 2.11 is used

$$V_{rf}(r_{ij}) = \frac{q_i q_j}{4\pi\varepsilon_0}\left(\frac{1}{r_{ij}} + \frac{\varepsilon_{rf} - 1}{2\varepsilon_{rf} + 1}\frac{r_{ij}^2}{r_{cut}^3}\right) \tag{2.12}$$

Where ε_{rf} is an adjustable parameter which describes the magnitude of the screening effect. When $\varepsilon_{rf} = 1$ the original Coloumb potential is retrieved.

The particle mesh ewald (PME) method[25, 26] is an alternative, more complex, method for calculating the electrostatic interactions. In this, the short range interactions are calculated directly and the long range contributions are calculated using an Ewald summation[27], which uses a fast Fourier transform to approximate the force.

Of these two methods, the RF method is simpler and computationally cheaper as the PME method requires a Fourier transform of coordinates. The RF method is not however suitable for all applications because of the assumption of a constant dielectric continuum. This is not valid in inhomogeneous systems, such as interfaces, where there can be differences in the electrostatic environment.

2.4 Moving particles

Now that the force on each particle has been defined, we can turn our attention to how this affects their position. The force on a body, $\mathbf{F}$, is related to its change in position, $\mathbf{r}$, by Newton's second law, shown in Equation 2.13

$$\mathbf{F} = m\frac{d^2\mathbf{r}}{dt^2} \tag{2.13}$$

Where m represents the mass of the body, and t the time.

While Equation 2.13 defines a continuous relationship between force and position, in our computer simulation everything must be done in discrete steps. This gives rise to integration algorithms which try to minimise the error and maximise their stability while still being computationally efficient. A quick derivation of the most commonly used method follows, while Chapter 6 deals with exploring new options in multiscale systems.

Considering a Taylor expansion of the position of a particle, $\mathbf{r}$, forwards a discrete step Δt forwards in time, we arrive at the Euler fowards approximation algorithm,

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t \tag{2.14a}$$

$$\mathbf{v}\,(t + \Delta t) = \mathbf{v}(t) + \frac{\mathbf{F}(t)}{m}\Delta t \tag{2.14b}$$

Where $\mathbf{v}$ is the velocity ($\frac{d\mathbf{r}}{dt}$), the error is of the order Δt^3, and Equation 2.13 has been substituted in.

This algorithm however is not time reversible, meaning that stepping backwards through time will not allow us to return to a given start point. This is a serious problem for MD, most notably because it means that momentum will not be conserved, causing simulations to gradually spiral out of control.

2.4.1 The Verlet algorithm

If we consider the same Taylor expansion, but instead take a step Δt backwards in time

$$\mathbf{r}(t-\Delta t) = \mathbf{r}(t) - \mathbf{v}(t)\Delta t + \mathbf{F}(t)\frac{\Delta t^2}{2m} + \mathcal{O}(\Delta t^3) \tag{2.15}$$

And then add Equations 2.14 and 2.15, after rearranging we arrive at the Verlet algorithm[28].

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + \mathbf{F}(t)\frac{\Delta t^2}{m} + \mathcal{O}(\Delta t^4) \tag{2.16}$$

Where the Δt and Δt^3 terms have cancelled out, leaving the error of order Δt^4. Because of the symmetrical way the steps are defined, this algorithm is now time reversible, and so can be widely used in MD. This formulation however does not explicitly feature the velocities, which are important for defining many properties of the system, such as the temperature.

2.4.2 Leapfrog algorithm

An alternate formulation for integrating the equations of motion is the Leapfrog algorithm[29]. This starts with Equation 2.14, but uses the velocity at a point in time half way towards

the point in time half way between when the positions are defined to define the new position

$$\mathbf{r}\left(t+\Delta t\right) = \mathbf{r}(t) + \mathbf{v}\left(t + {}^1\!/_2\Delta t\right)\Delta t + \mathcal{O}\left(\Delta t^4\right) \tag{2.17}$$

The velocities at this future half step are defined based on the previous half step velocities and the force at the current point in time.

$$\mathbf{v}\left(t + {}^1\!/_2\Delta t\right) = \mathbf{v}\left(t - {}^1\!/_2\Delta t\right) + \frac{\mathbf{F}\left(t\right)}{m}\Delta t + \mathcal{O}\left(\Delta t^3\right) \tag{2.18}$$

This is called the Leapfrog algorithm because of how the positions and velocities continually pass each other with each step. This formulation of the equations of motion is often more preferable to the Verlet algorithm as the velocities are calculated each step. If the velocities at precisely time t are required, these can be found from the two half step velocities

$$\mathbf{v}\left(t\right) = \frac{1}{2}\left(\mathbf{v}\left(t + {}^1\!/_2\Delta t\right) + \mathbf{v}\left(t - {}^1\!/_2\Delta t\right)\right) + \mathcal{O}\left(\Delta t^2\right) \tag{2.19}$$

2.5 Ensembles

Now that the we have defined how the forces are calculated and our solution to the equations of motion, we can begin generating results. As defined in the previous two

Sections, the force on all particles can be calculated and their positions accordingly updated. Performing this iteratively allows us to create a description of the system over time.

The positions and momenta can be periodically recorded to create a trajectory of the system. This series of snapshots of the system has a much more profound significance however, as the positions and momenta of the N particles in the system can be represented as a single point in $6N$ dimensional phase space. This phase space has, for all N particles, 3 dimensions for both positions ($\mathbf{r}^N$) and momenta ($\mathbf{p}^N$), and represents all possible states that the system could occupy. The process of MD has then allowed us to sample many point within this phase space with the collection of phase space points representing a statistical ensemble as originally defined by Gibbs[30].

The process as defined so far will conserve energy and samples the microcanonical or NVE ensemble. Different thermodynamic properties can be measured and optionally controlled to sample other thermodynamic ensembles, and a few of these will be described.

2.5.1 Temperature

The instantaneous temperature of a system with N particles can be defined according to classical mechanics as

$$T = \frac{2\mathcal{K}}{3k_B N} \tag{2.20}$$

Where k_B is the Boltzmann constant, the denominator 3 assumes a three dimensional system, and $\mathcal{K}$, the total kinetic energy of the system, is given by the sum of the kinetic energy of each particle

$$\mathcal{K} = \frac{1}{2}\sum_{i}^{N} m_i \mathbf{v}_i^2 \tag{2.21}$$

Yielding an expression for the instantaneous temperature as a function of the velocity of the particles in the system

$$T(t) = \frac{\sum_i^N m_i \mathbf{v}_i(t)^2}{3k_B N} \tag{2.22}$$

Thermostats

Whilst it is possible to perform simulations in the microcanonical ensemble, it is more common to control the average temperature of the system and sample the NVT or canonical ensemble. There are two main reasons for wanting to sample the canonical

ensemble, firstly real life situations are better represented by the canonical ensemble where fluctuations in temperature are moderated by the surroundings. Secondly, due to floating point rounding errors in computer, there will inevitably be some drift in the total energy of the system, and controlling the temperature offers a way to counteract this.

Temperatures are controlled using a thermostat which aims to maintain the system at a reference temperature T_0. Considering Equation 2.22 it is apparent that we could modify the temperature by simply rescaling the velocities of all particles in the system by a factor λ

$$T' = \frac{\sum_i^N m_i \left(\lambda \mathbf{v}_i(t)\right)^2}{3k_B N} \tag{2.23}$$

Where the target temperature can be attained immediately by using:

$$\lambda^2 = \frac{T_0}{T(t)} \tag{2.24}$$

A less heavy handed approach is the Berendsen thermostat[31], which controls the temperature with according to a coupling constant τ_T, which damps the response

$$\frac{dT(t)}{dt} = \frac{1}{\tau_T}\left(T_0 - T(t)\right) \tag{2.25}$$

The velocity rescaling factor λ is then given as

$$T(t+\Delta t) - T(t) = \frac{\Delta t}{\tau_T}\left(T_0 - T(t)\right) \tag{2.26a}$$

$$\lambda^2 = \frac{T(t+\Delta t)}{T(t)} = 1 + \frac{\Delta t}{\tau_T}\left(\frac{T_0}{T(t)} - 1\right) \tag{2.26b}$$

Where it can be seen that if the coupling constant is equal to the time step Equation 2.24 is recovered.

It is worth mentioning that the ensemble generated through using the Berendsen thermostat is not strictly a canonical ensemble, but instead an iso–kinetic ensemble. This is because whilst this thermostat will correctly control the mean temperature, energy fluctuations in the system are not properly captured.

To properly sample the canonical ensemble more complicated thermostats can be used. The velocity rescale thermostat[32] is similar to the Berendsen thermostat but includes a stochastic term which correctly replicates the fluctuations in temperature. The Nosé–Hoover thermostat[33, 34] instead controls the temperature by adding parameters to explicitly model the thermal bath. The equations of motion are extended to include a friction term ξ

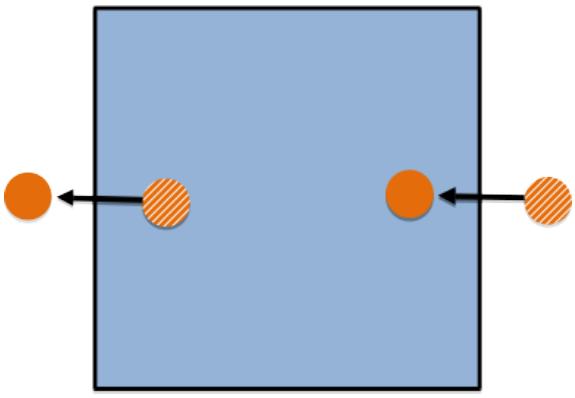


Figure 2.3: Periodic boundary conditions. The particle exiting the left of the simulation volume, reappears on the right with its momentum unchanged

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{\mathbf{F}_i(t)}{m_i} - \xi\mathbf{v}(t) \tag{2.27a}$$

$$\frac{d\xi}{dt} = \frac{1}{Q}\left(T(t) - T_0\right) \tag{2.27b}$$

Where the parameter Q works as a mass parameter to damp the response to differences in temperature. In this approach the total energy of the system and thermal reservoir are kept constant.

2.5.2 Volume and periodic boundaries

Because the number of particles simulated is necessarily finite, a solution is required to deal with the edges of the volume and allow the system to be treated as a continuum. In MD the most commonly used technique is periodic boundary conditions, where the simulated volume is considered to be a repeating entity, so that each face of the simulation volume is in contact with the opposite face.

The most common shape for the simulation box is a simple cuboid, whose edges can be described as three orthogonal vectors, denoted as $\mathbf{L} = \{L_x, L_y, L_z\}$. Any tessalating three dimensional shape could be used, included truncated octahedrons, rhombic dodecahedrons and parallelepipeds. These are primarily useful for minimising the "spare" volume around a large solvated molecule which would otherwise have to be filled with relatively uninteresting solvent particles [35].

When a particle passes through one of the faces of the primary unit cell, it appears on the opposite side with its momentum unchanged, as shown in Figure 2.3. This effect is similar to the movement in the game Pacman. This repeating geometry means that every particle's position within the system can be considered to be within the primary unit cell:

$$0 < \mathbf{r} \leq \mathbf{L} \tag{2.28}$$

Minimum image convention

When considering the vector between two particles $\mathbf{r}_{ij}$, the repeating geometry gives us an option of what this distance could be. If we picture the simulation volume being surrounded by 26 images of itself forming a 3×3 grid, it is clear that we could draw a vector between i in the primary unit cell and any of the 27 other versions of j.

The minimum image convention, Equation 2.29, defines what is the shortest version of this vector. As soon as any of the three components of the vector $\mathbf{r}_{ij}$ are larger than half the corresponding box vector, multiples of $\mathbf{L}$ may be subtracted.

$$\mathbf{r}_{ij,min} = \mathbf{r}_{ij} - \left\lfloor \frac{\mathbf{r}_{ij}}{\mathbf{L/2}} \right\rfloor \times \mathbf{L} \tag{2.29}$$

Where the brackets indicate a floor division.

For example, if $\mathbf{r}_{ij} = \{0.8L_x, 0.4L_y, 0.1L_z\}$, then there is a shorter version of the vector between these two particles described as $\mathbf{r}_{ij,min} = \{-0.2L_x, 0.4L_y, 0.1L_z\}$. This convention is used the majority of the time when considering distances, however for some measures, in particular those that follow the bonded structure of a molecule such as considering the end to end length of a polymer chain, the minimum image convention should be disregarded.

This convention also imposes a minimum on the size of the system as each of the box vectors must be at least double the cutoff of the nonbonded interaction range (r_{cut}) to avoid a particle seeing multiple images of another given particle. This limit is equally true in the analysis of simulation results, where the largest length which can be measured should be treated with care.

2.5.3 Pressure

The instantaneous pressure tensor ($\mathbf{P}$) within the simulation volume (V) can be determined according to the Clausius virial:

$$\mathbf{P} = \frac{k_B T N}{V} + \frac{1}{3V} \sum_{i<j} \mathbf{F}_{ij} \mathbf{r}_{ij} \tag{2.30}$$

Where the summation represents the inner virial and is performed over all pair-wise forces between particles, $\mathbf{F}_{ij}$ multiplied by the vector between them and the factor 3 in the denominator assumes a three dimensional system. The isotropic pressure, P, is then the average of the trace of the pressure tensor.

$$P(t) = \mathrm{Tr}(\mathbf{P}) \tag{2.31}$$

Barostats

Rather than conduct the simulation in a fixed volume, it may be preferable to simulate in a fixed pressure, allowing the NPT or isothermal isobaric ensemble to be sampled. This can be done by allowing the system volume to change slightly in response to fluctuations in pressure. If the pressure in the system is too high, the walls of the simulation box can move outwards in response and vice versa. This is controlled by the barostat, which tries to allow this volume change while trying not to perturb the system.

The simplest barostat is the Berendsen barostat[31], which in a similar manner to the previously described thermostat weakly couples the system pressure to an external bath.

$$\frac{dP(t)}{dt} = \frac{P_0 - P}{\tau_P} \tag{2.32}$$

Where τ_P is a time constant to damp the response and P_0 is the desired pressure. The positions of all particles $\mathbf{r}$ and the box vectors $\mathbf{L}$ are scaled by a factor μ defined as

$$\mu = \left(1 - \frac{\kappa_t \Delta t}{\tau_P}(P_0 - P)\right)^{\frac{1}{3}} \tag{2.33}$$

Where κ_t is the isothermal compressibility of the system.

Despite being relatively simple, this barostat is also very robust and performs well even when the system is far from equilibrium. More complex barostats exist, such as the Parrinello–Rahman barostat[36], which use the full pressure tensor to deform the box vectors individually.

2.5.4 Sampling from an ensemble

Now that we have defined the thermodynamic ensemble for our system, we can sample macroscopic quantities from it. For example, if there is a property A, which is measurable at a given time based upon the current position in phase space, that is $A = A(\mathbf{p}^N(t), \mathbf{r}^N(t))$, then to calculate the average of A requires observing over a large period of time

$$A_{average} = \frac{1}{\tau} \lim_{\tau \to \infty} \int_{t=0}^{\tau} A\left(\mathbf{p}^N(t), \mathbf{r}^N(t)\right) dt \tag{2.34}$$

An important hypothesis, originally by Boltzmann[37] is the ergodic hypothesis[1], which state that the ensemble average of a property $\langle A \rangle$ is equal to this time average. This means that we can calculate properties based on an average of frames of our MD trajectory, provided that it represents a good sampling of the thermodynamic

ensemble.

$$\langle A \rangle = \frac{1}{M} \sum^{M} A\left(\mathbf{p}^N, \mathbf{r}^N\right) \tag{2.35}$$

Where M represents the total number of frames that were analysed. This hypothesis allows us to confidently predict macroscopic properties as long as the system was able to freely explore a good sample of the phase space.

Bibliography

[1] D. A. McQuarrie, *Statistical mechanics*. Sausalito (Calif.): University Science Books, 2000.

[2] J. B. Gibson, A. N. Goland, M. Milgram, and G. H. Vineyard, "Dynamics of radiation damage," *Phys. Rev.*, vol. 120, pp. 1229–1253, 1960.

[3] B. J. Alder and T. E. Wainwright, "Phase transition for a hard sphere system," *The Journal of Chemical Physics*, vol. 27, no. 5, pp. 1208–1209, 1957.

[4] A. Rahman, "Correlations in the motion of atoms in liquid argon," *Phys. Rev.*, vol. 136, pp. A405–A411, 1964.

[5] A. Warshel and M. Karplus, "Calculation of ground and excited state potential surfaces of conjugated molecules. I. formulation and parametrization," *Journal of the American Chemical Society*, vol. 94, no. 16, pp. 5612–5625, 1972.

[6] M. Levitt and A. Warshel, "Computer simulation of protein folding," *Nature*, vol. 253, pp. 694–698, 1975.

[7] R. G. Parr and Y. Weitao, *Density-Functional Theory of Atoms and Molecules*. Oxford University Press, 1994.

[8] D. Marx and J. Hutter, *Ab initio molecular dynamics : Basic theory and advanced methods*. Cambridge, New York: Cambridge University Press, 2009.

[9] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman, "A second generation force field for the simulation of proteins, nucleic acids, and organic molecules," *Journal of the American Chemical Society*, vol. 117, no. 19, pp. 5179–5197, 1995.

[10] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus *Journal of Computational Chemistry*, vol. 4, no. 2, pp. 187–217, 1983.

[11] W. R. P. Scott, P. H. Hünenberger, I. G. Tironi, A. E. Mark, S. R. Billeter, J. Fennen, A. E. Torda, T. Huber, P. Krüger, and W. F. van Gunsteren, "The GROMOS biomolecular simulation program package," *The Journal of Physical Chemistry A*, vol. 103, no. 19, pp. 3596–3607, 1999.

[12] J. Ryckaert, G. Ciccotti, and H. J. Berendsen, "Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes," *Journal of Computational Physics*, vol. 23, no. 3, pp. 327–341, 1977.

[13] H. C. Andersen, "Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations," *Journal of Computational Physics*, vol. 52, no. 1, pp. 24–34, 1983.

[14] S. Miyamoto and P. A. Kollman, "Settle: An analytical version of the SHAKE and RATTLE algorithm for rigid water models," *Journal of Computational Chemistry*, vol. 13, no. 8, pp. 952–962, 1992.

[15] B. Hess, H. Bekker, H. J. C. Berendsen, and J. G. E. M. Fraaije, "LINCS: A linear constraint solver for molecular simulations," *Journal of Computational Chemistry*, vol. 18, no. 12, pp. 1463–1472, 1997.

[16] J. E. Jones, "On the determination of molecular fields. II. from the equation of state of a gas," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 106, no. 738, pp. 463–477, 1924.

[17] G. Mie, "Zur kinetischen theorie der einatomigen körper," *Annalen der Physik*, vol. 316, no. 8, pp. 657–697, 1903.

[18] J. D. Weeks, D. Chandler, and H. C. Andersen, "Role of repulsive forces in determining the equilibrium structure of simple liquids," *The Journal of Chemical Physics*, vol. 54, no. 12, pp. 5237–5247, 1971.

[19] D. Reith, M. Pütz, and F. Müller-Plathe, "Deriving effective mesoscale potentials from atomistic simulations," *Journal of Computational Chemistry*, vol. 24, no. 13, pp. 1624–1636, 2003.

[20] R. D. Groot and P. B. Warren, "Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation," *The Journal of Chemical Physics*, vol. 107, no. 11, pp. 4423–4435, 1997.

[21] P. Español, "Dissipative particle dynamics.," in *Handbook of Materials Modeling* (S. Yip, ed.), ch. 8, pp. 2503–2512, Dordrecht: Springer Netherlands, 2005.

[22] S. D. Stoddard and J. Ford, "Numerical experiments on the stochastic behavior of a Lennard-Jones gas system," *Phys. Rev. A*, vol. 8, pp. 1504–1512, Sep 1973.

[23] J. A. Barker and R. O. Watts, "Monte carlo studies of the dielectric properties of water-like models," *Molecular Physics*, vol. 26, no. 3, pp. 789–792, 1973.

[24] I. G. Tironi, R. Sperb, P. E. Smith, and W. F. van Gunsteren, "A generalized reaction field method for molecular dynamics simulations," *The Journal of Chemical Physics*, vol. 102, no. 13, pp. 5451–5459, 1995.

[25] T. Darden, D. York, and L. Pedersen, "Particle mesh Ewald: An N log(N) method for Ewald sums in large systems," *The Journal of Chemical Physics*, vol. 98, no. 12, pp. 10089–10092, 1993.

[26] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, "A smooth particle mesh ewald method," *The Journal of Chemical Physics*, vol. 103, no. 19, pp. 8577–8593, 1995.

[27] P. P. Ewald, "Die berechnung optischer und elektrostatischer gitterpotentiale," *Annalen der Physik*, vol. 369, no. 3, pp. 253–287, 1921.

[28] L. Verlet, "Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules," *Phys. Rev.*, vol. 159, pp. 98–103, 1967.

[29] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*. Bristol, PA, USA: Taylor & Francis, Inc., 1988.

[30] J. W. Gibbs, *Elementary principles in statistical mechanics developed with especial reference to the rational foundation of thermodynamics*. New York: C. Scribner, 1902.

[31] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," *The Journal of Chemical Physics*, vol. 81, no. 8, pp. 3684–3690, 1984.

[32] G. Bussi, D. Donadio, and M. Parrinello, "Canonical sampling through velocity rescaling," *The Journal of Chemical Physics*, vol. 126, no. 1, p. 014101, 2007.

[33] S. Nosé, "A unified formulation of the constant temperature molecular dynamics methods," *The Journal of Chemical Physics*, vol. 81, no. 1, pp. 511–519, 1984.

[34] W. G. Hoover, "Canonical dynamics: Equilibrium phase-space distributions," *Phys. Rev. A*, vol. 31, pp. 1695–1697, 1985.

[35] T. A. Wassenaar and A. E. Mark, "The effect of box shape on the dynamic properties of proteins simulated under periodic boundary conditions," *Journal of Computational Chemistry*, vol. 27, no. 3, pp. 316–325, 2006.

[36] M. Parrinello and A. Rahman, "Crystal structure and pair potentials: A molecular-dynamics study," *Phys. Rev. Lett.*, vol. 45, pp. 1196–1199, 1980.

[37] L. Boltzmann, *Vorlesungen über Gastheorie*. Leipzig: J. A. Barth, 1896.

Chapter 3

Coarse-Grained and Hybrid Simulations of Nanostructures

Preface

The following Chapter was originally published in the Encyclopedia of Nanotechnology, Springer Science[1]. A reprint of the first page is given in Appendix A.

Its inclusion here serves as an introduction to the different approaches to creating coarse–grained force fields. The theory associated with each approach along with an example of its application is given. The dual scale models which are the subject of this Thesis are briefly mentioned, with a more thorough review forthcoming in the next Chapter.

Coarse-Grained and Hybrid Simulations of Nanostructures

Richard J. Gowers and Paola Carbone
School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester

Definition

In computational chemistry coarse-grained (CG) models are defined as molecular models where some details (i.e., degrees of freedom) of the original chemical structure have been removed. The resulting models are a coarser description of the chemical systems compared with the original ones and can then be used to perform either molecular dynamics or Monte Carlo simulations[2]. The reduction of the models degrees of freedom enables the simulation of systems whose size is comparable with that of the experimental ones and the timescale spanned by these simulations can reach microseconds.

3.1 Overview

Computer modeling is a powerful technique to gain molecular level details of chemical systems under different physical conditions and enables to relate macroscopic observations with changes in the chemical and physical state of the system. However, all modeling techniques rely on computer hardware, and therefore their use is limited by the available computer power. State-of-art simulations can nowadays reach the size of few millions of atoms, but if standard high-performance computers are used, the system size usually does not exceed few hundred thousand particles. Indeed, during a molecular simulation, the number of interatomic interactions that must be computed every iteration is proportional to N^2, where N is the total number of system particles[2]. This heavy use of the CPUs limits not only the size of molecular models but also the timescale the system can be simulated for.

One way to circumvent this problem is to reduce the number of interacting particles (N) in the systems, simplifying the models and modifying the original interacting parameters to include in an implicit way the neglected details. The

simplest example of such coarse graining is the development of united–atom (UA) force fields[3] where the hydrogen atoms and the aliphatic carbon to which they are covalently bonded are modeled as a single entity. The assumption underlying the development of such force fields is that the physics of the model is not affected by neglecting the explicit interactions involving the aliphatic hydrogen atoms. A similar decision on how many and which atomistic details can be neglected in a molecular model (procedure known as mapping scheme) is the key decision that must be carefully made every time a new coarse-grained model is developed if some of the system chemical and physical features have to be maintained. It has been indeed shown that mapping schemes which retain different features of the original molecule perform differently depending on the property analyzed[4].

The model simplification done in the UA force fields can therefore be carried on at much larger scale: large group of atoms can be lumped up in single super-atoms (or beads) and even entire colloidal particles can be modeled as single rigid bodies.

A very broad distinction can be made between those CG models aiming at preserving some chemical details of the original system and those which instead preserve only the shape of large molecular aggregates. The former can reproduce with a certain degree of accuracy the system enthalpy, while the latter reproduce the only entropic effects, and it is typically employed in modeling colloidal systems [4, 5, 6, 7].

3.2 Procedures

Since different features of the atomistic model can be used as target properties, several procedures to develop the effective interactions between the beads have been proposed. Some of the most popular methods used in materials science are briefly introduced below.

3.2.1 Structural Based Model

The target property to reproduce is the structure of the atomistic model. The CG non-bonded part of the force field is therefore refined until it reproduces structural correlation functions such as the radial distribution function (RDF) of the atomistic system. The quality of the agreement is quantified using a merit function of the form

$$\chi^2 = \sum \left(y_{target} - y_{CG}\right)^2 \tag{3.1}$$

where y_{target} and y_{CG} are the correlation functions calculated from the atomistic and CG model, respectively. The sum is over all coordinates and simulations. The minimization of χ is a nonlinear inverse problem for which no analytical expression is

available and where linearization is unsuccessful. An iterative procedure is therefore run and the CG force field parameters are consequently adjusted. Two iterative procedures are usually employed: one uses Monte Carlo simulations[8] and the other one molecular dynamics[9]. In both cases, the new CG model is built upon the atomistic one and the existence of a one-to-one correspondence between the two models enables an easy interchange between the two model resolutions[4].

3.2.2 Thermodynamic Models

The aim of these types of models is to reproduce some thermodynamic properties of the atomistic system. The CG potential is usually modeled using a Mie (generalized Lennard-Jones) function

$$V(r) = C\epsilon \left[\left(\frac{\sigma}{r}\right)^n - \left(\frac{\sigma}{r}\right)^m \right] \tag{3.2}$$

where n and m are the exponents controlling the softness of the potential, σ is the particle diameter, ϵ is the potential well depth, and C is chosen in such a way that the minimum of the potential corresponds to ϵ.

The key parameters to control and iteratively optimize are ϵ and σ. In some procedure the exponents n and m can also be varied. The target properties for the CG model include partition coefficients[10], density, and interfacial tension[11]. Another more systematic approach to obtain the sought force field parameters involves the solution of a molecular-based equation of state (EoS) which takes advantage of the fact that for some molecular fluids the SAFT EoS can be solved[12].

3.2.3 Force Matching

In this type of method[13, 14], the CG interactions are again parameterized using the underlying atomistic interactions, but in this case the difference to be minimized is that between the atomistic and CG forces and the minimization is solved using a variational approach

$$\chi^2 = \frac{1}{3N} \langle \sum_{I=1}^{N} (F_{target} - F_{CG})^2 \rangle \tag{3.3}$$

where the angular brackets denote average over the trajectory, F_{target} is the net force on the atomistic site I, F_{CG} is the force on the same CG site, and N is the number of sites (beads) in the CG model. Within this type of CG model a one-to-one correspondence between the atomistic and CG resolution is again established, and reintroduction of atoms in the mesoscopic model is possible.

3.2.4 Excess Entropy Model

In this case the function which needs to be minimized is the relative entropy (S_{rel}) which is defined as the difference between the distributions of configurations generated by the atomistic (P_{at}) and CG (P_{CG}) models. Using the KullbackLeibler divergence formalist, one can define S_{rel} as

$$S_{rel}(U) = \int P_{at} \frac{P_{at}}{P_{CG}(U)} dR \tag{3.4}$$

where U is the CG potential and the average is evaluated over the CG configurational space, but weighted according to the atomistic probability distribution, P_{at}. S_{rel} provides a variational framework for determining the approximate CG potential (U) that reproduces target atomistic distributions[15, 16].

3.2.5 Dissipative Particle Dynamics

The techniques presented above despite simplifying the molecular model still retain a quite high degree of chemical specificity. If however much larger systems must be simulated, a mesoscopic approach such as dissipative particle dynamics (DPD) can be employed. Such method can sample large conformational space in relatively short time, and therefore systems of the order of hundreds of nanometers can be simulated[17, 18]. In this method, each bead represents a much larger amount of material than the previously discussed methods, such as an entire molecule or a few molecules of solvent. Because each DPD particle represents such large portion of the chemical system, the standard analytical functions used to model the particle-particle non-bonded interactions (Eq. 3.2) cannot be used. Therefore, in DPD the forces are expressed as the sum of three contributions

$$F_{ij} = F_{ij}^{C}(r_{ij}) + F_{ij}^{D}(\nu_{ij}) + F_{ij}^{R}(\theta) \tag{3.5}$$

where F_{ij}^{C} represents a conservative repulsive force between particles and it is notably weaker than the standard short distance forces (Eq. 3.2) allowing particles to pass through each other (the so-called soft potential). Such weak short-range interaction is a necessary consequence of the much larger mapping scheme used in this modeling technique. F_{ij}^{D} is a dispersive force between particles which is a function of their relative velocity and represents the effect of fluid viscosity. Finally, F_{ij}^{R} is a random force, a function of a Gaussian random number θ, which replicates the thermal and vibrational energy of the system.

Because all of these forces, including the random force, are applied between pairs of particles, momentum is conserved throughout the system. DPD simulations are tuned to replicate hydrodynamic properties by adjusting the strength of each of

these three forces. DPD has been used to model problems which are otherwise out of reach using particle-based methods, including modeling self-assembly and phase diagram of complex fluids.

3.2.6 Mean Field Theory

A radically different approach to coarse graining is that which simplifies the molecular systems at such level that its discrete nature (the fact that is made by individual atoms) is replaced by a continuum mean field. In this context the molecular system is modeled using a grid of points on which an effective field, representing the averaged interaction caused by the presence of all the system particles within a cutoff distance, which is identified with the mesh of the grid, acts. In its basic form, to obtain the field (H_{eff}), the free energy of the system is minimized with respect to the distribution of all possible configurations of the system, P. Since both P and H_{eff} are not known, an iterative process is used until self–consistency is reached[19]:

$$P = \frac{\exp\left(\frac{-W}{k_B T}\right)}{\sum_N \exp\left(\frac{-W}{k_B T}\right)} \tag{3.6}$$

$$W = \frac{\partial H_{eff}}{\partial P} \tag{3.7}$$

where k_B and T are the Boltzmann constant and the temperature, respectively, W is the intermolecular potential, and the summation is run over all the N possible states of the system.

3.3 Applications in Materials Science

The procedure briefly presented above can be used in a variety of contexts in both materials science and biology. All have been indeed used to model synthetic and biopolymers, complex and simple liquids, surfactants, and mixtures of them[4, 5, 6, 7]. Below, three examples of such applications in materials science are presented.

3.3.1 Predicting Self-Assembly Properties for Amphiphilic Copolymers

Amphiphilic copolymers are polymer chain formed by more than one type of monomer each with a different polarity. When dissolved in solvents, they self-assemble in a variety of morphologies which are function of the relative chemical affinity of the solvent and monomers and the chain microstructure (the monomer sequence within the chain). Such self-assembled nanometric structure can then be used in a variety

of applications from drug and gene delivery agents to emulsion stabilizers[20]. The *a priori* prediction of their phase diagram is very difficult to achieve, and modeling can help in avoiding the synthesis of many different materials. Using an atomistic model for such simulations is however not possible; therefore CG models have become a popular choice. Mainly using DPD[21] and thermodynamics model[22], it has been recently possible to follow at the self-assembly process and build entire phase diagrams for such systems[21].

3.3.2 Predicting the Long-Time Dynamics of Ionic Liquids

Ionic liquids (ILs) are high molecular weight ion pairs formed by organic cations and bulky counterions. They are liquid at room temperature and below so that they can be exploited in a considerable number of extraction and other chemical processes. Knowing the dynamics of ILs at low temperature can therefore improve the design of the ideal solvent for a specific extraction process; however such experiments are very difficult to carry out since the ILs viscosity increases very rapidly lowering the temperature[23]. CG modeling can help in gaining such data as the dynamics of the model is sped up compared with the experimental one by a factor which depends on the CG scheme used. Using CG models it has been possible to show that certain type of ILs with long aliphatic chains can self-assemble[24] and that at low temperatures they present an increased dynamic heterogeneity which is due to an increasing number of slow ions, while the amount of fast ions stays almost constant with temperature[25].

3.3.3 Calculating Interphase Thickness of Polymer Films on Solid Surfaces

When in contact with a solid surface, polymers exhibit complex behavior and understanding and characterizing this behavior is important in the design and development of nanostructures. Such systems are difficult to model as interactions at both small and long length scales need to be included.

A common approach to studying these systems therefore is to propagate information from the smaller length scales into the larger-scale models, and such an approach has been used for modeling polystyrene in contact with a gold surface[26]. Using data from atomistic-scale simulations of small amounts of the nanostructure, a coarser model was constructed (see Fig. 3.1). This model had reduced the number of particles in the polymer by means of coarse graining and also modeled the gold surface implicitly. This meant that each particle in the polymer interacted with the surface through a single function of normal distance instead of being a sum of interactions with many particles in the surface. This mesoscale model enabled the

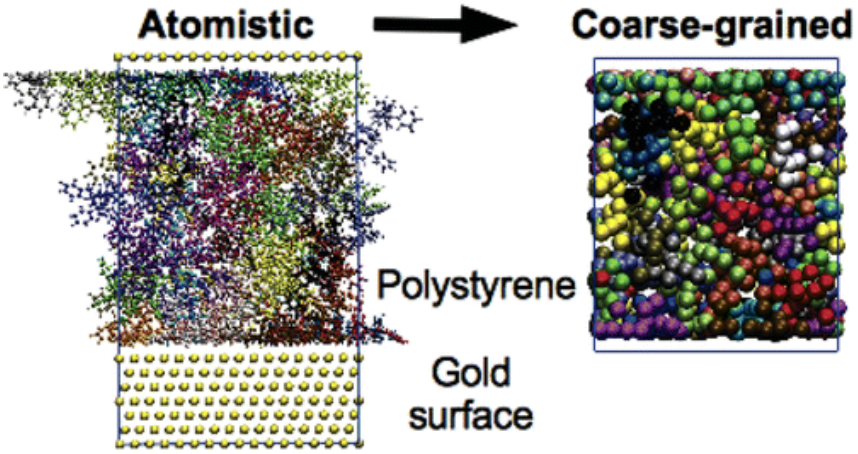


Figure 3.1: An example of the reduction in detail possible when using coarse graining.

simulations of much larger volumes and led to the identification of an interphase thickness in which the polymer presents structural properties different than those in bulk.

3.4 Future Directions

Coarse-graining techniques have become powerful and well-established tools to access large length and timescales. The use of such techniques however is not without any drawbacks. In certain cases the degrees of freedom coarse-grained away in the mesoscale models represent crucial details without which the behavior of the models no longer faithfully recreate the phenomena that are to be studied. This is, for example, the case in molecular systems where short-range directional non-bonded interactions such as hydrogen bonds are present[4].

To circumvent this limitation typical of traditional coarse-graining techniques, the possibility of mixing both coarser and finer levels of detail in a single simulation in so-called hybrid-scale simulations has been recently explored. These two levels of detail are present simultaneously in the model allowing specific parts of the system to be modeled at a higher level of detail while allowing other parts to use a computationally cheaper model[27].

Two main approaches can be identified for such hybrid-scale simulations. Indeed one can mix either CG and atomistic force fields or particle-based (CG or atomistic) and field models. Both approaches are briefly explained below.

3.4.1 Hybrid Atomistic/Coarse-Grained Models

In models where atoms and CG beads are simultaneously present, in order to calculate the interactions between pairs of atoms or beads, preexisting standard force

fields can be used[28]. However, interactions between atoms and beads, the so-called cross terms, must also be evaluated. These cross term interactions could be parameterized using one of the previously described coarse-graining methods; however this is an undesirable option because it would greatly aggravate the work required to derive the model and increase the number of parameters within the model. Instead, a commonly used solution is the use of virtual sites (VS) which are geometrical points identified in the finer (atomistic) resolved regions of the model which collect the interactions coming from the model coarser parts. If the CG model is derived from the atomistic one, as it is in the case of structural based or force matching models, the CG mapping scheme is built on the chain atomic structure and each bead center of mass corresponds either to a specific atom or to the center of mass of all the atoms lumped up in it. Thus, it is easy to identify in the atomistic resolved part of the model the positions of the virtual beads and use them as pinning position to collect any mixed interactions. The VS can therefore be seen as coarse-grained representation of a group of atoms, placed at their center of mass, which facilitate their interaction with coarse-grained beads. When a coarse-grained particle interacts with the atoms of a certain group, it is with the virtual site representation of that group. To ensure that all particles can interact with each other, all atoms must be mapped to such a virtual site.

Once all such pairwise interactions have been evaluated, the net force on each virtual site is transferred onto the constituent atoms. This is done on a mass-weighted basis, so that heavier atoms get a larger share of the force from the virtual site. This process ensures that Newtons third law is obeyed and, when used in molecular dynamics simulations, that linear momentum is conserved:

$$\mathbf{F}_{atom} = \mathbf{F}_{VS}\frac{m_{atom}}{m_{VS}} \tag{3.8}$$

where $\mathbf{F}$ refers to the net force and m the mass of the particle. Using this method, non-bonded interactions no longer cross between the different resolutions and the coarse-grained force field can be used to describe and parameterize these interactions. This means that once an atomistic and coarse-grained force field is chosen, it is also possible to start creating hybrid-scale models.

Using virtual sites, models which mix both atoms and coarse-grained beads can be constructed. In these models selected chemical motifs are kept at a fine level of detail, while all other sections are coarse–grained. The choice of where to leave fine levels of detail is left to intuition, typically dipoles, including hydrogen bonding sites, or areas where steric factors are of great importance are kept at fine resolution. This approach allows a detailed description of some parts of the model to be combined with a less computationally intensive description of other parts of the model[28, 29].

The Hamiltonian for the simulation system can be defined as

$$\mathcal{H} = \mathcal{K}_{atoms} + \mathcal{K}_{CG} + V_{bonds} + V_{nb(CG-CG)} + V_{nb(AA-AA)} + V_{nb(VS-CG)} \tag{3.9}$$

where $\mathcal{K}$ refers to the contribution from kinetic energy and V refers to potential energy between particles.

3.4.2 Adaptive Resolution Models

If what needs to be modeled at high resolution is a particular recurring chemical motif in the molecular system, then the model resolution can be static, which means that during the construction of the model, one decides which chemical moieties are modeled with the different resolutions and this model choice is kept for the entire simulation. However, in some cases it is more suited to define a geometrical region within the simulation box which is modeled with a different resolution. In this case, the total number of degrees of freedom of the model will change over the time depending on how many chemical species enter or leave that region (or regions)[30]. Usually systems are constructed to have a region of interest containing molecules modeled at atomic details, surrounded by a region of coarse-grained molecules. As simulations of such systems progress, the molecules are free to move between these two regions of different resolution, and a method to allow the transition between the two model resolutions is therefore required. To handle such transition, a buffer region (known as healing region) between the two levels of detail is used which allows the molecules to gradually change from one level of detail to another. In this healing region, the models switch from using one force field to the other as a function of their position within the healing region

$$V = \lambda V_{AA} + (1 - \lambda) V_{CG} \tag{3.10}$$

where λ refers to a switching function between the two force fields. Great care must be taken to ensure that there is no discontinuity in the force acting on the particles, as this would create mass transfer between different regions of the simulation box. Transitioning from the coarse region into the fine detail region is particularly difficult as it involves reintroducing the degrees of freedom that have previously been discarded.

3.4.3 Hybrid Particle Field Models

As the computational cost of both molecular dynamics and Monte Carlo simulation methods scales as $\mathcal{O}(N^2)$, increasingly larger simulations become too demanding to

produce results in a timely manner. An alternative to this particle-based approach is hybrid models where the particles surrounding a specific atoms or CG bead is replaced by a continuum field and the interparticle non-bonded interactions are calculated with respect to that external field. This method scales as $\mathcal{O}(N)$, making the simulation of extremely large systems possible.

This approach is becoming popular in modeling polymer blends or solutions where the polymer chains self-assemble creating specific morphologies. A commonly used field method for simulating such polymer nanostructures is the single-chain mean field theory (SCMF)[31]. In this, single polymer chains or large molecules are separated and each determines its non-bonded interaction with respect to an external field rather than having to interact with every other particle in every other molecule. To construct the mean field, self-consistent field (SCF) theory is used, where the particle density of a given chemical species (ϕ_A) is measured at all points on a discrete grid. The force caused by the mean field for a particle of species A at a given position is then given as

$$F_A(r) = -k_B T \sum_{A'} \chi_{AA'} \frac{\partial \phi_{A'}(r)}{\partial r} - \frac{1}{\kappa} \left(\sum_{A} \frac{\partial \phi_A(r)}{\partial r} - 1 \right) \tag{3.11}$$

where k_B and T are the Boltzmann constant and temperature. The term $\chi_{AA'}$ is a Flory-Huggins mixing parameter between two species and must be defined between all combinations of species and κ represents the compressibility of the system. The first term defines how favorably different species can mix, while the second term tries to balance the particle density. This field varies both spatially and temporally and must be updated as the simulation progresses. However the variation of the field is slow with respect to the intramolecular forces, meaning that several particle moves can be done before the field must be updated. Within this approach while the intermolecular interactions are calculated via the SCMF theory, the intramolecular interactions (i.e., bonds, angles, and torsions) are handled at the same level of precision as conventional particle-based simulations. Initial work on this method used Monte Carlo to perform the particle moves[32] as the use of Monte Carlo allows the possible conformational space of the molecules to be sampled effectively. More recently molecular dynamics has also been used[33]. Since the calculation of the interactions with a discrete grid is computationally quick to perform, with each particle only interacting with eight different points rather than all their neighbors, this method is also particularly well suited to the increasing parallel nature of the hardware available, with some software able to achieve hyper-parallelism (achieving a speedup greater than n using n processors)[33].

Bibliography

[1] R. J. Gowers and P. Carbone, "Coarse-grained and hybrid simulations of nanostructures," in *Encyclopedia of Nanotechnology* (B. Bhushan, ed.), pp. 1–10, Dordrecht: Springer Netherlands, 2014.

[2] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*. New York, NY, USA: Clarendon Press, 1989.

[3] J. W. Ponder and D. A. Case, "Force fields for protein simulations," *Advances in protein chemistry*, vol. 66, pp. 27–85, 2003.

[4] H. A. Karimi-Varzaneh, N. F. A. van der Vegt, F. Müller-Plathe, and P. Carbone, "How good are coarse-grained polymer models? A comparison for atactic polystyrene," *ChemPhysChem*, vol. 13, no. 15, pp. 3428–3439, 2012.

[5] G. Voth, *Coarse-Graining of Condensed Phase and Biomolecular Systems*. Boca Raton, FL: CRC Press/Taylor and Francis Group, 2009.

[6] P. Carbone and C. Avendaño, "Coarse-grained methods for polymeric materials: enthalpy- and entropy-driven models," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 4, no. 1, pp. 62–70, 2014.

[7] W. G. Noid, "Perspective: Coarse-grained models for biomolecular systems," *The Journal of Chemical Physics*, vol. 139, no. 9, p. 090901, 2013.

[8] A. P. Lyubartsev and A. Laaksonen, "Calculation of effective interaction potentials from radial distribution functions: A reverse monte carlo approach," *Phys. Rev. E*, vol. 52, pp. 3730–3737, 1995.

[9] J. Baschnagel, K. Binder, P. Doruker, A. Gusev, O. Hahn, K. Kremer, W. Mattice, F. Müller-Plathe, M. Murat, W. Paul, S. Santos, U. Suter, and V. Tries, "Bridging the gap between atomistic and coarse-grained models of polymers: Status and perspectives," in *Advances in Polymer Science: Viscoelasticity Atomistic Models Statistical Chemistry*, vol. 152 of *ADVANCES IN POLYMER SCIENCE*, pp. 41–156, Springer, 2000.

[10] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, "The MARTINI force field: Coarse grained model for biomolecular simulations," *The Journal of Physical Chemistry B*, vol. 111, no. 27, pp. 7812–7824, 2007.

[11] W. Shinoda, R. Devane, and M. Klein, "Multi-property fitting and parameterization of a coarse grained model for aqueous surfactants," *Molecular Simulation*, vol. 33, no. 1-2, pp. 27–36, 2007.

[12] C. Avendaño, T. Lafitte, A. Galindo, C. S. Adjiman, G. Jackson, and E. A. Müller, "Saft-γ force field for the simulation of molecular fluids. 1. A single-site coarse grained model of carbon dioxide," *The Journal of Physical Chemistry B*, vol. 115, no. 38, pp. 11154–11169, 2011.

[13] F. Ercolessi and J. B. Adams, "Interatomic Potentials from First-Principles Calculations: The Force-Matching Method," *EPL (Europhysics Letters)*, vol. 26, pp. 583–588, 1994.

[14] W. G. Noid, J.-W. Chu, G. S. Ayton, V. Krishna, S. Izvekov, G. A. Voth, A. Das, and H. C. Andersen, "The multiscale coarse-graining method. I. A rigorous bridge between atomistic and coarse-grained models," *The Journal of Chemical Physics*, vol. 128, no. 24, p. 244114, 2008.

[15] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[16] M. S. Shell, "The relative entropy is fundamental to multiscale and inverse thermodynamic problems," *The Journal of Chemical Physics*, vol. 129, no. 14, p. 144108, 2008.

[17] R. D. Groot and P. B. Warren, "Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation," *The Journal of Chemical Physics*, vol. 107, no. 11, pp. 4423–4435, 1997.

[18] P. Español, "Dissipative particle dynamics.," in *Handbook of Materials Modeling* (S. Yip, ed.), ch. 8, pp. 2503–2512, Dordrecht: Springer Netherlands, 2005.

[19] A. Altland and B. D. Simons, *Condensed Matter Field Theory*. Cambridge: Cambridge University Press, 2 ed., 2010.

[20] P. Alexandridis and B. Lindman, *Amphiphilic Block Copolymers Self-Assembly and Applications*. Amsterdam: Elsevier, 2000.

[21] V. Ortiz, S. O. Nielsen, D. E. Discher, M. L. Klein, R. Lipowsky, and J. Shillcock, "Dissipative particle dynamics simulations of polymersomes," *The Journal of Physical Chemistry B*, vol. 109, no. 37, pp. 17708–17714, 2005.

[22] S. Nawaz and P. Carbone, "Coarse–graining poly(ethylene oxide)–poly(propylene oxide)–poly(ethylene oxide) (PEO–PPO–PEO) block copolymers using the MARTINI force field," *The Journal of Physical Chemistry B*, vol. 118, no. 6, pp. 1648–1659, 2014.

[23] R. D. Rogers and K. R. Seddon, "Ionic liquids–solvents of the future?," *Science*, vol. 302, no. 5646, pp. 792–793, 2003.

[24] Y. Wang and G. A. Voth, "Unique spatial heterogeneity in ionic liquids," *Journal of the American Chemical Society*, vol. 127, no. 35, pp. 12192–12193, 2005.

[25] H. A. Karimi-Varzaneh, F. Müller-Plathe, S. Balasubramanian, and P. Carbone, "Studying long-time dynamics of imidazolium-based ionic liquids with a systematically coarse-grained model," *Physical Chemistry Chemical Physics*, vol. 12, pp. 4714–4724, 2010.

[26] K. Johnston and V. Harmandaris, "Hierarchical multiscale modeling of polymersolid interfaces: Atomistic to coarse-grained description and structural and conformational properties of polystyrenegold systems," *Macromolecules*, vol. 46, no. 14, pp. 5741–5750, 2013.

[27] C. F. Abrams, L. Delle Site, and K. Kremer, "Dual-resolution coarse-grained simulation of the bisphenol–A–polycarbonate/nickel interface," *Phys. Rev. E*, vol. 67, p. 021807, 2003.

[28] A. J. Rzepiela, M. Louhivuori, C. Peter, and S. J. Marrink, "Hybrid simulations: combining atomistic and coarse-grained force fields using virtual sites," *Physical Chemistry Chemical Physics*, vol. 13, no. 22, pp. 10437–10448, 2011.

[29] N. Di Pasquale, D. Marchisio, and P. Carbone, "Mixing atoms and coarse-grained beads in modelling polymer melts," *The Journal of Chemical Physics*, vol. 137, no. 16, p. 164111, 2012.

[30] M. Praprotnik, L. Delle Site, and K. Kremer, "Multiscale simulation of soft matter: From scale bridging to adaptive resolution," *Annual Review of Physical Chemistry*, vol. 59, no. 1, pp. 545–571, 2008.

[31] M. Müller and J. J. de Pablo, "Computational approaches for the dynamics of structure formation in self-assembling polymeric materials," *Annual Review of Materials Research*, vol. 43, no. 1, pp. 1–34, 2013.

[32] K. C. Daoulas, M. Müller, J. J. de Pablo, P. F. Nealey, and G. D. Smith, "Morphology of multi-component polymer systems: single chain in mean field simulation studies," *Soft Matter*, vol. 2, pp. 573–583, 2006.

[33] G. Milano and T. Kawakatsu, "Hybrid particle-field molecular dynamics simulations for dense polymer systems," *The Journal of Chemical Physics*, vol. 130, no. 21, p. 214106, 2009.

Chapter 4

Multiscale molecular dynamics

4.1 Introduction

Multiscale modelling refers to the exploration of a scientific problem through approaching it using models of different scales, blending together the strengths of each approach. For computational modelling, a variety of different well established techniques exist with Figure 4.1 showing the length and time scale they operate on. The length and time scale they produce results for is inherently linked, with techniques which model smaller length scales yielding small amounts of modelled time as well.

Molecular dynamics (MD) is a powerful tool for the study of soft matter systems, where properties have characteristic lengths of angstroms to tens of nanometers and the energy of changes in the system is around 5 k_BT[2]. Atomistic scale MD can

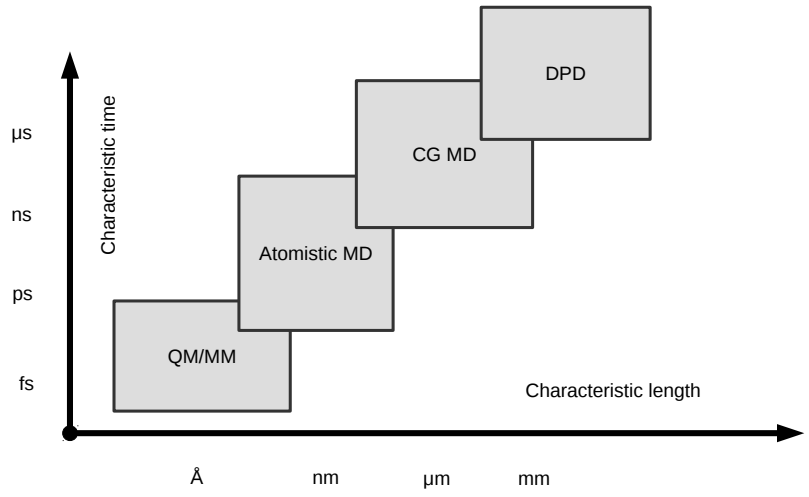


Figure 4.1: The different time and length scales accessible with various particle based modelling techniques. Adapted from Ref [1].

often struggle to model large enough length and time scales to properly encapsulate a problem[3], whilst coarse-grained (CG) MD, which can sample two orders of magnitude more of both scales, often lacks many important details in its results. Multiscale modelling techniques can be used to blur the boundaries between these two scales, allowing a wider range of length and time scales to be explored.

The area of interest for this research is the interface between atomistic and CG models and this Chapter aims to give a review of some of the approaches that are actively being developed. Section 4.2 introduces dual scale models which are used throughout the rest of this research while Section 4.3 gives an overview of some complimentary multiscale methods.

4.2 Dual scale models

The most direct approach to multiscale modelling is to simply construct a model that simultaneously has elements from two distinct levels of resolution. These models were first proposed by Christen and van Gunsteren[4], and have been referred to as a variety of names, including hybrid, multigrained, multiresolved and dual scale. These models consist of a mix of atoms (also known as fine particles) and larger beads (also known as coarse particles or superatoms), which represent many atoms.

With these particles coexisting within the same simulation volume, the force field needs to define how to model interactions between these different resolution particles. Attempts have been made to simply extend existing force fields to have parameters for cross resolution particle interactions, either through extending the CG force field parameterisation to include cross resolution terms[5] which increases the parameter space, or through application of the atomistic force field mixing rules between atoms and beads[6], which assumes some sort of similarity between the potentials.

4.2.1 Virtual sites

To prevent the need for cross resolution interactions there needs to be some sort of proxy between the two scales, which mediates interactions between particles of different resolution. One solution to this is the virtual site method first proposed by Rzepiela and coworkers[7].

As previously stated, in an atomistic resolution model the total force on each atom is given as a sum over its interaction with all other atoms:

$$\mathbf{F}_i = \sum_j \mathbf{F}_{ij}\left(r_{ij}\right) \tag{4.1}$$

Where $\mathbf{F}$ denotes the force, i and j refer to the indices of the atoms and $\mathbf{r}_{ij}$ the distance between them. We can also arbitrarily group atoms such that each atom belongs to one and only one group and perform the same summation over each group and then atoms in each group:

$$\mathbf{F}_{i\in\alpha} = \sum_{\beta}\sum_{j\in\beta}\mathbf{F}_{ij}\left(r_{ij}\right) \tag{4.2}$$

The groupings that we will use will be a CG mapping scheme. All atoms are mapped into an alternate imaginary representation in CG space, known as a virtual site (VS). The position of each VS is given as the center of mass of its constituent atoms:

$$\mathbf{R}_{\alpha} = \frac{1}{M_{\alpha}}\sum_{i\in\alpha} m_i\mathbf{r}_i \tag{4.3a}$$

$$M_{\alpha} \equiv \sum_{i\in\alpha} m_i \tag{4.3b}$$

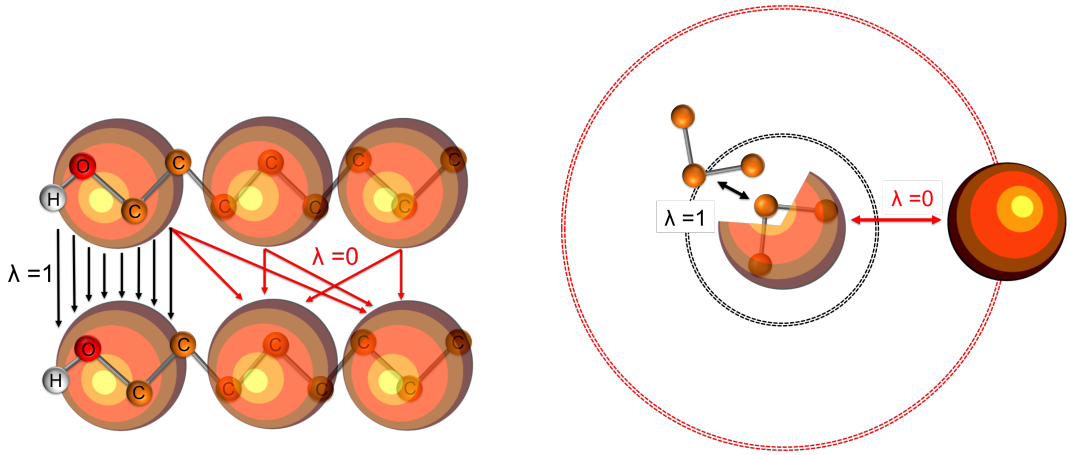
Where $\mathbf{r}$ and $\mathbf{R}$ refer to positions in atomistic and CG space respectively, m is the mass, lower case indices run over atoms and greek lower case indices run over VS particles.

We can then rewrite Equation 4.2 to allow contributions from forces calculated in both atomistic space and CG space:

$$\mathbf{F}_i = \sum_{\beta}\left(\sum_{j\in\beta}\lambda_{\alpha\beta}\mathbf{F}_{ij}^{atomistic}(\mathbf{r}_{ij}) + \frac{m_i}{M_{\alpha}}(1-\lambda_{\alpha\beta})\mathbf{F}_{\alpha\beta}^{CG}(\mathbf{R}_{\alpha\beta})\right) \tag{4.4}$$

Where $i \in \alpha$ and the superscripts indicate that parameters for forces can come from either the atomistic or CG force field. λ is a switching function defined between each VS pair that controls in which coordinate space interactions are calculated. If $\lambda = 1$, then the original atomistic force field is recovered, whilst decreasing λ introduces more of the CG force field. The choice of λ is the defining feature of different dual scale methods and is discussed in further detail below. It is important to note that λ is defined at the VS level with the atoms inheriting from their VS, i.e. $\lambda_{ij} \equiv \lambda_{\alpha\beta}$ with $i \in \alpha$ and $j \in \beta$. This ensures that the interactions between two sets of atoms are not counted twice.

Using Equation 4.4, all force on VS particles gets transferred onto the underlying atoms, as by definition $\sum_{i\in\alpha} {}^{m_i}/_{M_{\alpha}} \equiv 1$. Therefore the equations of motions are not applied to the VS and the positions of the atoms within the system can be updated according to the standard equations of motion. The positions of the VS move through the continual application of Equation 4.3.



(a) Type dependent λ. The level of detail of interactions is defined by the types of the VS.

(b) Range dependent λ. Nearby interactions involve many atoms while longer interactions use virtual site representations.

Figure 4.2: Visualising different schemes for dual resolution models

4.2.2 Approaches to mixing force fields

Using this methodology we can define a dual scale model through an atomistic set of positions, a mapping scheme to translate this into a CG representation and the parameter λ. Three methods for defining λ will be outlined, each with their own merits. These all seek to allow the crucial details in a model to be faithfully recreated, while in other places allowing a CG description to be substituted in.

Type based dual scale

The most common approach in designing a dual scale model is to define λ on the basis of the types of the VS involved[8, 9]. All VS particles are assigned into types based on their chemical nature, certain VS type pairings are defined coarse, so $\lambda_{\alpha\beta} = 0$, while others pairings are defined atomistic with $\lambda_{\alpha\beta} = 1$.

This allows for certain pairings of VS types to be treated with atomistic detail, while other pairings are treated with fewer degrees of freedom. Some VS pairs are poorly modelled using only the two coordinates due to the rotational degrees of freedom between them and therefore these would be better modelled with $\lambda = 1$. For instance when dealing with a simple alcohol molecule, shown in Figure 4.2a, the interactions between the VS particles containing the OH group is defined as atomistic, while all other pairings are CG. Unlike the other schemes that will be mentioned, in this scheme λ remains constant throughout the simulation.

Range based dual scale

It is also possible to define λ based on the distance between two VS particles[10, 11], so that $\lambda_{\alpha\beta} = \lambda_{\alpha\beta}(\mathbf{R}_{\alpha\beta})$, shown in Equation 4.5. With this scheme at short

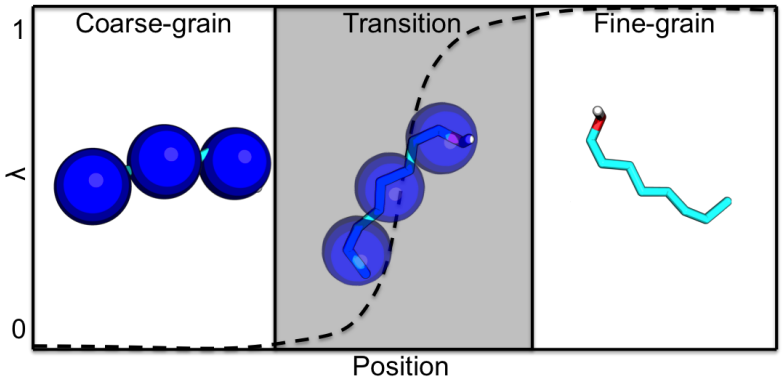


Figure 4.3: Adaptive resolution simulations. Particles change their resolution based on their position within the simulation box. Adapted from Ref [14].

distances all interactions are calculated at atomistic detail while at larger distances interactions are calculated at CG detail using the VS, as shown in Figure 4.2b. As originally described in Section 2.3.2, a gradual change between the two force fields is necessary to prevent any abrupt changes in the energy and force.

$$\lambda_{\alpha\beta}(\mathbf{R}_{\alpha\beta}) = \begin{cases} 1 & \mathbf{R}_{\alpha\beta} \leq r_{atomistic} \\ 0 < \lambda < 1 & r_{atomistic} < \mathbf{R}_{\alpha\beta} < r_{coarse} \\ 0 & \mathbf{R}_{\alpha\beta} \geq r_{coarse} \end{cases} \tag{4.5}$$

The rationale behind this approach is at closer distances the extra degrees of freedom allow for a more accurate description of the interaction, while at longer distances this level of detail is unnecesary. The downside of this approach is that in the intermediate region both the atomistic and CG force field contributions are calculated, making this more computationally expensive than just the atomistic system.

Adaptive resolution models

Finally, the most ambitious dual scale scheme is the adaptive resolution or AdResS scheme, first proposed by Praprotnik and coworkers[12, 13]. In this method, the λ between two particles is based on their positions within the simulation volume, so $\lambda_{\alpha\beta} = \lambda_{\alpha\beta}(\mathbf{R}_\alpha, \mathbf{R}_\beta)$. This allows a region of the simulation volume to be designated to exist in atomistic detail while the volume surrounding this exists in CG detail, with molecules free to move between these regions. Between these two regions a buffer healing region is required to allow molecules to gradually change the resolution of their representation, and avoid step changes in the force field. This setup is visualised in Figure 4.3.

In order for molecules to freely move between the atomistic and coarse regions it is necessary for the regions to be at equilibrium with each other, otherwise an artificial

interface would exist, causing unnatural behaviour. The reintroduction of atomistic degrees of freedom as particles increase their λ however, has been shown to cause an interface[15]. To resolve this the magnitude of the deviation in a particular property between the atomistic and CG regions is calculated, and then an external field is applied to the healing region to correct the deviation. This approach has been used to correct various target properties including chemical potential[16], pressure[17], and the total energy[18].

These approaches however are mutually exclusive, therefore all properties cannot be corrected, meaning that adjusting the chemical potential leads to unconserved energy and vice versa. Currently there is no clear consensus as to which of these different approaches is the most correct. Furthermore, this correction step also presents an additional layer of parameterisation compared to other dual scale models.

4.2.3 Choice of CG force field

Dual scale models are typically built starting with an atomistic model and then adding a CG force field to this, such as those described in Chapter 3. It has been shown that strictly speaking, the CG force field should reproduce the same distributions as the atomistic force field[19]. This would favour using a "bottom up" force field where parameters are fitted to match the atomistic system and both iterative Boltzmann inversion (IBI)[20] and force-matching[5] have previously been used. However, "top down" force fields such as Martini have also been used successfully and have been able to reproduce the thermodynamics properties which they were parameterised around[7, 8]. Therefore the choice of which CG force field to employ can be made freely according to the usual arguments for what is best for the problem at hand.

4.3 Alternative multiscale models

4.3.1 The ELBA coarse grained model

Reintroducing atoms to a CG model is not the only method for bringing back the details they represent. For a group of atoms with a dipole, when the atoms are mapped to a single coordinate the most valuable piece of information lost is the direction of this dipole. The ELBA coarse grained model represents particles as both a coordinate and a vector[21], which allows for a description of both their position and the direction they are facing, as shown in Figure 4.4. This model is then loosely multiscale as the position defines the position of the large scale details while the vector is used as a representation of the smaller scale details.

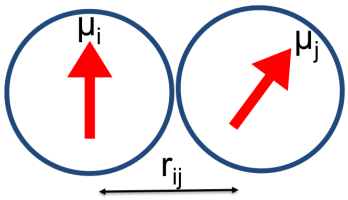


Figure 4.4: Two interacting particles in the ELBA model

The particles in the ELBA model have both a normal Lennard-Jones potential between them and a dipole potential shown in Equation 4.6. The dipole term is a function of the pairwise distance, r_{ij} while also taking into account the orientation, each given as the vector $\boldsymbol{\mu}$, of the two particles.

$$V_{dipole} = \frac{1}{4\pi\epsilon_0}\left[1 - 4\left(\frac{r_{ij}}{r_{cut}}\right)^3 + 3\left(\frac{r_{ij}}{r_{cut}}\right)^4\right]\left[\frac{1}{r_{ij}^3}(\boldsymbol{\mu}_i \cdot \boldsymbol{\mu}_j) - \frac{3}{r_{ij}^5}(\boldsymbol{\mu}_i \cdot \mathbf{r}_{ij})(\boldsymbol{\mu}_j \cdot \mathbf{r}_{ij})\right] \tag{4.6}$$

This model has been able to correctly reproduce many structural and thermodynamic properties of water[22], something which even some atomistic models struggle to do. In addition, it is possible to mix atomistic solutes with ELBA water[23] allowing for solvation with a computationally less expensive solvent.

4.3.2 Parallel multiscale models

All approaches so far have consisted of a single multiscale simulation with both scales of model directly interacting with each other. An alternative to this paradigm is a multiscale approach formed by parallel simulations of different scale which periodically exchange information[24].

Backmapping approaches

CG simulations are able to explore the phase space of possible configurations much faster than the corresponding atomistic simulations. This increased speed comes from both the simulations being computationally cheaper to perform and the modelled diffusion of molecules being much higher. In real terms, this high speed traversal of phase space corresponds to firstly a higher probability of fulfilling the ergodic hypothesis described in Section 2.5.4 and secondly increasing the speed at which a representative sample can be gathered.

The weakness of results generated from these simulations is that they feature no atomistic details. A multiscale approach to this problem is therefore to periodically take structures predicted by a CG approach and translate these structures into an atomistic description, as shown in Figure 4.5. The process of translating a CG

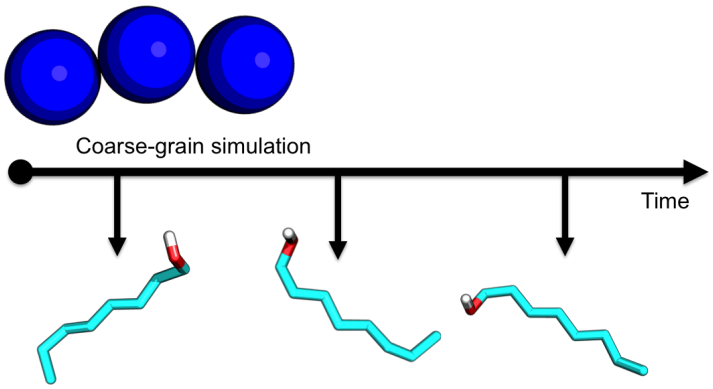


Figure 4.5: Backmapping many atomistic systems from a long coarse grained simulation

structure into a higher resolution form is known as backmapping and is a complex challenge for larger molecules such as polymers[25, 26]. To perform this task it is often necessary to employ many heuristics based upon expectations of the resultant structure, and therefore many specialised tools have been created for each field[27, 28, 29]. Unfortunately, it is not always possible to backmap a given structure to atomistic resolution as a CG model might enter conformational space which is either ambiguous or impossible to backmap from[30].

Resolution exchange models

A more continuous approach to sampling from multiple noninteracting simulations is replica exchange molecular dynamics or REMD[31]. This approach typically uses multiple parallel simulations of identical systems at different temperatures and periodically exchanges the coordinates of two simulations provided that the resulting probability of each new state is acceptable. In order to make proposed exchanges of coordinates probable, the gap in temperature between different simulations must be small, therefore multiple replicas at incremental temperatures are used.

As the system is more mobile at higher temperatures due to the higher energy, the simulations at higher temperatures are able to avoid minima which the lower temperature simulations may get stuck in for extended periods of time. This allows for a much improved sampling at the lowest temperature simulation, which benefits from being moved into different energy minima, known as basin hopping.

For the purposes of multiscale modelling, parallel simulations of the same system modelled at different levels of resolution can be performed. The coordinates of higher and lower resolution representations can be exchanged based on their proba-

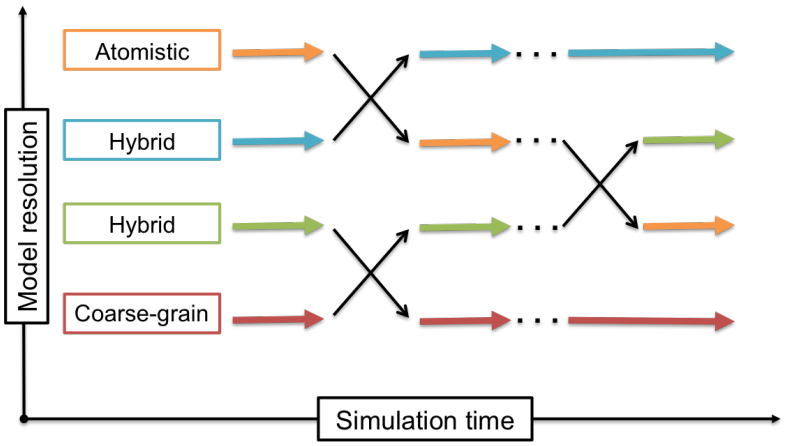


Figure 4.6: Visualising the exchange of coordinates of parallel multiscale simulations using REMD

bilities using similar rules to the previously described scheme. Simulations at lower resolution are able to more easily explore complex energy landscapes, allowing them to perform the role usually filled by high temperature simulations in REMD.

Using only atomistic and CG models for this would result in a very low probability of an exchange being accepted. Therefore techniques are required to represent the various stages between these two states to improve the acceptance ratio: Different parts of a model can be exchanged between resolutions at different times to allow for smaller overall changes in the system[32]; fractional representations between resolutions can be used, similar to the adaptive resolution approach[4]; the structures can be energy minimised to allow them to adjust to the new force field before the exchange is attempted[33].

4.4 Conclusion

The choice of which of the different dual scale approaches presented is best for a given application is highly dependent on the geometry of the system. The rest of this work focuses on the use of type based λ models, which are best suited to molecules which have identifiable areas which benefit from more degrees of freedom in their model.

If instead no such distinction can be made, a range dependent λ model could offer a better solution as the rotational degrees of freedom will have a larger influence on the potential at shorter distances. To date this type of model has not been widely used.

Finally, adaptive resolution models offer an option for systems where there is a region of the simulation volume which requires extra attention. Examples of this include the active site of biological molecules, adsorption onto surfaces or interfaces between two phases.

Ultimately, the standard by which the long term success of dual scale models will be judged is not against the original atomistic or CG models but instead the alternate novel techniques that are being developed simultaneously. Parallel multiscale techniques allow for separate simulations to be coupled, with the advantage that fewer compromises to the force field have had to be made in their setup as each individual simulation can use a "pure" implementation. Finally, models such as the ELBA model offer novel ways to express the different length scales in a model and that perhaps a rethink in how we construct particle based models is required to properly capture their behaviour.

Bibliography

[1] D. Andrienko and K. Kremer, "Simulations," in *Macromolecular Engineering: Precise Synthesis, Materials Properties, Applications.* (K. Matyjaszewski, Y. Gnanou, and L. Leibler, eds.), KGaA: WILEY-VCH Verlag, 2007.

[2] C. Peter and K. Kremer, "Multiscale simulation of soft matter systems," *Faraday Discuss.*, vol. 144, pp. 9–24, 2010.

[3] J. J. de Pablo, "Coarse-grained simulations of macromolecules: From DNA to nanocomposites," *Annual Review of Physical Chemistry*, vol. 62, no. 1, pp. 555–574, 2011.

[4] M. Christen and W. F. van Gunsteren, "Multigraining: An algorithm for simultaneous fine-grained and coarse-grained simulation of molecular systems," *The Journal of Chemical Physics*, vol. 124, no. 15, 2006.

[5] Q. Shi, S. Izvekov, and G. A. Voth, "Mixed atomistic and coarse-grained molecular dynamics: simulation of a membrane-bound ion channel," *The Journal of Physical Chemistry B*, vol. 110, no. 31, pp. 15045–15048, 2006.

[6] J. Michel, M. Orsi, and J. W. Essex, "Prediction of partition coefficients by multiscale hybrid atomic-level/coarse-grain simulations," *The Journal of Physical Chemistry B*, vol. 112, no. 3, pp. 657–660, 2008.

[7] A. J. Rzepiela, M. Louhivuori, C. Peter, and S. J. Marrink, "Hybrid simulations: combining atomistic and coarse-grained force fields using virtual sites," *Phys. Chem. Chem. Phys.*, vol. 13, pp. 10437–10448, 2011.

[8] T. A. Wassenaar, H. I. Ingólfsson, M. Prieß, S. J. Marrink, and L. V. Schäfer, "Mixing MARTINI: Electrostatic coupling in hybrid atomisticcoarse-grained biomolecular simulations," *The Journal of Physical Chemistry B*, vol. 117, no. 13, pp. 3516–3530, 2013.

[9] R. J. Gowers and P. Carbone, "A multiscale approach to model hydrogen bonding: The case of polyamide," *The Journal of Chemical Physics*, vol. 142, no. 22, 2015.

[10] L. Shen and H. Hu, "Resolution-adapted all-atomic and coarse-grained model for biomolecular simulations," *Journal of Chemical Theory and Computation*, vol. 10, no. 6, pp. 2528–2536, 2014.

[11] S. Izvekov and G. A. Voth, "Mixed resolution modeling of interactions in condensed-phase systems," *Journal of Chemical Theory and Computation*, vol. 5, no. 12, pp. 3232–3244, 2009.

[12] M. Praprotnik, L. Delle Site, and K. Kremer, "Adaptive resolution molecular-dynamics simulation: Changing the degrees of freedom on the fly," *The Journal of Chemical Physics*, vol. 123, no. 22, 2005.

[13] M. Praprotnik, K. Kremer, and L. Delle Site, "Adaptive molecular resolution via a continuous change of the phase space dimensionality," *Phys. Rev. E*, vol. 75, p. 017701, Jan 2007.

[14] N. Goga, A. Rzepiela, M. N. Melo, A. de Vries, A. Hadar, A. Markvoort, S. Nedea, and H. Berendsen, *Methods for multiscale modeling of membranes*, pp. 139–170. Academic Press, 2012.

[15] M. Praprotnik, L. Delle Site, and K. Kremer, "Multiscale simulation of soft matter: From scale bridging to adaptive resolution," *Annual Review of Physical Chemistry*, vol. 59, no. 1, pp. 545–571, 2008.

[16] S. Poblete, M. Praprotnik, K. Kremer, and L. Delle Site, "Coupling different levels of resolution in molecular simulations," *The Journal of Chemical Physics*, vol. 132, no. 11, 2010.

[17] S. Fritsch, S. Poblete, C. Junghans, G. Ciccotti, L. Delle Site, and K. Kremer, "Adaptive resolution molecular dynamics simulation through coupling to an internal particle reservoir," *Phys. Rev. Lett.*, vol. 108, p. 170602, Apr 2012.

[18] R. Potestio, S. Fritsch, P. Español, R. Delgado-Buscalioni, K. Kremer, R. Everaers, and D. Donadio, "Hamiltonian adaptive resolution simulation for molecular liquids," *Phys. Rev. Lett.*, vol. 110, p. 108301, Mar 2013.

[19] N. Goga, M. N. Melo, A. J. Rzepiela, A. H. de Vries, A. Hadar, S. J. Marrink, and H. J. C. Berendsen, "Benchmark of schemes for multiscale molecular dynamics simulations," *Journal of Chemical Theory and Computation*, vol. 11, no. 4, pp. 1389–1398, 2015.

[20] N. Di Pasquale, D. Marchisio, and P. Carbone, "Mixing atoms and coarse-grained beads in modelling polymer melts," *The Journal of Chemical Physics*, vol. 137, no. 16, p. 164111, 2012.

[21] M. Orsi and J. W. Essex, "The ELBA force field for coarse-grain modeling of lipid membranes," *PLoS ONE*, vol. 6, pp. 1–22, 12 2011.

[22] M. Orsi, "Comparative assessment of the ELBA coarse-grained model for water," *Molecular Physics*, vol. 112, no. 11, pp. 1566–1576, 2014.

[23] M. Orsi, W. Ding, and M. Palaiokostas, "Direct mixing of atomistic solutes and coarse-grained water," *Journal of Chemical Theory and Computation*, vol. 10, no. 10, pp. 4684–4693, 2014.

[24] G. S. Ayton, W. G. Noid, and G. A. Voth, "Multiscale modeling of biomolecular systems: in serial and in parallel," *Current Opinion in Structural Biology*, vol. 17, no. 2, pp. 192—198, 2007.

[25] W. Tschöp, K. Kremer, O. Hahn, J. Batoulis, and T. Bürger, "Simulation of polymer melts. ii. from coarse-grained models back to atomistic description," *Acta Polymerica*, vol. 49, no. 2-3, pp. 75–79, 1998.

[26] X. Chen, P. Carbone, G. Santangelo, A. Di Matteo, G. Milano, and F. Müller-Plathe, "Backmapping coarse-grained polymer models under sheared nonequilibrium conditions," *Phys. Chem. Chem. Phys.*, vol. 11, pp. 1977–1988, 2009.

[27] K. Anderson, R. Mukherjee, J. Critchley, J. Ziegler, and S. Lipton, "POEMS: parallelizable open-source efficient multibody software," *Engineering with Computers*, vol. 23, no. 1, pp. 11–23, 2006.

[28] L. E. Lombardi, M. A. Martí, and L. Capece, "CG2AA: backmapping protein coarse-grained structures," *Bioinformatics*, 2015.

[29] M. R. Machado and S. Pantano, "SIRAH tools: mapping, backmapping and visualization of coarse-grained models," *Bioinformatics*, 2016.

[30] V. A. Harmandaris, N. P. Adhikari, N. F. A. van der Vegt, and K. Kremer, "Hierarchical modeling of polystyrene: From atomistic to coarse-grained simulations," *Macromolecules*, vol. 39, no. 19, pp. 6708–6719, 2006.

[31] Y. Sugita and Y. Okamoto, "Replica-exchange molecular dynamics method for protein folding," *Chemical Physics Letters*, vol. 314, no. 12, pp. 141–151, 1999.

[32] E. Lyman and D. M. Zuckerman, "Resolution exchange simulation with incremental coarsening," *Journal of Chemical Theory and Computation*, vol. 2, no. 3, pp. 656–666, 2006.

[33] P. Liu and G. A. Voth, "Smart resolution replica exchange: An efficient algorithm for exploring complex energy landscapes," *The Journal of Chemical Physics*, vol. 126, no. 4, 2007.

Part II

Results

Chapter 5

Introducing a dual scale methodology: Polyamide

5.1 Preface

This Chapter presents the first attempt by the author at using a dual scale model, following an approach first set out by Di Pasquale and Carbone[1], which follows the type based λ methodology described in Section 4.2.2.

The system of choice, polyamide, is a more strenuous test for this methodology for many reasons. Firstly, the previously published models for polymers using this methodology featured very few intramolecular transitions between atomistic and CG. With the polyamide model however there are four atomistic to CG transitions per monomer, meaning that both the treatment of bonded interactions and intramolecular nonbonded interactions will be thoroughly tested. As highlighted in Chapter 4 in a dual scale model the transitions between different levels of resolution need to be seamless. The resulting bonded structure of this model was fair and identified as an area for future improvement.

Secondly, the atomistic model for polyamide had partial charges around the amide group, which had not previously been done with this methodology. These partial charges are a clear example of a moiety in the molecule which would benefit from extra degrees of freedom, making this ideal for the type based λ dual scale approach. Polyamide is able to form hydrogen bonds and these were chosen as a key property to judge the success of the model created. Overall the dual scale model is able to recreate the hydrogen bonding with a fair degree of accuracy and this is one of the more positive findings of this work.

Behind the scenes a large amount of software was created to facilitate this work. The simulations in this Chapter were performed using in house code specfically written for this work to accomodate the novel model. The code was created through

adapting an existing code and the changes made are detailed in Chapter 7. One of the important features of this code is that the execution time scales correctly with the number of particles in the system, which allowed for results to be gathered in a timely manner. In a similar theme, the code was made parallel using technologies described in Chapter 8. This allowed for the computational hardware made available to be fully utilised. Finally, various analysis tools were written and published under an open source licence, they are reprinted in Appendix E.

The rest of this Chapter was originally published in Volume 142, page 224907 of the Journal of Chemical Physics, published by AIP Publishing[2], and is reproduced here without modification. A reprint of the first page is given in Appendix A. The Supporting Information for this Chapter is given in Appendix B.

A multiscale approach to model hydrogen bonding: The case of polyamide

Richard J. Gowers and Paola Carbone
School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester

Abstract

We present a simple multiscale model for polymer chains in which it is possible to selectively remove degrees of freedom. The model integrates all-atom and coarse-grained potentials in a simple and systematic way and allows a fast sampling of the complex conformational energy surface typical of polymers whilst maintaining a realistic description of selected atomistic interactions. In particular, we show that it is possible to simultaneously reproduce the structure of highly directional non-bonded interactions such as hydrogen bonds and efficiently explore the large number of conformations accessible to the polymer chain. We apply the method to a melt of polyamide removing from the model only the degrees of freedom associated to the aliphatic segments and keeping at atomistic resolution the amide groups involved in the formation of the hydrogen bonds. The results show that the multiscale model produces structural properties that are comparable with the fully atomistic model despite being five times faster to simulate.

5.2 Introduction

The use of chemical-specific coarse-grained (CG) models to simulate soft materials such as polymers, surfactants or high viscous liquids has become very popular since the early pioneering attempts [3, 4, 5]. This is because despite the computational power available, the modelling of soft, slow relaxing materials is still a challenge when using all-atom (AA) force fields. Specific coarse-grained models can overcome the problem of the slow sampling of complex energy surfaces, reducing the number of degrees of freedom of the molecular system, and in some cases flattening out the energy landscape.[6, 7] In the past years several approaches to the coarse-graining of soft matter have been proposed. Most of them use a multiscale approach where data

obtained from detailed atomistic simulations are targeted and the CG force field parameters are refined until they match the target properties. Over the years, different target properties for the CG models have been proposed including using structural data,[8] mechanical properties,[9] thermodynamic data such as density values,[10] partition coefficients,[11] inter-particle forces,[12] configurational entropy[13] or potential of mean force.[14] Using CG models, phenomena not accessible before via Molecular Dynamics can now be modelled, for example it is now possible to gain a molecular understanding of complicated self-assembling processes involving polymers, surfactants or ionic liquids.[15, 16, 17, 18]

Despite the obvious advantages in using CG models, there still exist some drawbacks. For example, it is sometimes complicated to understand how to model charged systems such as polyelectrolytes or ionic liquids: the charge is indeed delocalised on several atoms and the use of a point charge located on a single bead might not be the appropriate way to model it.[19] Another limit of a CG approach is the lack of a proper description of the hydrogen bonds (HBs).[20, 21, 22] The atoms involved in these highly directional non-bonded interactions are grouped into a single (or different) CG beads and the interaction is averaged away with other non-bonded interactions. In the case of biopolymers where the hydrogen bond network is responsible for their three dimensional structure, this problem has been solved developing specific CG force field parameters for each aminoacid or DNA base,[23, 24, 25] however such highly targeted approaches cannot be used for other synthetic materials whose 3D structure is not known *a priori*. The formation and disruption of the hydrogen bonding network is also intrinsically related to the system dynamics which can be altered when the model lacks an explicit treatment of it.[26] We have shown that this is indeed the case for structural-based CG models which implicitly take into account the HBs interactions in their effective potentials obtained from radial distribution functions.[26, 27] The dynamics of the CG model in this case correlates with the dynamics of the HBs network. At high temperature the latter has a negligible effect on the former as the HBs network is very weak, but lowering the temperature the importance of the presence of HBs interactions between the amide groups becomes dominant.

One possible solution to this problem is to develop a multi resolved model, where the molecular system is modelled at two different levels of resolution. These two differently resolved models can be linked by the use of replica exchange [28, 29, 30] or be used simultaneously in a single simulation. The latter approach uses atoms and CG beads simultaneously and has been employed to simulate simple fluids,[31] polymers,[1, 32, 33] and selected biological systems.[34, 35, 36, 37, 38] In particular we have shown that it is possible to seamlessly integrate an AA force field with a structural based CG force field in modelling single polymer chains.[1] Recently we

have also developed a simple multiple time step scheme which can take advantage of the intrinsic division between different length and time scales in the dynamics of these kind of models. [39]

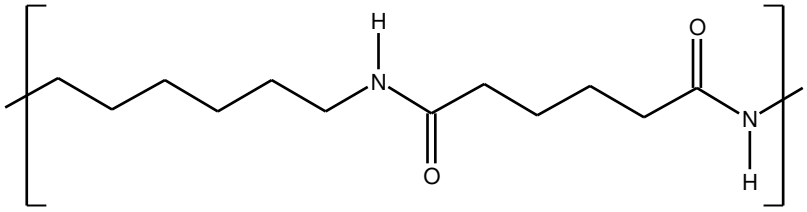
In this contribution we develop an AA-CG dual resolved model for a polymer system, polyamide, which has a well-defined HB network and test its ability to correctly reproduce the thermodynamics and dynamical properties of the HBs. In developing the hybrid model we follow our strategy of mixing a structural based CG model with the AA one used to develop the former but here we also include new features as charge-charge interactions between atoms and two different thermostats to treat separately atoms and beads. This approach to multiscale modelling of soft matter can open the possibility of a selective reduction of the complexity of the chemical system without compromising with the accuracy of the model.

5.3 Computational details

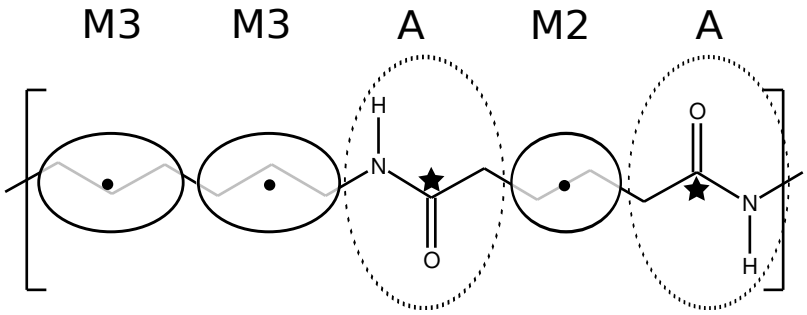
5.3.1 Construction of the hybrid model

Following our previously proposed strategy[1], the hybrid model is built by combining an atomistic model,[40] shown in Figure 5.1a, and a CG model,[41] shown in Figure 5.1c. The CG model is fairly detailed, containing around 3 heavy atoms per bead, yet it has proved to be unable to properly described the directionality of the HBs.[26] The strategy therefore when constructing the hybrid scale model was to reduce the degrees of freedom of the model following the mapping scheme of the CG model, with the exception of bead A, which instead is kept at atomistic resolution. It is hoped that this would allow the model to fully reproduce the behaviour of the atomistic model's hydrogen bonding, whilst benefiting from the increase in computational speed caused by the reduction in detail of the model. The resulting model, shown in Figure 5.1b, features a mixture of both atoms and beads along the polymer chains.

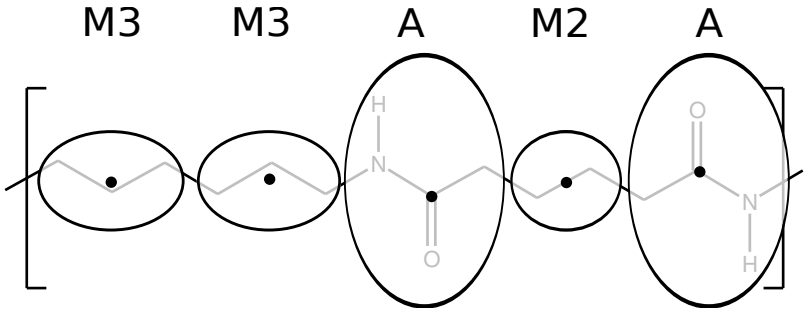
The bead-bead and atom-atom interactions are modelled using the corresponding monoresolved force fields without modifications, while the atom-bead cross term interactions introduced make use of virtual sites. A virtual site (VS) is centred on each sp^2 carbon and are used as a massless, coarse-grained representation of the atoms which would make up an A bead in the CG model. These allow these atoms to interact with the beads in the system using parameters taken directly from the CG forcefield, meaning that no extra force field parameters need to be derived.[1] Each time step, once all bead–VS interactions have been evaluated, the net force on each VS is transferred onto each atom assigned to it on a mass weighted basis, given in Equation 5.1.



(a) Atomistic



(b) Hybrid



(c) Coarse-Grained

Figure 5.1: Comparison of the different mapping schemes for a monomer of polyamide. Solid lines around groups of atoms indicate the mapping onto a single bead whilst dotted lines indicate the boundary of a virtual site. The centre of beads is indicated with a point while the centre of the virtual sites are indicated with a star

$$\mathbf{F}_{atom} = \mathbf{F}_{VS} \times \frac{m_{atom}}{\sum m_{atom}} \tag{5.1}$$

Where $\mathbf{F}$ and m represent force and mass respectively, and the summation of mass goes over all atoms in the VS. In this way linear momentum is conserved throughout the system. The Hamiltonian, $\mathcal{H}$, for this system can be defined as the sum of different energetic contributions, presented in Equation 5.2.

$$\begin{aligned} \mathcal{H} = \quad & \mathcal{K}_{AA} + \mathcal{K}_{CG} + V_{AA} + V_{CG} \\ & + V_{AA-CG(bonded)} + V_{VS-CG(bonded)} \\ & + V_{VS-CG(nonbonded)} \end{aligned} \tag{5.2}$$

Where $\mathcal{K}$ represents the kinetic energy contributions from both the atomistic and coarse-grained sections. Note that virtual sites have neither mass or velocity, meaning that they do not contribute to the kinetic energy in the Hamiltonian. V_{AA} and V_{CG} represent contributions from purely atom-atom and bead-bead interactions, both bonded and nonbonded. $V_{AA-CG(bonded)}$ represents contributions from the direct connectivity between atoms and beads along the polymer chain while $V_{VS-CG(bonded)}$ represents bonded interactions between beads and virtual sites. Finally, $V_{VS-CG(nonbonded)}$ represents contributions from nonbonded interactions between beads and VS.

For the hybrid model presented, parameters for the atomistic interactions came from work by Goudeau et al,[40] whilst parameters for the CG interactions came from later work based on this atomistic force field,[41] and were derived using iterative Boltzmann inversion (IBI).[42] The nonbonded interactions between virtual sites and beads used the parameters developed for the CG force field without modification.

Bonds are necessary to connect atoms and beads along the backbone, in our case between the nitrogen and M3 bead, and the alpha carbon and M2 bead, as shown in Figure 5.1b. These are modelled using a harmonic function; first the probability distribution, $P(\mathbf{r}_{ij})$, is calculated from the atomistic simulation, and then the potential as a function of pairwise distance, $V_{bond}(\mathbf{r}_{ij})$, is determined using a Boltzmann inversion,[42] given in Equation 5.3a.

$$V_{bond}(\mathbf{r}_{ij}) = -k_B T \log\left(P(\mathbf{r}_{ij})\right) \tag{5.3a}$$

$$V_{bond}(\mathbf{r}_{ij}) \approx \frac{1}{2} k_{bond}(\mathbf{r}_{ij} - \mathbf{r}_0)^2 \tag{5.3b}$$

Where k_B is the Boltzmann constant and T, the temperature. This potential is then fitted to a harmonic function, Equation 5.3b, with two parameters, the equilibrium bond length, $\mathbf{r}_0$, and the strength of the bond, k_{bond}.

Angular potentials are also required in the model wherever any bead–atom–atom or bead–bead–atom sequence occurs, there are six such angles in our model. The angular potentials between atoms and beads are also defined according to a Boltzmann inversion of the corresponding distributions obtained from the atomistic model, Equation 5.4. Because these angles typically show more complex behaviour compared to the stiff, symmetric behaviour of the bonds, they are not converted to an analytical form but instead kept as a tabulated potentials. The suitability of using a harmonic model for these bonds, the parameters derived and the details of the angular potentials used and the shape of the probability distributions are all given in Sections S1 and S2 of the Supporting Information.[43]

$$V_{angle}(\theta) = -k_B T \log\left(P(\theta)\right) \tag{5.4}$$

Angular potentials are also in place between beads and virtual sites, these potentials were necessary as the atomistic fragments are interspersed between the CG beads. For example, considering the M3–A–M2 angle in the CG model shown in Figure 5.1c. In the hybrid model the A bead is now at atomistic resolution but in order to preserve the correct rigidity of the chain, an angular potential between the M3 and M2 beads is needed. By using the VS as a point, the angular potential from the CG force field is used without modification. The net force from these angles on the VS is distributed onto the atoms using Equation 5.1.

5.3.2 Simulation details

Molecular dynamics simulations were carried out using a modified version of the IBIsCO simulation software.[44] The system was composed of 24 chains of polyamide 6-6, each with a length of 20 monomer units. Atomistic, hybrid, and coarse-grained scale models were studied. The starting configuration for the atomistic simulation was taken from Goudeau et al,[40] whilst the coarse-grained and hybrid scale configurations were generated from this atomistic configuration, using the mapping schemes detailed in Figure 5.1. For each model, from an initial configuration equilibrated at a temperature of 400 K, five different systems at 50 K temperature intervals between 300 and 500 K were created by adjusting the target temperature of the thermostat in 5 K intervals and running the simulation until no drift in any physical properties was detectable.

All simulations were carried out in the NPT ensemble with a pressure of 101.3 kPa, enforced using a Berendsen barostat with a coupling time of 5 ps. Simulations were carried out in cubic orthogonal boxes, with a side length of approximately 5.5 nm and with periodicity in all three dimensions. A Berendsen thermostat with a coupling time of 0.2 ps was used to maintain the target temperature.[45] The choice of the barostat and thermostat and their coupling times is consistent with what was employed in the atomistic and coarse-grained simulations. [26, 27]

For the hybrid model, the atomistic and coarse-grained sections are treated with separate thermostats, both with identical relaxation times and target temperatures, to ensure that both the atoms and beads were kept at the target temperature. Without the use of separate thermostats it was observed that while the average temperature of the system would be constrained at the target temperature, the atomistic and CG components of the system would not remain at the target temperature but instead diverge around it. Figures showing the difference in temperature when using a single thermostat as well as the results of using separate thermostats are given in Section S5 of the Supporting Information.[43]

An integration time step of 1 fs was used for the hybrid scale and atomistic

models, whilst for the coarse-grained model a time step of 7 fs was used. Once equilibrated, all simulations were run for a minimum of 300 ns with frames being saved every 20 ps and all results presented are an average of this timespan. A cutoff radius of 0.9 nm and 1.2 nm was used for the atoms and beads respectively. Electrostatic interactions in both the atomistic and hybrid scale models were handled using the reaction field method,[46] using the same cutoff radius of 0.9 nm. Verlet neighbour lists were used to manage the pairwise interactions for both the atoms and beads, with a skin radius of 0.2 nm and an update frequency of 10 steps.

5.4 Results

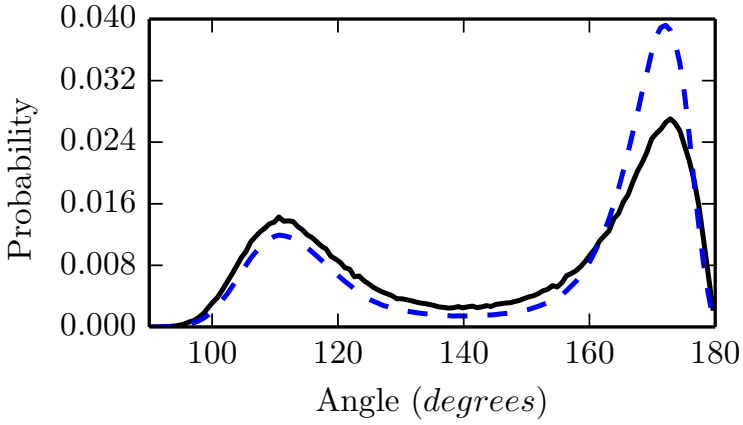
5.4.1 Model validation and static behaviour

To validate the choices made in the construction of the model, the structures generated by the hybrid model at the reference temperature, 400 K, are compared against those made by the atomistic and CG models. To allow comparison between all models, the results of the hybrid and atomistic models have been mapped according to the CG mapping scheme where necessary.

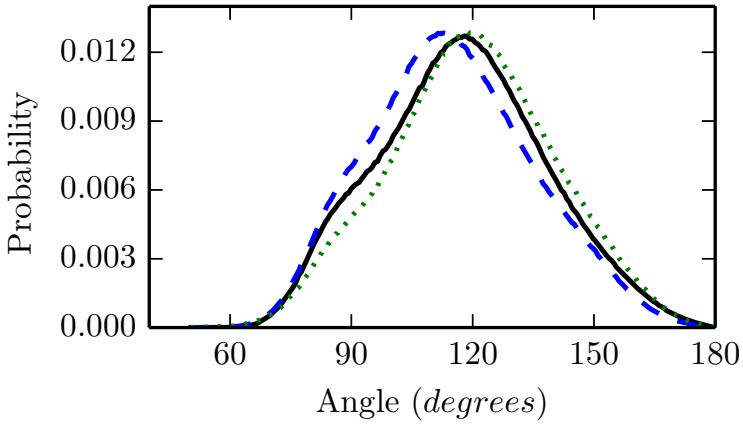
Firstly the probability distributions of the bead-atom distance and bead-VS angles in the hybrid model can be compared against the other two models. Overall these are good, with the worst cases for each shown in Figure 5.2. The performance of these angles could have been improved by refining the angle potentials using iterative Boltzmann inversion but for consistency we decided to keep the original force field parameters without further modifications.

The conformation of a single chain at longer scales can be examined by analysing the intramolecular radial distribution functions, $g(r)$. The worst performing of these, for M2-M2, is presented in Figure 5.3a. At distances larger than 1.0 nm, the $g(r)$ calculated for the hybrid model matches well with the atomistic one, however at shorter distances the two curves differ. The total number of beads at a given distance can be found by integrating the $g(r)$, $n(r) = 4\pi \int_0^r r^2 g(r) dr$. This is plotted in Figure 5.3b and shows that, for this pair of beads, the both the CG and hybrid models predict a more crowded local environment compared to the atomistic one.

The intramolecular $g(r)$ is not included in the optimization procedure used to obtain the CG potentials, indicating that perhaps at this length scale the lack of atomistic details such as torsional flexing limits the correct sample of the local conformations. Larger sized versions of all intramolecular $g(r)$ and $n(r)$ are given in Section S2 of the Supporting Information.[43] Overall, considering that the original force fields parameters have not been changed, the agreement between the two models is satisfactory considering also that the structural properties involving directly



(a) C–M2–C

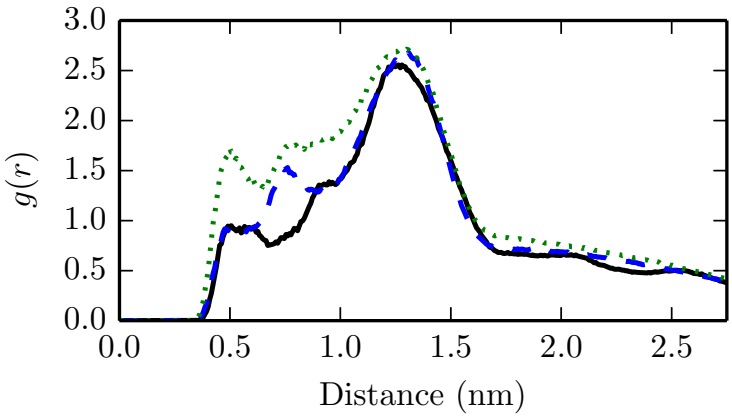


(b) M2–A–M3

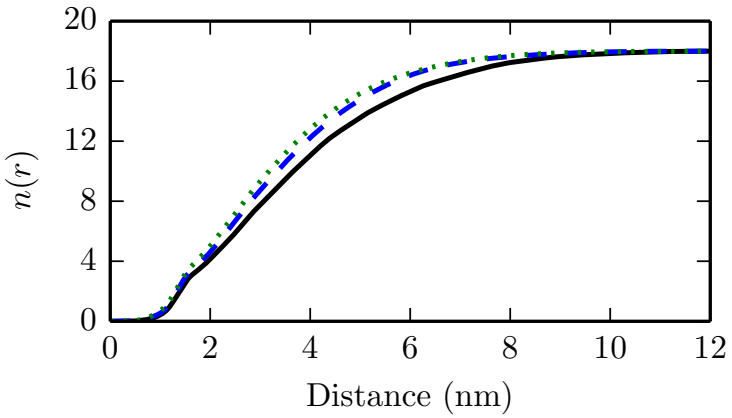
Figure 5.2: Comparison of probability distributions for the two worst performing angles in the hybrid model. The atomistic model is shown by the solid black line, the hybrid model by the blue dashed line and the CG model by the green dotted line. Note that a comparison of the CG model in figure (a) is impossible as it lacks the carbon atoms.

the amide bead are well reproduced, with further examples of this in Section S2 of the Supporting Information.[43] Further refinements of some of the bonded potentials however, could improve the quality of the final model performances.

The average properties over the entire chain length can be compared by looking at the end-to-end distance and radius of gyration of individual polymer chains. The probability distribution of these properties over 250 ns of simulated time are presented in Figure 5.4. From these distributions two important observations can be made; firstly the atomistic model has a jagged distribution, indicating that in the simulated time the atomistic chains sampled only few conformations. On the other hand, the hybrid and CG models show much smoother distributions for both the radius of gyration and end-to-end distance indicating that the chains have had



(a) Intramolecular radial distribution function

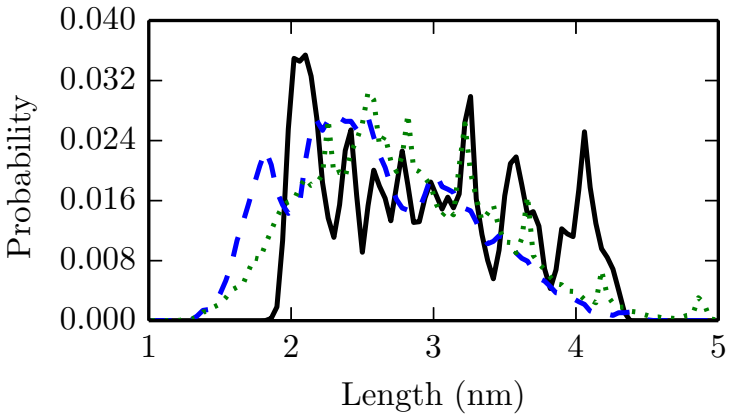


(b) Cumulative radial distribution function

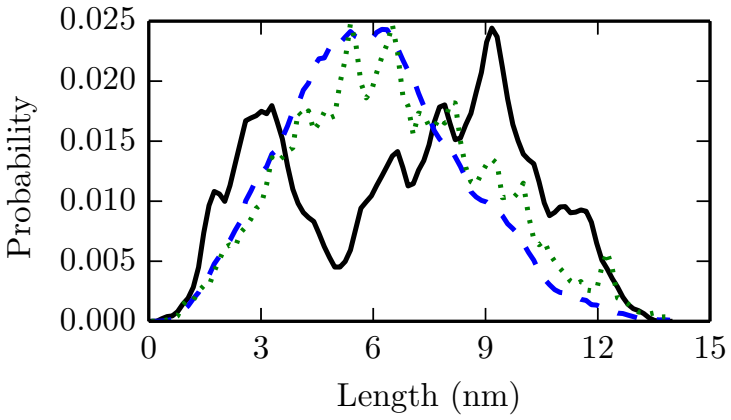
Figure 5.3: Comparison of M2-M2 intramolecular radial distribution functions between different models. Uses same line type as Figure 5.2

sufficient time to explore the conformational space, and that the structures produced are equilibrated and uncorrelated to the starting configuration. In particular the bimodal distribution of the end-to-end distance calculated from the atomistic trajectory seems to indicate that the model is trapped in a local minima. The same distribution calculated at a higher temperature (500K) show the expected mono-peaked distribution, shown in Section S4 of the Supporting Information.[43]

The overestimation of the intramolecular $g(r)$ at low distances which was observed in Figure 5.3a can be explained by considering the distribution of radii of gyration. The increase in probability of chains with a low radius of gyration, indicating a tightly coiled conformation, would correspond to an intramolecular $g(r)$ having higher values at short distances. This might be a consequence of the coarse-grained angular potentials being used, common to both the CG and hybrid models, being slightly too soft.



(a) Radius of gyration



(b) End to end distance

Figure 5.4: Distribution of single chain properties in different models. Line types as Figure 5.2

The persistence length of a polymer chain is a measure of its stiffness, formally defined as the length at which the tangents at two points along a polymer chain become decorrelated. This can be measured by considering the autocorrelation of bond vectors at increasing separation, given in Equation 5.5.

$$C(n) = \langle \cos \theta_{i,i+n} \rangle = \left\langle \frac{\mathbf{a}_i \cdot \mathbf{a}_{i+n}}{|\mathbf{a}_i||\mathbf{a}_{i+n}|} \right\rangle \tag{5.5}$$

Where $C(n)$ measures the correlation of bond i with a bond n bonds further along the backbone, with $\mathbf{a}_i$ representing the bond as a vector. Angular brackets represent an average over all possible start points in all chains. This correlation function is then fitted to an exponential equation, Equation 5.6

$$C(n) \approx \exp\left(-\frac{n\overline{l_B}}{l_P}\right) \tag{5.6}$$

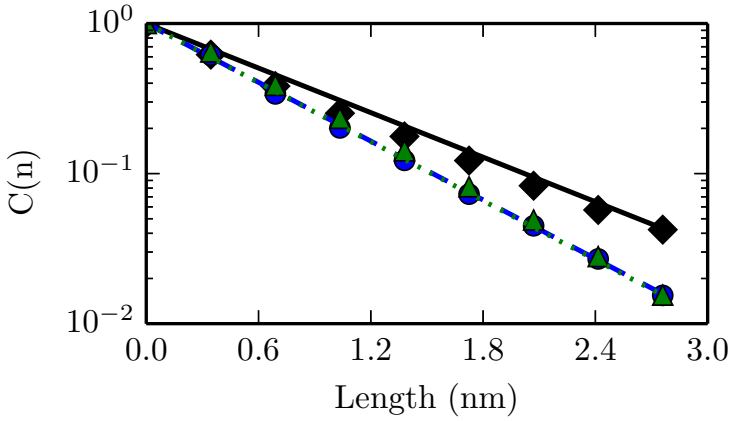


Figure 5.5: Comparison of the bond autocorrelation for different models. The plot uses same labelling as Figure 5.2

where $\overline{l_B}$ represents the mean bond length and l_P the persistence length. This method yield persistence lengths of 0.88, 0.67 and 0.66 nm for the atomistic, hybrid and CG models respectively indicating that the CG and hybrid chains are more flexible than the atomistic model; a comparison of the decay curves for each model and their respective fits are given in Figure 5.5.

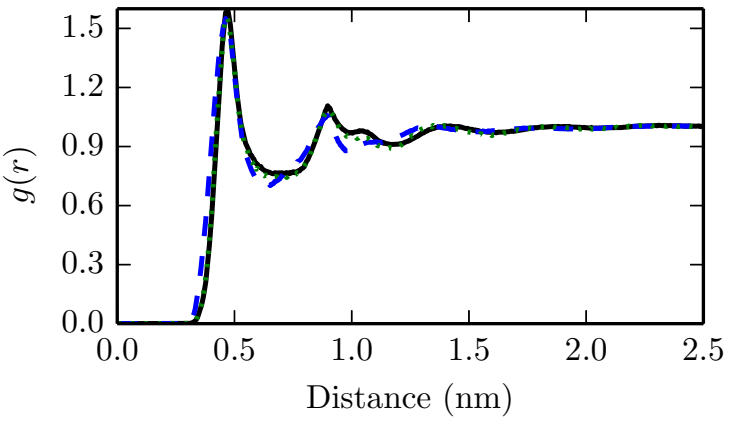
To explore how the nonbonded forcefield for each model is behaving, the packing of chains can be examined by considering the interchain $g(r)$ for different beads in the system. Overall, these show fair agreement, including the A–A $g(r)$, shown in Figure 5.6a, which will be decisive when considering the hydrogen bonding in the system. The worst performing is the M2–M2 $g(r)$ shown in Figure 5.6b.

Finally, as the simulations were run in the NPT ensemble, the density can be compared. The atomistic, hybrid and CG models had a resulting density of 1023, 1097 and 1057 $kg\ m^{-3}$ respectively. As part of the IBI forcefield derivation, the CG model was tuned to match the atomistic model's density,[42] however this step was not repeated when constructing the hybrid model, and so the resulting agreement between the predicted densities is another indication that the forcefields are able to work together.

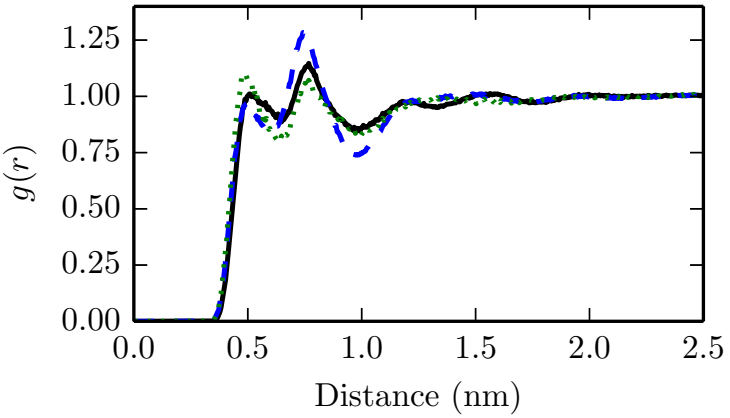
Overall the hybrid model appears to be preserving the same structural features of the original atomistic model, indicating that the procedure followed in constructing of the model is working as intended.

5.4.2 Hydrogen bonding

Now that the hybrid model has been shown to be able to reproduce the bulk structures displayed by the atomistic model with a fair degree of accuracy, the analysis can be focused on the formation of hydrogen bonds (HB) in the hybrid model. These



(a) A–A



(b) M2–M2

Figure 5.6: Intermolecular radial distribution function, $g(r)$, comparison between different scale models at 400 K. Line types as in Figure 5.2

play an important part in the behaviour of polyamide and represent a stringent test for the hybrid model.

In polyamide HBs are formed between the amide groups with the nitrogen atoms donating their hydrogen to an oxygen. In a molecular model the hydrogen bonds are normally defined by a geometric criterion which includes the distance between the hydrogen and the acceptor atom and the value of the donor-hydrogen-acceptor planar angle. Following our previous work,[27] throughout this paper two amide groups are considered hydrogen-bonded if the H$\cdots$O distance is less than 0.3 nm and the N–H$\cdots$O angle is greater than 130 degrees.

The probability distribution of an oxygen forming a hydrogen bond at a given angle and distance is given in Figure 5.7 for both hybrid and atomistic models at 400 K. By integrating across the contour map, it can be calculated that the average number of hydrogen bonds per amide group is 0.6 in the hybrid model compared

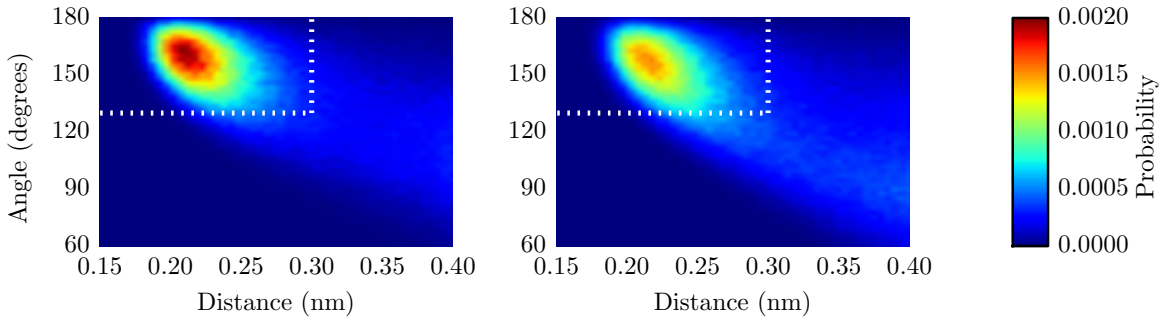


Figure 5.7: Contour map for hydrogen bonding in the atomistic (left) and hybrid (right) systems. The values for the contour map are the probability of a given oxygen forming a bond at a given angle and distance. The boundary for the geometric criteria of a hydrogen bond used in this work is indicated using a white dashed line.

with 0.7 obtained for the atomistic one. Although the average number of HBs is in fair agreement between the two models, from Figure 5.7 it can be observed that the peak angle and distance at which HBs form is more sharply defined in the atomistic model.

In order to quantify this discrepancy between the two models in energetic terms, the enthalpy and entropy of hydrogen bond formation is calculated using the model proposed by Schroeder et al.[47] This treats the breaking of hydrogen bonds as a reversible process with an equilibrium constant K. The value of this constant can be estimated by considering the fraction of amide groups which are hydrogen bonded, X, as shown in Equation 5.7. This approach assumes that all donators and acceptors only form a single hydrogen bond each; with analysis of the results showing that this assumption is reasonable.

$$(NH\ldots O)_{bonded} \overset{K}{\rightleftharpoons} (NH)_{free} + (O)_{free} \tag{5.7a}$$

$$K = \frac{\left[(NH)_{free}\right]\left[(O)_{free}\right]}{\left[(NH\ldots O)_{bonded}\right]} \approx \frac{(1-X)^2}{X} \tag{5.7b}$$

The natural logarithm of K can be related to the change in enthalpy, ΔH, and entropy, ΔS, through Equation 5.8, where R is the gas constant and T the temperature.

$$\log K = -\frac{\Delta H}{RT} + \frac{\Delta S}{R} \tag{5.8}$$

By considering K over a range of temperatures, and assuming that ΔS remains constant over this range, the values for ΔH and ΔS can be found through fitting with a straight line the values of $\log K$ against temperature, as shown in Figure 5.8.

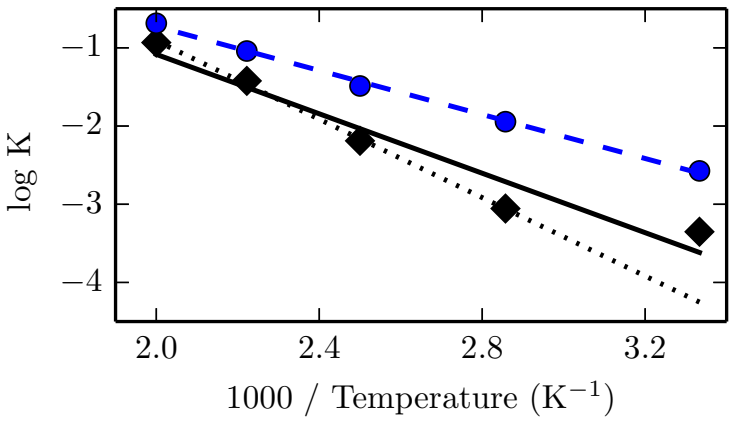


Figure 5.8: Change in equilibrium constant for hydrogen bond breakage with varying temperature for both atomistic (black diamonds) and hybrid (blue circles) models. The lines represent the best fitting for the two sets of data, for the atomistic data two lines have been plotted, the solid line including all points, and the dotted line excluding the data for T=300 K

This yields an enthalpy change of -15.8±0.27 and -11.7±0.04 kJ mol^{-1} and an entropy change of +22.6±0.71 and +17.4±0.11 J mol^{-1} K^{-1} for the formation of a hydrogen bond in the atomistic and hybrid models respectively. It can be seen however that the point in the atomistic data corresponding to the lowest temperature simulated, 300 K, does not fit well with the other 4 points, possibly due to poor equilibration at this low temperature. By considering a line of best fit from only the first 4 data points, a much better fit is produced, yielding however to a worse agreement with the hybrid model results. The new values of ΔH and a ΔS are now -20.8±0.01 kJ mol^{-1} and +34.1±0.03 J mol^{-1} K^{-1} respectively. The hybrid model therefore now underestimates the enthalpy by around 40%, although both enthalpy values (obtained from the atomistic and hybrid model) lay within the experimental data obtained for low molecular weight amides which range between -14.5 to -27 kJ mol^{-1}.[48]

The difference in free energy values obtained for the atomistic and hybrid models can be ascribed to three different effects. Firstly we should notice that in the hybrid model the atomistic amide groups are surrounded by bulky CG beads which can sterically hinder the formation of some HBs when the chains (or fragment of the same chains) are aligned. Secondly the mismatch between the local conformation of the models observed in the angular distribution of Figure 5.2a and intra-chain $g(r)$ of Figure 5.3a could indicate that the amide groups along the chain are less accessible in the hybrid model than in the atomistic one. Considering that we use a geometric criterion to identify the number of HBs in the system, this small conformational difference could be the reason of apparently weaker HB observed in

the hybrid model. The agreement between the local and global conformation at temperatures different than $400K$ (the CG force field optimization temperature) is however good (see Section S6 Supplementary Information)[43] indicating that the the hybrid model can be used in a range of temperatures.

Finally another reason for the lower enthalpy value could be that the hybrid model is characterised by different dynamics than the atomistic one. It is indeed known[26, 49] that the coarse-graining of an atomistic model always induces a speed up in its dynamics due to both the reduced number of degrees of freedom and the smoother and softer CG interaction potentials.[7, 50, 51] We have shown that the local dynamics of a multiresolved polymer chain is indeed affected (i.e. the chain moves faster) at short time scales by the presence of fast-moving beads bonded to slower atoms.[39] It is then possible that such speed up weakens the HB network and a detailed analysis of its dynamics is reported below. It is important to notice however that although using Equation 5.8 the mismatch between the two models seems remarkable, looking at the actual difference between the predicted number of HBs, the disagreement looks less problematic. Indeed the hydrid model underestimates the number of total HBs (intra- and inter- chain) by only 15%, 13%, 10% and 8% for the simulations performed at 350 K, 400 K, 450 K and 500 K respectively.

5.4.3 Hydrogen-bond dynamics

The dynamic behaviour of the hydrogen bonds can be analysed by considering their individual average lifetime . The time autocorrelation function used to measure hydrogen bonds is given in Equation 5.9.[52]

$$C_x(t) = \left\langle \frac{\sum h_{ij}(t_0)h_{ij}(t_0+t)}{\sum h_{ij}(t_0)^2} \right\rangle \tag{5.9}$$

where h_{ij} is a binary measure of whether a pairing ij meets the geometric hydrogen bond criteria; $h_{ij} = 1$, or not, $h_{ij} = 0$. The summation is performed over all possible pairings, ij, in our case all oxygen atoms are considered possible acceptors and all nitrogen atoms are possible donors. Angular brackets represent an average over many different starting times in the trajectory. The subscript x refers to the two different definitions for measuring h_{ij} at future points in time, continuous or intermittent, detailed below. The code used to calculate this has been included as part of the MDAnalysis Python package.[53]

In the definition for continuous lifetime, once a particular hydrogen bond is broken, it is then always considered broken even if the bond subsequently reforms. This definition therefore measures the average time a pair remains intact and yields the average hydrogen bond lifetime. Intermittent hydrogen bond lifetime allows bonds which were considered broken to be reformed and counted again at a future

Table 5.1: Hydrogen bonding lifetime for atomistic (AA) and hybrid (HY) models over a range of temperatures

T (K)	τ_C (ps)		Ratio
	AA	HY	
500	0.221	0.154	0.70
450	0.548	0.304	0.55
400	0.861	0.421	0.49
350	1.28	0.541	0.42

point in time, therefore measuring the time that a particular hydrogen bonded pair remains in the same vicinity, yielding information on the structural relaxation time of the polymer. For both definitions outlined above, the relevant lifetime, τ_x, is defined as

$$\tau_x = \int_0^\infty \left(C_x\left(t\right) - \left\langle C_x\left(t = \infty\right)\right\rangle\right) dt \tag{5.10}$$

To measure the continuous hydrogen bond lifetime, τ_C, simulations were run at different temperatures and the trajectories were saved at a resolution of 0.05 ps. The results of the autocorrelation functions were fitted using the sum of two exponential decays, shown in Equation 5.11. This could then be integrated analytically to find the hydrogen bond lifetimes, τ_C, over a range of different temperatures for both models, the values obtained are presented in Table 5.1. The hydrogen bond lifetime in the hybrid model was approximately half of that of the atomistic model at the reference temperature, 400 K.

$$C_C(t) \approx A_1 \exp\left(\frac{-t}{\tau_1}\right) + (1 - A_1) \exp\left(\frac{-t}{\tau_2}\right) \tag{5.11}$$

To compare the mechanism by which hydrogen bonds break in the atomistic and hybrid scale models, the shape of C_C can be compared. By applying a linear transformation in the time axis of the hybrid model results using the ratio of the hydrogen bond lifetimes, the two decay curves can be directly compared. This is shown in Figure 5.9, with the two decay curves overlapping completely, indicating that the breakage of hydrogen bonds follows the same mechanism but is accelerated by a single constant factor in the hybrid model. This ratio of hydrogen bond lifetimes, third column of Table 5.1, decreases with increasing temperature which matches observations on the relative dynamics of the atomistic and CG dynamics.[26] and represents the speedup of the hydrid model compared to the atomistic one. To obtain the free energy difference between the two models, the data can be plotted against the values of the corresponding temperatures and be fitted to the Arrehnius equation, $\exp(\Delta E/k_B T)$,[54] where ΔE represents the difference in barrier heights

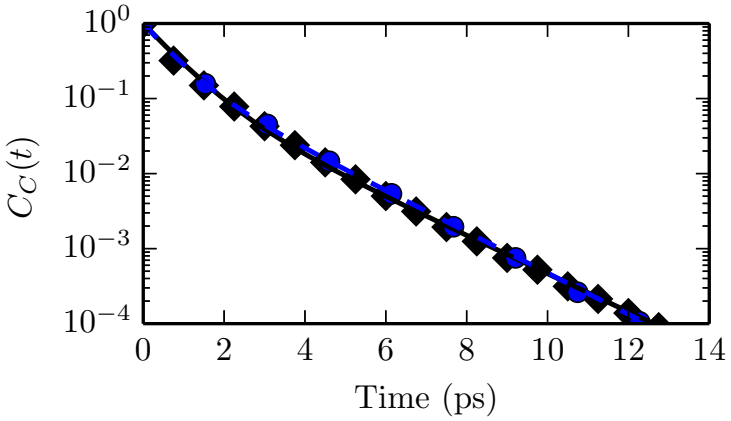


Figure 5.9: Comparison of the continuous hydrogen bonding time autocorrelation function. The results for the hybrid system have been rescaled in the time axis as described in the text. Black diamonds indicate the atomistic results and blue circles indicate the hybrid results. The black solid line indicate the fit used for atomistic results, while the fit for hybrid results is given as a blue dashed line.

between the atomistic and hybrid model which in this case is 0.003 k_BT .

The intermittent hydrogen bond lifetime, τ_I, for the hybrid system was calculated by numerically integrating Equation 5.10 and yielded a value of 565 ps. This was not possible for the atomistic system as the value of C_I showed a plateau for times above 300 ns, indicating that the the fuction had not fully decorrelated. Instead, the atomistic τ_I was estimated assuming that the difference between the C_I data for the two models could again be described as a linear transformation in the time axis by a factor a. The value for a was then estimated by minimizing the resulting residual function, E, between thetwo functions, as defined in Equation 5.12.

$$E = \sum_{t=0}^{t=300\ ns} \left(C_{I\ atom}(t) - C_{I\ hybrid}(at)\right)^2 \tag{5.12}$$

A value of 27.3 was found to provide the closest agreement between the two sets of data, the agreement between the two lines is plotted in Figure 5.10. This therefore gives an estimate for the τ_I of the atomistic system of 15.4 ns. The value of this new scaling parameter a is independent on the time range used for the fitting, this is shown in Section S7 of the Supporting Information.[43]

To validate the time ratio, a, obtained using Equation 5.12, C_I was numerically integrated until it reaches the value of 0.05 (Equation 5.13). This yielded times of 3230 and 116 ps for the atomistic and hybrid models respectively, giving a ratio of 27.8, which supports the value found above.

$$\tau\prime = \int_{C_I=1.0}^{C_I=0.05} C_I(t)dt \tag{5.13}$$

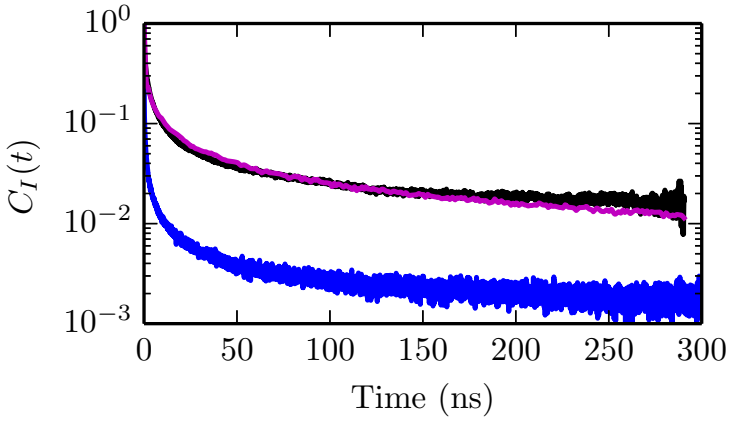


Figure 5.10: Intermittent hydrogen bond correlation functions. The black and blue lines represent the atomistic and hybrid model results, while the magenta line shows the hybrid results after the time transformation.

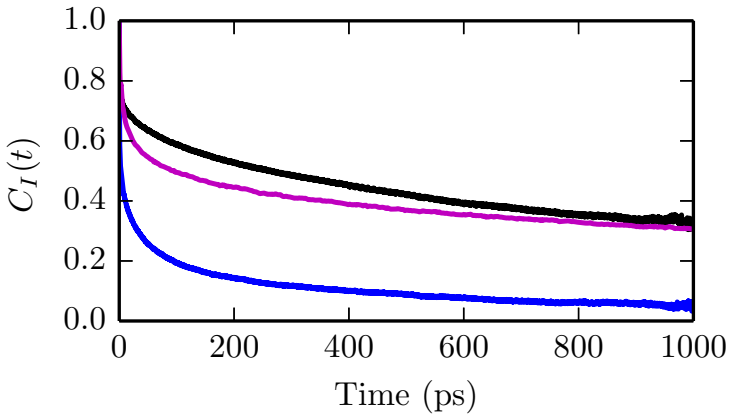


Figure 5.11: Detail of the first nanosecond of intermittent hydrogen bond correlation functions. Uses the same colour scheme as Figure 5.10

Although using the scale parameter a brings the two sets of results close to each other, small differences are visible. Considering the results for the first nanosecond, which are again calculated at 0.05 ps resolution and shown in Figure 5.11, it is apparent that the shifted hybrid results are not perfectly matching the atomistic results. This discrepancy between the two lines indicates that the difference in intermittent lifetime of hydrogen bonds between the two models cannot entirely be described using a single scalar factor, and that this ratio represents the average difference in time scale between the two models. This result is reasonable as the short time scale the dynamics is dominated by the local conformational changes, where the atoms are involved, while at longer time scale the dynamics is dominated by the bead motions.

Considering that previous work on CG polyamide has estimated the ratio in

Table 5.2: Comparison of the computational cost of the models examined.

Model	Number of pairs	Normalised execution time
Atomistic	4,087,000	1.0
Hybrid	863,000	0.20
CG	188,000	0.057

structure relaxation times at around 150,[26] with an observed ratio of around 30, the hybrid model's dynamics seem to be situated somewhere between the atomistic and CG model one.

5.5 Computational Performance

In molecular dynamics simulations, approximately 90% of the computational work to be done is the calculation of nonbonded interactions between particles.[46] Measuring the total number of pairs that need to be evaluated therefore, is a good measure of the computational cost of a model, independent of the algorithms used in the implementation. These results are shown in Table 5.2 along with the average execution time for the calculation of a single time step, normalised against the atomistic model.

From these results it can be seen that the hybrid model has substantially less pairwise interactions to calculate, resulting in a model which is roughly five times quicker to run. Also of note is that the execution time scales approximately inversely to the number of pairs. This indicates that whilst there are extra calculations to be performed in using the hybrid model, such as calculation the positions of the virtual sites and distribution of forces to the atoms, this does not impose a significant difference in the speed of the model.

It is important to remember that the execution times presented are for a single time step, but that CG models can integrate larger amounts of time with each step. The work in this paper used an integration time step of 1 fs for both the CG and atomistic parts of the system, however work has been done which allows the atomistic and CG portions of a hybrid scale model to be treated using its "native" time step.[39] If implemented this would again increase the speed of the hybrid model and help bridge the gap towards the CG model's speed.

5.6 Summary and Conclusions

We have developed a novel multiscale model for macromolecular systems characterized by a strong and interconnected hydrogen bond network. The model was obtained mixing atoms and coarse-grained (CG) beads and it combines a quick

exploration of the conformational space typical of coarse-grained models with an atomistic detail description of the hydrogen bond interactions. If both atomistic and structural-based coarse-grained potentials are already available, the construction of the new model requires little additional parametrisation and is easy to set up.

The procedure was tested on a bulk of polyamide 6-6 which is known to have a well-defined hydrogen bond network that dominates most of its structural and dynamical properties. Overall the model has proved to be able to reproduce both thermodynamic and static properties of the bulk. Since the CG potential was optimized to reproduce the atomistic pressure, the dual-resolved model when simulated at constant pressure reproduces the correct density. The predicted structure analysed through various target pair distributions is also in reasonably good agreement with that of the mono-resolved models

We have shown that the hybrid model can form hydrogen bonds and that their total number was in good agreement with the corresponding atomistic data with the difference between the models becoming smaller upon increasing temperature. However, this disagreement between the two models was amplified when the thermodynamics of the hydrogen bond is examined. The enthalpic and entropic components of the HB free energy are quite different although the data from both models fell within the wide experimental range. The numerical mismatch was then attributed to a difference noticed in the intra-chain $g(r)$ calculated between the beads neighbouring the atomistic amide fragments and probably due to steric effects. The different dynamics of the two models could be also responsible for the mismatch. A further refinement of some components of the force field could lead to a better agreement at least from a structural point of view.

The dynamics of the hybrid scale model was shown to have complicated behaviour. At the small length scale of the individual hydrogen bond lifetime the atomistic and hybrid models are characterized by the same dynamics although shifted by a factor of two with the former slower than the latter. At larger length scales this factor was around 30. These results show that the hybrid model behaves as an atomistic model at short time scale where the atomistic interactions are dominant, and as a mesoscopic model at larger time scale where the impact of having fast and larger beads becomes evident.

Despite the fact that the mixed force field can properly model in full detail the formation of hydrogen bonds, the model is still much faster than the atomistic one. This is not only because there are, with the present mapping scheme, five times fewer interaction pairs to calculate against the atomistic model, but also because within the same simulation time the conformational space is sampled more efficiently.

Our work opens the possibility to be able to model large molecular systems

with relatively small effort and at the same time capture with atomistic details interactions which determine their behaviour.

Acknowledgements

This work was funded by BBSRC grant BB/J014478/1. The first author would like to thank Flor Siperstein and Carlos Avendaño from the University of Manchester for useful discussions. The authors would like to acknowledge the assistance given by IT Services and the use of the Computational Shared Facility at The University of Manchester.

Bibliography

[1] N. Di Pasquale, D. Marchisio, and P. Carbone, "Mixing atoms and coarse-grained beads in modelling polymer melts," *The Journal of Chemical Physics*, vol. 137, no. 16, p. 164111, 2012.

[2] R. J. Gowers and P. Carbone, "A multiscale approach to model hydrogen bonding: The case of polyamide," *The Journal of Chemical Physics*, vol. 142, no. 22, p. 224907, 2015.

[3] J. Baschnagel, K. Binder, P. Doruker, A. Gusev, O. Hahn, K. Kremer, W. Mattice, F. Müller-Plathe, M. Murat, W. Paul, S. Santos, U. Suter, and V. Tries, "Bridging the gap between atomistic and coarse-grained models of polymers: Status and perspectives," in *Advances in Polymer Science: Viscoelasticity Atomistic Models Statistical Chemistry*, vol. 152 of *ADVANCES IN POLYMER SCIENCE*, pp. 41–156, Springer, 2000.

[4] W. Tschöp, K. Kremer, J. Batoulis, T. Bürger, and O. Hahn, "Simulation of polymer melts. i. coarse-graining procedure for polycarbonates," *Acta Polymerica*, vol. 49, no. 2-3, pp. 61–74, 1998.

[5] W. Tschöp, K. Kremer, O. Hahn, J. Batoulis, and T. Bürger, "Simulation of polymer melts. ii. from coarse-grained models back to atomistic description," *Acta Polymerica*, vol. 49, no. 2-3, pp. 75–79, 1998.

[6] P. Carbone and C. Avendaño, "Coarse-grained methods for polymeric materials: enthalpy- and entropy-driven models," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 4, no. 1, pp. 62–70, 2014.

[7] H. A. Karimi-Varzaneh, N. F. A. van der Vegt, F. Müller-Plathe, and P. Carbone, "How good are coarse-grained polymer models? A comparison for atactic polystyrene," *ChemPhysChem*, vol. 13, no. 15, pp. 3428–3439, 2012.

[8] F. Müller-Plathe, "Coarse-graining in polymer simulation: From the atomistic to the mesoscopic scale and back," *ChemPhysChem*, vol. 3, no. 9, pp. 754–769, 2002.

[9] S. Cranford and M. J. Buehler, "Twisted and coiled ultralong multilayer graphene ribbons," *Modelling and Simulation in Materials Science and Engineering*, vol. 19, no. 5, p. 054003, 2011.

[10] W. Shinoda, R. Devane, and M. Klein, "Multi-property fitting and parameterization of a coarse grained model for aqueous surfactants," *Molecular Simulation*, vol. 33, no. 1-2, pp. 27–36, 2007.

[11] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, "The MARTINI force field: Coarse grained model for biomolecular simulations," *The Journal of Physical Chemistry B*, vol. 111, no. 27, pp. 7812–7824, 2007.

[12] W. G. Noid, J. Chu, G. S. Ayton, V. Krishna, S. Izvekov, G. A. Voth, A. Das, and H. C. Andersen, "The multiscale coarse-graining method. i. a rigorous bridge between atomistic and coarse-grained models," *The Journal of Chemical Physics*, vol. 128, no. 24, p. 244114, 2008.

[13] A. Chaimovich and M. S. Shell, "Coarse-graining errors and numerical optimization using a relative entropy framework," *The Journal of Chemical Physics*, vol. 134, no. 9, p. 094112, 2011.

[14] E. Brini, V. Marcon, and N. F. van der Vegt, "Conditional reversible work method for molecular coarse graining applications," *Physical Chemistry Chemical Physics*, vol. 13, no. 22, pp. 10468–10474, 2011.

[15] M. Velinova, D. Sengupta, A. V. Tadjer, and S. J. Marrink, "Sphere-to-rod transitions of nonionic surfactant micelles in aqueous solution modeled by molecular dynamics simulations," *Langmuir*, vol. 27, no. 23, pp. 14071–14077, 2011.

[16] M. L. Klein and W. Shinoda, "Large-scale molecular dynamics simulations of self-assembling systems," *Science*, vol. 321, no. 5890, pp. 798–800, 2008.

[17] H. A. Karimi-Varzaneh, F. Müller-Plathe, S. Balasubramanian, and P. Carbone, "Studying long-time dynamics of imidazolium-based ionic liquids with a systematically coarse-grained model," *Physical Chemistry Chemical Physics*, vol. 12, pp. 4714–4724, 2010.

[18] C. Luo and J. U. Sommer, "Growth pathway and precursor states in single lamellar crystallization: Md simulations," *Macromolecules*, vol. 44, no. 6, pp. 1523–1529, 2011.

[19] P. A. Golubkov, J. C. Wu, and P. Ren, "A transferable coarse-grained model for hydrogen-bonding liquids," *Physical Chemistry Chemical Physics*, vol. 10, pp. 2050–2057, 2008.

[20] A. V. Savin, I. P. Kikot, M. A. Mazo, and A. V. Onufriev, "Two-phase stretching of molecular chains," *Proceedings of the National Academy of Sciences*, vol. 110, no. 8, pp. 2816–2821, 2013.

[21] T. Shen and S. Gnanakaran, "The stability of cellulose: a statistical perspective from a coarse-grained model of hydrogen-bond networks.," *Biophysical Journal*, vol. 96, no. 8, pp. 3032–3040, 2009.

[22] A. Milani, M. Casalegno, C. Castiglioni, and G. Raos, "Coarse-grained simulations of model polymer nanofibres," *Macromolecular Theory and Simulations*, vol. 20, no. 5, pp. 305–319, 2011.

[23] A. Voegler Smith and C. K. Hall, "*alpha*-helix formation: Discontinuous molecular dynamics on an intermediate-resolution protein model," *Proteins: Structure, Function, and Bioinformatics*, vol. 44, no. 3, pp. 344–360, 2001.

[24] S. Auer and D. Kashchiev, "Phase diagram of α-helical and β-sheet forming peptides," *Physical Review Letters*, vol. 104, p. 168105, 2010.

[25] D. De Sancho and A. Rey, "Evaluation of coarse grained models for hydrogen bonds in proteins," *Journal of Computational Chemistry*, vol. 28, no. 7, pp. 1187–1199, 2007.

[26] H. A. Karimi-Varzaneh, P. Carbone, and F. Müller-Plathe, "Fast dynamics in coarse-grained polymer models: The effect of the hydrogen bonds," *The Journal of Chemical Physics*, vol. 129, no. 15, p. 154904, 2008.

[27] H. A. Karimi-Varzaneh, P. Carbone, and F. Müller-Plathe, "Hydrogen bonding and dynamic crossover in polyamide-66: A molecular dynamics simulation study," *Macromolecules*, vol. 41, no. 19, pp. 7211–7218, 2008.

[28] P. Liu and G. A. Voth, "Smart resolution replica exchange: An efficient algorithm for exploring complex energy landscapes," *The Journal of Chemical Physics*, vol. 126, no. 4, p. 045106, 2007.

[29] P. Liu, Q. Shi, E. Lyman, and G. A. Voth, "Reconstructing atomistic detail for coarse-grained models with resolution exchange," *The Journal of Chemical Physics*, vol. 129, no. 11, p. 114103, 2008.

[30] J. O. B. Tempkin, B. Qi, M. G. Saunders, B. Roux, A. R. Dinner, and J. Weare, "Using multiscale preconditioning to accelerate the convergence of iterative molecular calculations," *The Journal of Chemical Physics*, vol. 140, no. 18, p. 184114, 2014.

[31] M. Praprotnik, L. Delle Site, and K. Kremer, "Adaptive resolution molecular-dynamics simulation: Changing the degrees of freedom on the fly," *The Journal of Chemical Physics*, vol. 123, no. 22, p. 224106, 2005.

[32] L. Delle Site, S. Leon, and K. Kremer, "BPA-PC on a Ni(111) surface: The interplay between adsorption energy and conformational entropy for different chain-end modifications," *Journal of the American Chemical Society*, vol. 126, no. 9, pp. 2944–2955, 2004.

[33] B. Ensing, S. O. Nielsen, P. B. Moore, M. L. Klein, and M. Parrinello, "Energy conservation in adaptive hybrid atomistic/coarse-grain molecular dynamics," *Journal of Chemical Theory and Computation*, vol. 3, no. 3, pp. 1100–1105, 2007.

[34] S. Izvekov and G. A. Voth, "Mixed resolution modeling of interactions in condensed-phase systems," *Journal of Chemical Theory and Computation*, vol. 5, no. 12, pp. 3232–3244, 2009.

[35] M. R. Machado, P. D. Dans, and S. Pantano, "A hybrid all-atom/coarse grain model for multiscale simulations of dna," *Physical Chemistry Chemical Physics*, vol. 13, no. 40, pp. 18134–18144, 2011.

[36] J. Michel, M. Orsi, and J. W. Essex, "Prediction of partition coefficients by multiscale hybrid atomic-level/coarse-grain simulations," *The Journal of Physical Chemistry B*, vol. 112, no. 3, pp. 657–660, 2008.

[37] A. J. Rzepiela, M. Louhivuori, C. Peter, and S. J. Marrink, "Hybrid simulations: combining atomistic and coarse-grained force fields using virtual sites," *Physical Chemistry Chemical Physics*, vol. 13, no. 22, pp. 10437–10448, 2011.

[38] Q. Shi, S. Izvekov, and G. A. Voth, "Mixed atomistic and coarse-grained molecular dynamics: simulation of a membrane-bound ion channel," *The Journal of Physical Chemistry B*, vol. 110, no. 31, pp. 15045–15048, 2006.

[39] N. Di Pasquale, R. J. Gowers, and P. Carbone, "A multiple time step scheme for multiresolved models of macromolecules," *Journal of Computational Chemistry*, vol. 35, no. 16, pp. 1199–1207, 2014.

[40] S. Goudeau, M. Charlot, C. Vergelati, and F. Müller-Plathe, "Atomistic simulation of the water influence on the local structure of polyamide 6,6," *Macromolecules*, vol. 37, no. 21, pp. 8072–8081, 2004.

[41] P. Carbone, H. A. Karimi-Varzaneh, X. Chen, and F. Müller-Plathe, "Transferability of coarse-grained force fields: The polymer case," *The Journal of Chemical Physics*, vol. 128, no. 6, p. 064094, 2008.

[42] D. Reith, M. Pütz, and F. Müller-Plathe, "Deriving effective mesoscale potentials from atomistic simulations," *Journal of Computational Chemistry*, vol. 24, no. 13, pp. 1624–1636, 2003.

[43] R. J. Gowers and P. Carbone, "Supporting information for: A multiscale approach to model hydrogen bonds: The case of polyamide." See supplemental material at URL for further details on the forcefield parameters, structural results, thermostat details and fitting of intermittent hydrogen bond lifetimes.

[44] H. A. Karimi-Varzaneh, H.-J. Qian, X. Chen, P. Carbone, and F. Müller-Plathe, "IBIsCO: A molecular dynamics simulation package for coarse-grained simulation," *Journal of Computational Chemistry*, vol. 32, no. 7, pp. 1475–1487, 2011.

[45] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," *The Journal of Chemical Physics*, vol. 81, no. 8, pp. 3684–3690, 1984.

[46] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*. New York, NY, USA: Clarendon Press, 1989.

[47] L. R. Schroeder and S. L. Cooper, "Hydrogen bonding in polyamides," *Journal of Applied Physics*, vol. 47, no. 10, pp. 4310–4317, 1976.

[48] G. C. Pimentel and A. L. McClellan, *The Hydrogen Bond*. San Francisco: Freeman and Co, 1960.

[49] D. Fritz, K. Koschke, V. A. Harmandaris, N. F. A. van der Vegt, and K. Kremer, "Multiscale modeling of soft matter: scaling of dynamics," *Physical Chemistry Chemical Physics*, vol. 13, pp. 10412–10420, 2011.

[50] S. Izvekov and G. A. Voth, "Modeling real dynamics in the coarse-grained representation of condensed phase systems," *The Journal of Chemical Physics*, vol. 125, no. 15, p. 151101, 2006.

[51] C. Hijón, P. Español, E. Vanden-Eijnden, and R. Delgado-Buscalioni, "Mori-zwanzig formalism as a practical computational tool," *Faraday Discuss.*, vol. 144, pp. 301–322, 2010.

[52] D. Rapaport, "Hydrogen bonds in water," *Molecular Physics*, vol. 50, no. 5, pp. 1151–1162, 1983.

[53] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, "MDAnalysis: A toolkit for the analysis of molecular dynamics simulations," *Journal of Computational Chemistry*, vol. 32, no. 10, pp. 2319–2327, 2011.

[54] P. K. Depa and J. K. Maranas, "Dynamic evolution in coarse-grained molecular dynamics simulations of polyethylene melts," *The Journal of Chemical Physics*, vol. 126, no. 5, p. 054903, 2007.

Chapter 6

Multiple Time Step schemes

6.1 Preface

After the successful creation of a methodology for dual scale simulations[1], allowing a mixture of atoms and beads within the same molecular model, attention was then directed towards how to maximise the benefits of using this approach. One of the benefits of using a CG model is that a larger integration time step can be used when advancing the system in time. Ultimately this leads to a faster sampling of conformational space and dynamic properties of materials, as for each simulated step, more phase space is traversed. However in dual scale models the smaller atomistic integration time step has to be used, leading to an inefficient use of computation as the CG part is calculated more often than is necessary.

The work is this Chapter aims to allow the CG parts of the system to operate at their larger native time step with the atomistic part proceeding at their smaller time step. This is done by allowing the CG nonbonded interactions to be evaluated only every m steps, while the atomistic interactions are calculated every step.

The simulations in this Chapter were again performed with in house code detailed in Chapter 7, with the necessary modifications detailed in Section 7.4. The overall computational gain in the systems studied was modest, however for other systems with proportionally more coarse particles the gain would be much larger.

This rest of this Chapter was originally published in Volume 35, Issue 16, page 1199, of the Journal of Computational Chemistry, published by Wiley Periodicals, and is reproduced here without modification. A reprint of the first page along with the cover image for this Issue, which was taken from this submission, are given in Appendix A.

Contribution details

The idea and and supporting theory of applying an MTS scheme to a hybrid model was originally developed by Nicodemo Di Pasquale and Paola Carbone. The author contributed to this work through implementing the algorithm, designing and performing analysis of results, as well as discussions and helping in the preparation of the manuscript.

A Multiple Time Step Scheme for Multiresolved Models of Macromolecules

Nicodemo Di Pasquale[1], Richard J. Gowers[2] and Paola Carbone[2]
1 - Dipartimento di Scienza dei Materiali e Ingegneria Chimica, Politecnico di Torino, Torino, Italy
2 - School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester

Abstract

In hybrid particle models where coarse-grained beads and atoms are used simultaneously, two clearly separate time scales are mixed. If such models are used in molecular dynamics simulations, a multiple time step (MTS) scheme can therefore be used. In this manuscript, we propose a simple MTS algorithm which approximates for a specific number of integration steps the slow coarse-grained beadbead interactions with a Taylor series approximation while the atomatom ones are integrated every time step. The procedure is applied to a previously developed hybrid model of a melt of atactic polystyrene (Di Pasquale, Marchisio, and Carbone, J. Chem. Phys. 2012, 137, 164111). The results show that structure, local dynamics, and free diffusion of the model are not altered by the application of the integration scheme which can confidently be used to simulate multiresolved models of polymer melts.

6.2 Introduction

In recent years, the development of novel computational techniques which combine different levels of resolution of molecular models has become an exciting new field in computational chemistry and physics[2, 3]. These multiresolved models could in fact be used to overcome one of the most challenging problems in molecular dynamics (MD) simulations namely the difficulty in sampling large conformational spaces with models which retain atomistic details. Commonly the problem of the reduced length scale of MD simulations is solved by resorting to coarse-grained (CG) models where

the degrees of freedom of the molecular system are reduced and the energy surface to sample is consequently flattened out.

Although these techniques[4, 5] have proved to be extremely powerful in predicting structural[6, 7], thermodynamic[8], and dynamical[9] properties of complex and slow relaxing systems, the simplification of the chemical structure of the molecular system prevents gaining a detailed picture of its atomistic interactions unless atoms are reinserted in the coarse model during a second stage of the simulation[10, 11]. One of the solutions to this problem is the development of molecular models which combine two levels of resolution one of which is at the atomistic level. In literature, there are already a few examples of such models developed for simple fluids[12], polymers[1, 13, 14], and selected biological systems[15, 16, 17, 18, 19, 20]. The development of such models poses of course several fundamental questions related to their thermodynamics[2, 3, 21]. In particular, in cases where atoms are combined with beads, two particle-based models characterized by different dynamics are mixed[22, 23]. In most of the cases it has been shown that the dynamics of a CG model is the same as the corresponding atomistic one except that the former is some orders of magnitude faster than the latter[9, 24]. Additionally, in a CG model, the collapse of several atoms into a single bead and consequently the removal of the fast oscillations of the atomatom bond interactions allow the use of a time step (Δt) to integrate the equation of motion which can be considerably larger than that used for all-atoms (AA) models. In CG MD simulations, a Δt between 10 and 20 fs is considered appropriate, whereas in an AA simulation, a Δt of 1 or 2 fs (depending on whether bonds involving the light hydrogen atoms are allowed to oscillate) is normally used.

Therefore, when two models whose dynamics can be sampled at two different time scales are mixed together, a multiple time step (MTS) scheme might be in principle applied. The use of a MTS algorithm would in fact fully exploit the advantage of using a dually resolved model and would speed up the simulation time even further by reducing considerably (as a function of the number of beads in the system) the computational time. In the past years, several MTS schemes have been proposed and validated for fully atomistic models. They include: generalized Verlet integration[25], RESPA[26] and its derivation MOLLY[27], and Langevin normal mode methods[28]. These methods are all based on the fact that already in an atomistic resolved model, it is not always necessary to sample all the atom pair interactions with the same frequency and, because of the existence of different time and spatial scales in the interactions, every component of the interaction could be considered with its own specific time.

This approach in theory increases the efficiency of the simulation because of the computational time saved not computing interactions that can be sampled less fre-

quently. For an AA model, however, the computational time saved is limited and the different time step lengths must be chosen with some criteria. In particular, due to the resonance instability[29], the interval between the sampling of slow interactions must be taken around half the period of the fastest vibrational oscillation. In the Langevin normal mode methods, this instability can be damped out by adding a friction term; however, this diverts the system from true Hamiltonian dynamics[30]. In addition, in purely atomistic models, sometimes the division between the different time scales is not straightforward and care must be taken for an efficient splitting of the different type of interactions[31].

In literature, there are few successful examples of the usage of MTS schemes to model polymer melts. For example, atomistic resolved polyethylene melt has been simulated using the reversible reference system propagator algorithm (rRESPA)[32, 33]. In this case, exploiting the intrinsic wide spectrum of time scale exhibit by polymers, the authors associated the fast varying part of the Liouville operator to the bonded (angles and torsions) interactions and the slow varying part to the nonbonded interactions. Using this splitting of the interactions, a time step up to 5 times the smallest one (2fs) could be used.

In a hybrid system composed by atoms and CG particles, the formulation of the MTS scheme can indeed take advantage of the fact that only the interactions between the CG beads can enter in the scheme leaving the atomistic part to evolve with its own time step. However, although there is an intrinsic division between different length and time scales in the system dynamics, a clear separation between bonded and nonbonded interactions cannot be made as beads and atoms are connected by harmonic potential. A clear distinction based on the interparticle distance cannot be made too as atoms and beads can be either partitioned in specific regions of the simulation box or (as in the present case) randomly distributed in the space. Moreover when a single molecule is modeled using a dual resolution (as in the case of polymers), the MTS scheme needs to be applied also to pairs covalently bonded if they are beads.

Recently we have proposed a simple algorithm to mix atoms and CG beads in modeling polymer melts[1]. This model already uses separate cutoff distances and frequency of neighbor list updates for atoms and beads. The model has been successfully applied to simulate high molecular weight melts of flexible (polyethylene) and semirigid (polystyrene) polymers and could open the way to multiscale modeling of polymer melts.

In this article, we propose a MTS scheme which uses a Taylor series expansion to approximate beadbead interactions which do not need to be sampled as frequently as the atomatom ones. The technique is akin to the first attempt to develop a MTS by Streett et al.[34] and although it has the same drawbacks (i.e., it does not conserve

the total energy and it is not symplectic), it appears a reasonable computational approach to be used in multiscale simulation.

6.3 Multiple Time Step scheme

In the MTS scheme, the equation of motion is integrated every time step, Δt, for certain particles and only every $m\Delta t$ (where m is a positive integer number) for others. In the case of a multiresolved model such that presented here, single molecules are modeled using both atoms and beads which can interact through both bonded and nonbonded interactions. Therefore, it appears natural that only atomatom interactions should be sampled every say 1 or 2 fs (Δt), while the much slower beadbead interactions should be integrated less frequently (every Δmt). However, due to the nature of the model, during these m time steps, as the motion of the atoms affects the positions of the beads both via the bonded and nonbonded potential, the forces acting on the latter ($\mathbf{F}_{BB}$) cannot be kept constant and should be approximated in some way. To do so we decide to use their Taylor expansion:

$$\mathbf{F}_{BB}(t_n + k\Delta t) = \sum_{i=0}^{+\infty} \frac{(k\Delta t)^i}{i!} \left(\frac{\partial^{(i)} \mathbf{F}_{BB}}{\partial t^{(i)}} \right)_{t_n} \text{with } k = 1, \ldots, m \tag{6.1}$$

where $\mathbf{F}_{BB}$ is the force acting between two CG beads resulting from all the contributions (e.g., bonds, angles, and nonbonded interactions), m is the number of time steps to approximate, t is the time (with inline image), and i is the Taylor series expansion order. The force $\mathbf{F}_{BB}$ does not explicitly depend on the time, and the time dependence is exerted through the value of the bead coordinates $\mathbf{x}(t)$. Accordingly, even if it is not correct to write $\mathbf{F}(t)$ [or its analogous discrete notation $\mathbf{F}(t_n)$], we will make use of the notation $\mathbf{F}(t)$ to intend $\mathbf{F}(\mathbf{x}(t))$ [and its discrete counterpart will be $\mathbf{F}(t_n)$ to intend $\mathbf{F}(\mathbf{x}(t_n))$].

To be used in a calculation, the Taylor series needs to be truncated at a certain value h and eq. 6.1 becomes

$$\mathbf{F}_{BB}(t_n + k\Delta t) = \sum_{i=0}^{h} \frac{(k\Delta t)^i}{i!} \left(\frac{\partial^{(i)} \mathbf{F}_{BB}}{\partial t^{(i)}} \right)_{t_n} \text{with } k = 1, \ldots, m \tag{6.2}$$

In this article, the choice of h (which is related to the truncation error in the above series expansion) is obtained from considerations on the order of accuracy of the algorithm used to integrate the Newton equation [in our case the leap-frog (LF) algorithm].

The LF algorithm which reads as

$$\mathbf{V}(t_{n+\frac{1}{2}}) = \mathbf{V}(t_{n-\frac{1}{2}}) + \mathbf{F}(t_n)\Delta t + \mathcal{O}(\Delta t^3) \tag{6.3}$$

and

$$\mathbf{x}(t_{n+1}) = \mathbf{x}(t_n) + \mathbf{V}(t_{n+\frac{1}{2}})\Delta t + \mathcal{O}(\Delta t^4) \tag{6.4}$$

where $\mathbf{V}(t_n)$ is the particle velocity at time t_n and $\mathbf{x}(t_n)$ is its position, is an algorithm of accuracy of order two [in eqs. 6.3 and 6.4, we consider the mass of the particles (atoms or beads) all equal to one for the sake of simplification of the notation]. If we substitute eq. 6.2 in eqs. 6.3 and 6.4, we obtain

$$\mathbf{V}(t_{n+\frac{1}{2}+k}) = \mathbf{V}(t_{n-\frac{1}{2}+k}) + \left(\sum_{i=0}^{h} \frac{k^i \Delta t^{i+1}}{i!} \left(\frac{\partial^{(i)} \mathbf{F}}{\partial t^{(i)}} \right)_{t_n} \right) + \mathcal{O}(\Delta t^3) \tag{6.5}$$

and

$$\mathbf{x}(t_{n+1+k}) = \mathbf{x}(t_{n-1+k}) + \mathbf{V}(t_{n+\frac{1}{2}+k})\Delta t + \mathcal{O}(\Delta t^4) \tag{6.6}$$

from which we can see that to conserve the order of accuracy of two we can choose $h = 1$. By putting $h = 1$ we can write eq. 6.5 as

$$\mathbf{V}(t_{n+\frac{1}{2}+k}) = \mathbf{V}(t_{n-\frac{1}{2}+k}) + \sum_{i=0}^{1} \frac{k^i \Delta t^{i+1}}{i!} \left(\frac{\partial \mathbf{F}(t_n)}{\partial t} \right)_{t_n} + \mathcal{O}(\Delta t^3) \tag{6.7}$$

and then

$$\mathbf{V}(t_{n+\frac{1}{2}+k}) = \mathbf{V}(t_{n-\frac{1}{2}+k}) + \mathbf{F}(t_n)\Delta t + k\Delta t^2 \left(\frac{\partial \mathbf{F}(t_n)}{\partial t} \right) + \mathcal{O}(\Delta t^3) \tag{6.8}$$

We decide to calculate the derivatives of the force which appear in eq. 6.8 using a numerical derivation using a Centered Approximation Scheme. Therefore, to calculate the derivatives with at least an accuracy of order two and maintain the accuracy of the algorithm, three values of the forces must be calculated explicitly and the derivative can be written as

$$\left(\frac{\partial \mathbf{F}(t_n)}{\partial t} \right)_{t_n} = \frac{\mathbf{F}(t_{n-2}) - \mathbf{F}(t_n)}{2\Delta t} + \mathcal{O}(\Delta t^3) \tag{6.9}$$

The complete algorithm includes thus two different parts: (i) for three time steps the forces are evaluated every inline image with the LF algorithm [eqs. 6.3 and 6.4]; (ii) these values of the forces are then collected and used for the calculation of the forces for the next m time steps using the Taylor expansion [eq. 6.8]. The choice of m should depend on the system under investigation. Within our multiscale model, the MTS method is used to approximate the forces only between beads [i.e., between beads which are not directly connected (i.e., through bonds) to atoms]. The computational advantage gained in using this MTS scheme depends on the number of time steps for which the beadbead are approximated (m) and the number of

atoms (N_A) and beads (N_{CG}) of the system. If we do not account for the cpu time spent in computing the numerical derivatives necessary for the Taylor expansion, knowing the number of time steps that are calculated explicitly, i, and the number of time steps approximated with the Taylor expansion, m, and considering that the evaluation of the intermolecular forces requires calculation time proportional to N^2 (where $N = N_{CG} + N_A$), is the total number of particles in the system), the computer time saved in using the MTS scheme is roughly proportional to the number of CG beads in the system times m, the number of time steps approximated with the Taylor expansion.

6.4 Computational Details

The MTS scheme is tested on both a CG model and a hybrid AA-CG model developed for a melt of atactic polystyrene. Details about the models can be found in these two references[1, 35]. Their features are briefly reported below. The CG model is developed using the IBI procedure which develops the beadbead interactions using structural properties obtained from detailed atomistic simulations performed on the system of small size[36]. Distributions of bond distances, angles, torsions (when necessary), and radial distribution functions (RDFs), are subjected to a Boltzmann inversion [see eq. 6.10, where k_B is the Boltzmann constant and T is the temperature and $P(r)$ is a pair distribution] to find the corresponding potentials of mean force which, to become the effective, pairwise potential used in the simulation, is then iteratively optimized against these structural information.

$$V_1(r) = -k_B T \log(P(r)) \tag{6.10}$$

In this work, the IBI CG potential used to model the polystyrene melt is that developed by Qian et al.[35] that has the advantage of using a simple mapping scheme (one bead corresponds to one monomer unit). Atoms in each PS monomer are merged into one CG bead located at the center of mass of the repeat unit. The CG superatoms are distinguished as R and S according to their absolute configuration of the parent monomers.

The CG-AA model is that reported in Ref. [1] and consists of beads and atoms embedded into the same macromolecular chain (see Fig. 6.1). In this case, each polymer chain is modeled using both atoms and CG beads in equal number (i.e., 50% of the monomers are at low, CG, resolution and 50% of them are modeled at high, AA, resolution). The intermolecular potential that acts among the CG beads is that derived from the IBI procedure explained above while for the atomistic part the Transferable Potentials for Phase Equilibria Force Field (TraPPE) force field

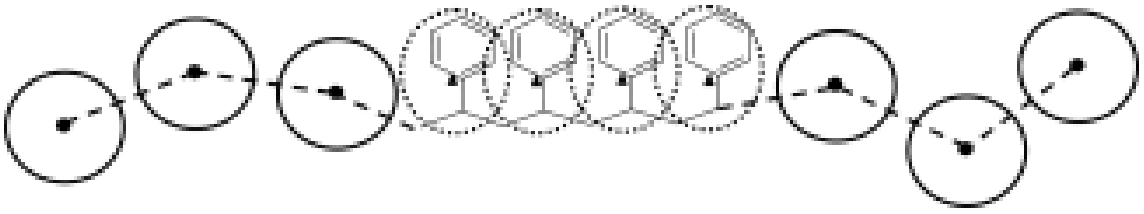


Figure 6.1: Example of the possible structure of the hybrid model for polystyrene: the black filled circles represent the centers of mass of the CG beads, whereas the black filled triangles represent the position of the VS in the atomistic fragment of the polymer chain. In this example, the polymer chain is formed by six CG (solid line) and four atomistic (dotted line) resolved monomers.

is used[37]. The total energy of the hybrid model is calculated as the sum of the contributions coming from the atomsatoms, beadsbeads, and mixed (atoms-beads) interactions. The latter include bonded (which ensure the connectivity of the chain) and nonbonded ones. The bonded terms are modeled using a harmonic potential which is the only potential through which atoms and beads directly interact. All the other mixed interactions are mediated by the virtual sites (VS) which are used as a pinning point to collect and distribute to the atoms the intermolecular forces due to the interactions with the beads.

The melt is composed of 15 chains each of 80 monomers (in the CG-AA model 40 monomers modeled as beads and 40 modeled with atoms). The simulations are performed in the canonical ensemble with a temperature set at 500 K (i.e., above the experimental glass transition temperature) for 100 ns in all cases. The density of the system is set to 940 kg m^{-3}, which corresponds to the experimental density at ambient conditions[35]. Table 6.1 summarizes all the simulation details. It should be noticed that although the pressure is not coupled with any barostat its value oscillates around 1 atm. These results show that the hybrid AA-CG force field reproduces the correct density. Three different sets of simulations are performed: one using a pure CG model where the MTS scheme with $m = 10$ [see eq. 6.2, where m represents the number of time steps during which the forces are approximated] is used; another one performed on the CG-AA model where different value of m (6, 8, and 10) are used. For comparison, a simulation of the hybrid CG-AA model without using the MTS scheme is also performed and analyzed. All the simulations have been carried out with a modified version of the parallel code IBIsCO[38], using the Berendsen thermostat[39] with coupling time of 0.2 ps. The smallest time step used is 1 fs.

Model type	Length of the simulation	Value of m
CG	100 ns	10
CG-AA	100 ns	0 (no MTS)
CG-AA	100 ns	6
CG-AA	100 ns	8
CG-AA	100 ns	10

Table 6.1: List of the simulations performed. CG stands for pure coarse-graining, CG-AA is the hybrid model, $m = 0$ (see eq. 6.2) indicates that the multiple-time-step scheme is not applied.

6.5 Results

6.5.1 Energy conservation and error on the force

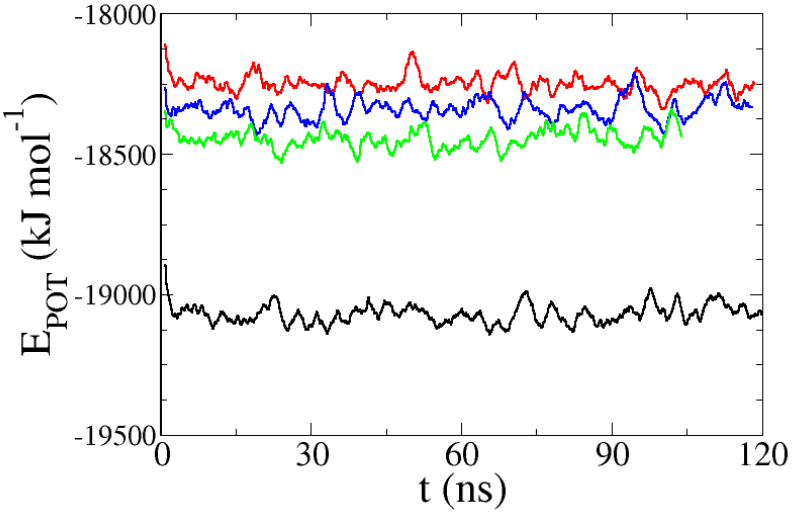
During the simulations that used the MTS scheme, the forces acting between the CG beads are approximated using eq. 6.2. The algorithm proposed is not time reversible and the total energy of the system is not conserved, however, when simulations are performed in the canonical ensemble, no drift in both potential and kinetic energy can be observed (see Fig. 6.2) and the total energy is kept constant by the action of the thermostat. It is interesting to notice that the three MTS schemes produce almost the same fluctuations in both kinetic and potential energy components of the total energy (Fig. 6.2) irrespectively to the value of m used. As the fluctuations in the potential energy should not be affected by the use of the thermostat,[27] this result might indicate that the use of the MTS affects mainly only the oscillations of the kinetic component of the total energy as it is shown in Figure 6.2.

To quantify whether there are any drifts in the energy, the following analysis is performed: first the energy values are interpolated through a least-squares fit of the data to a straight line, and then the averaged deviation (ΔE) from the fitted value (E_0) is calculated as

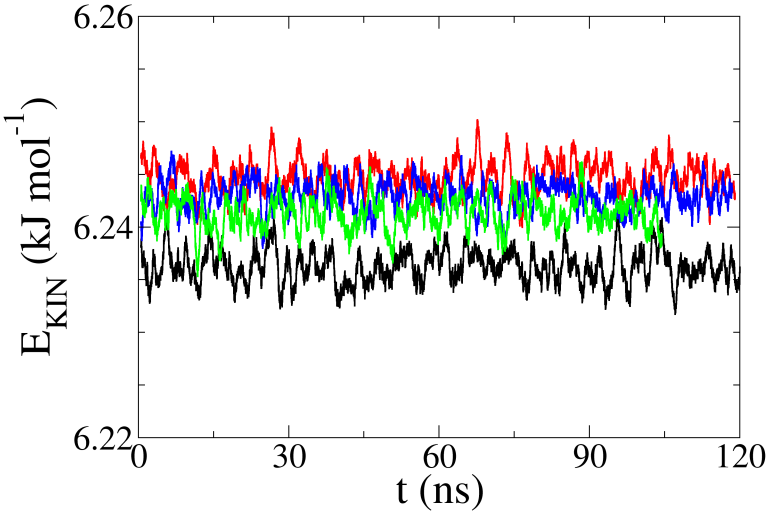
$$\Delta E = \frac{1}{t_{max}} \int_0^{t_{max}} \left| \frac{E(t) - E_0}{E_0} \right| dt \tag{6.11}$$

where t_{max} is the total time of the simulation, t is the time, and $E(t)$ is the energy at time t. Using this definition, the total drift of the energy is of the order of 10^5 for the kinetic and 10^4 for the potential component irrespectively if the MTS scheme is applied or not.

The averaged difference between the approximated force and its true value can be estimated calculating the weighted average fractional deviation ΔF between the beadbead forces as approximated by the Taylor expansion ($\mathbf{F}^a$) and its true value ($\mathbf{F}^t$) calculated integrating the equation of motion every time steps,



(a)



(b)

Figure 6.2: Running average (calculated every 200 ps) of the potential (E_{POT}) and kinetic (E_{KIN}) energy per mole of molecules extracted from the simulations performed on the hybrid AA-CG model without the use of the MTS scheme (black curve) and with the MTS scheme with $m = 10$ (red curve), $m = 8$ (blue curve), and $m = 6$ (green curve). The meaning of the symbol m is explained in eq. 6.2 and in the text. From the values of the kinetic energy, the kinetic temperature (T_K) can be obtained using the following formula $T_K = \frac{2}{3}\frac{1}{k_B N_A} E_{KIN}$ where k_B is the Boltzmann constant and N_A is the Avogadro number.

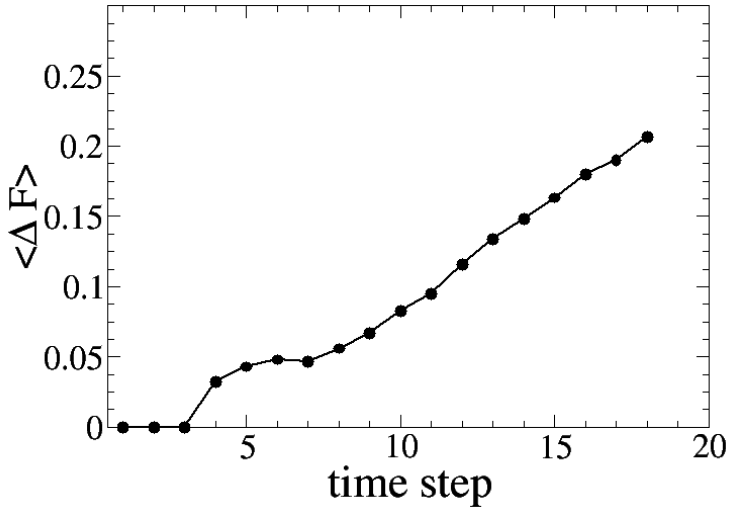


Figure 6.3: Error on the approximated forces calculated using eq. 6.12.

$$\Delta F = \frac{1}{3}\sum_{i=1}^{N_{CG}}\sum_{\alpha=x,y,z}\left(\left|\frac{\mathbf{F}_{i\alpha}^{t}-\mathbf{F}_{i\alpha}^{a}}{\sum_{i=1}^{N_{CG}}|\mathbf{F}_{i\alpha}^{t}|}\right|\right) \tag{6.12}$$

where N_{CG} is the total number of beads. In the summation, each force difference is weighted by the total force ($\sum_{i=1}^{N_{CG}}|\mathbf{F}_{i\alpha}^{t}|$) to take into account the fact that forces of very different magnitude act on the system. In fact, the values of intermolecular forces coming from the soft beadbead interactions (IBI potentials) are much smaller than those derived from the atomatom interactions modeled using the Lennard-Jones potential. The value of ΔF can be used to provide an initial criterion for selecting the MTS parameter m. The fractional deviation values ΔF are reported in Figure 6.3. The first three points represent the time steps for which the force is calculated exactly (necessary to compute the numerical derivative [see eq. 6.9]. As expected the error increases with the number of time steps approximated. The error is calculated for approximation up to $m = 15$. As it will be shown below, a value of $m = 10$ to which corresponds a value of $\Delta F < 0.15$ is a reasonable choice for this system.

6.5.2 Effect of the MTS scheme on structural properties

The macroscopic properties of polymer melts are dictated by the polymer chain structure and bulk morphology. The most important test to do on the MTS scheme proposed here is then to check whether it can preserve the correct structural properties over a long period of time. Table 6.2 reports the values for the radius of gyration (R_g) and end-to-end distance (R_{ee}) calculated for the different systems under investigation. The MTS scheme with a value of m up to 10 does not alter the

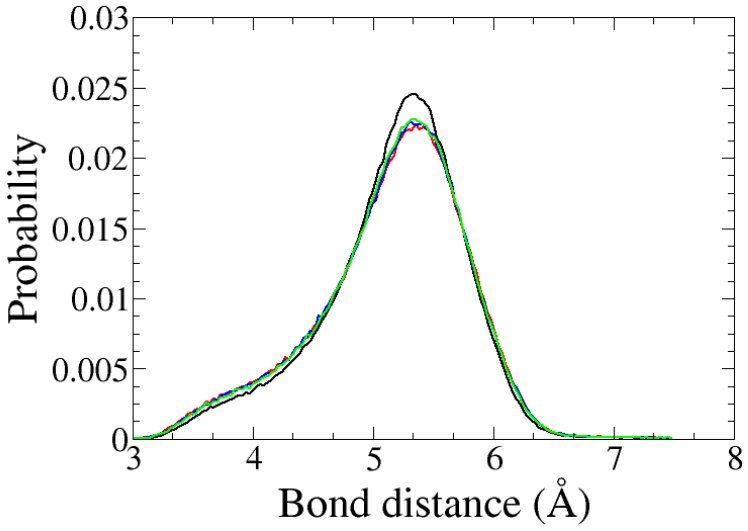


Figure 6.4: Distributions of the distances between beadsbeads, beads-virtual site along the polymer chain obtained from the simulation without the MTS scheme applied and with the MTS scheme applied with $m = 6$ (MTS6), $m = 8$ (MTS8), and $m = 10$ (MTS10). Color scheme as in Figure 6.2.

single chain properties which are all within their errors. Figures 6.4 and 6.5 show that the angles and bonds distributions along the chain calculated between beads and VS are only mildly affected by the application of the MTS scheme.

Figure 6.6 shows the intrachain RDFs calculated among the atoms belonging to the atomistic resolved segment of the polymer chain. The comparison is made

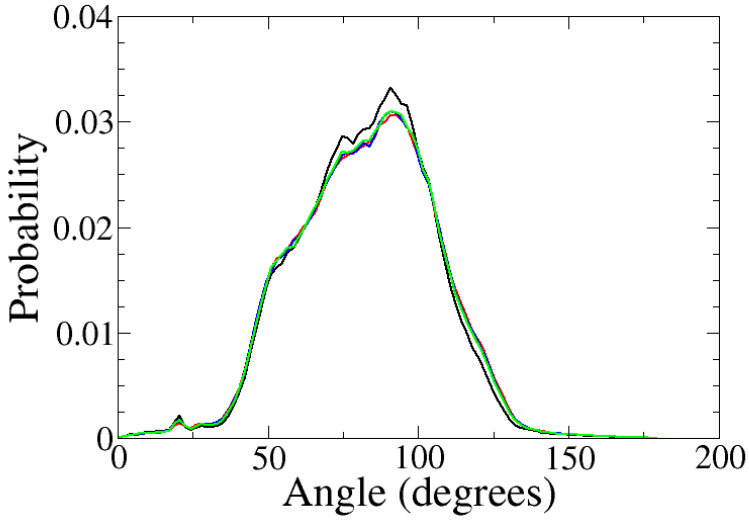


Figure 6.5: Distributions of the plane angles between bead-bead-bead, bead-VS-VS, bead-bead-VS, VS-VS-VS along the polymer chain obtained from the simulation without the MTS scheme applied and with the MTS scheme applied with $m = 6$ (MTS6), $m = 8$ (MTS8), and $m = 10$ (MTS10). Color scheme as in Figure 6.2.

Model type	Value of m	$\mathbf{R_g}$ (Å)	$\mathbf{R_{ee}}$ (Å)
CG	0 (no MTS)	21.8±0.5	52±5
CG	10	22.7±0.6	56±6
CG-AA	0 (no MTS)	21.0±0.9	53±9
CG-AA	6	20.2±0.7	48±8
CG-AA	8	20.1±0.7	48±8
CG-AA	10	19.9±0.8	47±7

Table 6.2: Radius of gyration (R_g) and end-to-end distance (R_{ee}) calculated for the various systems under investigation.

between the curve obtained from a simulation without MTS scheme (time step of 1 fs) and those obtained with the MTS scheme applied with $m = 6$, 8, and 10 [see eq. 6.2]. It appears clear that the use of the MTS scheme does not affect the chain structure at the atomistic level irrespective of the number of time steps approximated with the Taylor expansion. This result is reasonable as the magnitude of the bonding part of the force (which is not approximated) exerted by the atoms on the atoms is larger in magnitude compared with that (bonding and nonbonding) due to the CG beads (which is approximated). Moreover the position of the atoms is only indirectly affected by the use of the MTS scheme. In fact, all the atomatom and atombead (mediated through the VS) interactions are calculated every time steps and the only effect on the atoms positions is that due to the relative positions of the beads (whose interactions have been instead approximated). We will see later however that although the atoms should only be marginally affected by the application of the MTS scheme, their interchain relative positions are actually slightly perturbed when the forces are approximated. The total interchain RDFs calculated between the centers of mass of all beads and VS in the chain compare also very well with that obtained from the system simulated without MTS (Fig. 6.7). However, if the contributions to the total RDF coming from the atoms (VS) and beads are accounted separately, we can observe discrepancy in the distributions (Fig.6.8). The RDFs calculated separately only between the atomistic resolved monomers (using the position of the VS, Fig. 6.8a) or the CG ones (Fig. 6.8b) seem to indicate that the interchain atoms positions are slightly affected by the use of the MTS scheme which pushes both the atoms and the beads a little bit closer to each other. Beads and atoms are instead slightly more far apart compared with the simulation performed without MTS applied (Fig. 6.8c). Although these small discrepancies in the height of the peaks their positions are always correct and anyway the difference between the curves is always within the RDF uncertainty (calculated as the standard deviation of the RDF of each chain) which is about 2%. It can also be noticed that all the three MTS schemes provide almost the same results indicating that approximating the force for 6, 8, or 10 time steps does not affect the results. To verify what is

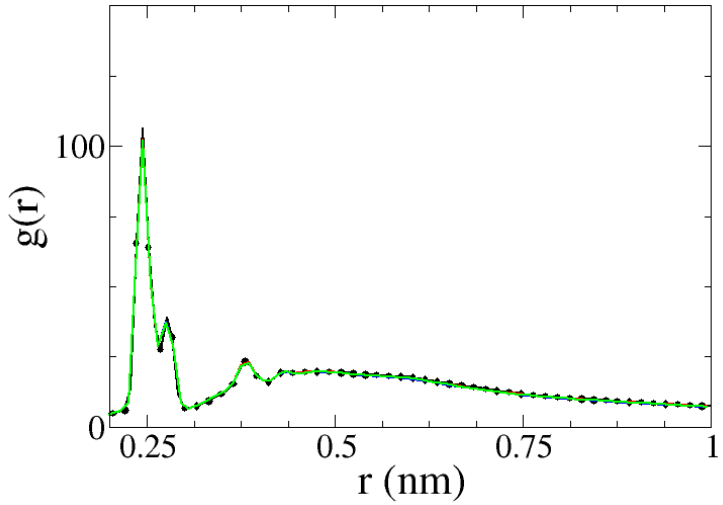


Figure 6.6: Comparison between the RDFs ($g(r)$) calculated for the atoms belonging to the same chain performed without the MTS scheme (black circles) and with the MTS scheme with $m = 6$, 8, and 10. Color scheme as in Figure 6.2.

the highest number of time steps that can be approximated, we perform simulations with $m = 15$ and $m = 19$. We notice that in both cases, the system is not stable and the simulation stops after few cycles. Finally, the structural form factor $S(k)$[10] (Fig. 6.9) calculated as

$$S(k) = 1 + 4\pi\rho \int_0^b r^2 (g(r) - 1) \frac{\sin(kr)}{kr} dr \tag{6.13}$$

where $\rho = N/V$ is the number density of beads and VS, $g(r)$ is the corresponding interchain and intrachain RDF, and b is the half of the simulation box length, is obtained from all the simulations performed. As expected since both intra and inter- match with each other, the three $S(k)$ curves overlap. The spectra present the typical peak at 1.4 $\mathring{A}^{-1}$, however, maybe due to the fact that the calculation is done excluding the atoms but including only the VS (i.e., a pseudo-CG model), the peak at lower k known as polymerization peak (at around 0.75 $\mathring{A}^{-1}$)[10], which can be observed in experimental and AA simulation spectra, is only just visible. In Figure 6.9, the $S(k)$ calculated for a single chain using the definition inline image where r_{ij} represents the distance between the bead (or VS) i and j belonging to the same chain, is also reported. The plot is normalized against the chain radius of gyration (see Table 6.2) and the number of beads within the chain ($N = 80$). In Figure 6.9b, the same single chain $S(k)$ is plotted in logarithmic scale together with the theoretical curve corresponding to a polymer chain in theta solvent ($S(k) \propto k^{-1/\nu}$ with $\nu = 0.5$)[40]. A perfect match between the curves can be observed, showing that the polymer chains are all well relaxed and converge to the same configuration.

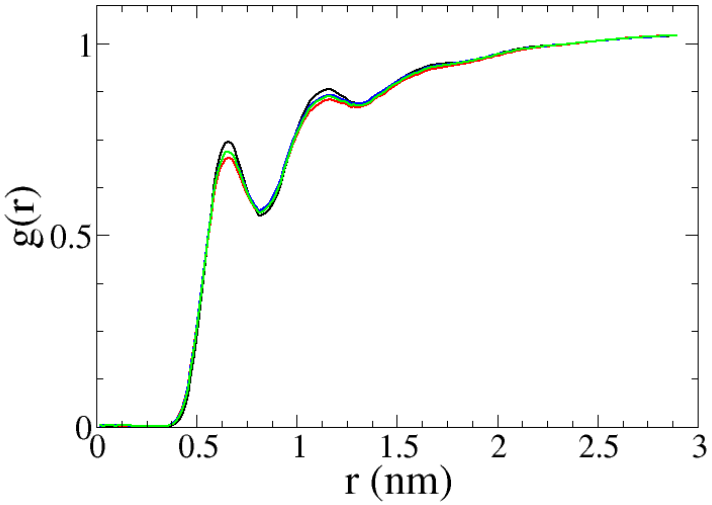


Figure 6.7: Comparison between the intermolecular RDF ($g(r)$) calculated between the center of mass of the beads and VS for the simulation performed without MTS scheme and with MTS with m = 6, m = 8, and m = 10. Color scheme as in Figure 6.2.

At this point it is interesting to check whether the approximation introduced by the use of the Taylor expansion is, itself, reasonable. Figure 6.10 reports the total intermolecular RDF calculated for the pure CG model with and without the use of the MTS scheme ($m = 10$). It can be observed that no artifacts on the structural properties of the melt are introduced by the use of the MTS algorithm.

6.5.3 Effect of the MTS scheme on the dynamical properties

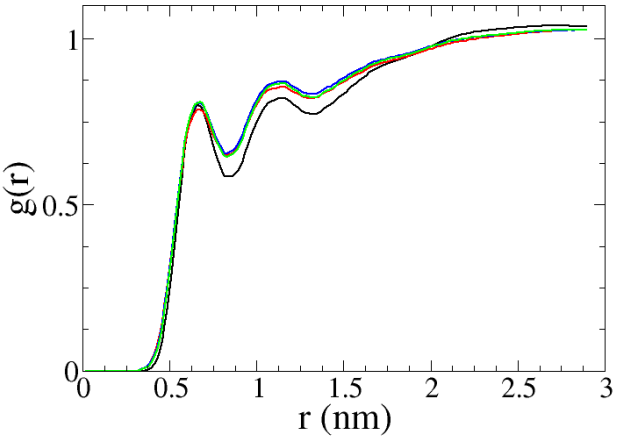
To verify whether the proposed MTS scheme generates the correct dynamics of the system, the spectral densities calculated on the MD trajectory obtained from the hybrid model simulated with m = 0, m = 6, 8, and 10 are compared with the pure atomistic and CG models (simulated without MTS scheme). The spectral density $I(\nu)$ is calculated from the Fourier transformation of the velocity autocorrelation functions $C_{VV}(t)$ through the formula

$$I(\nu) = \frac{1}{\pi} \int_0^{+\infty} C_{VV}(t) \cos(\nu t) dt \tag{6.14}$$

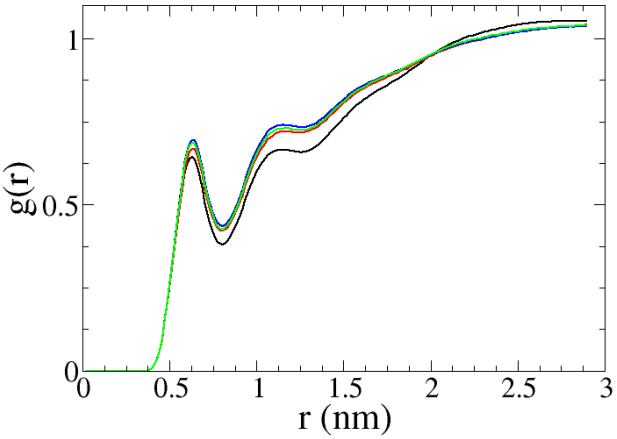
where C_{VV} is the normalized autocorrelation function of the velocities (V) of all the particles (atoms and beads) present in the system at time t

$$C_{VV}(t) = \frac{\langle \mathbf{V}(0) \cdot \mathbf{V}(t) \rangle}{\langle \mathbf{V}(0) \cdot \mathbf{V}(0) \rangle} \tag{6.15}$$

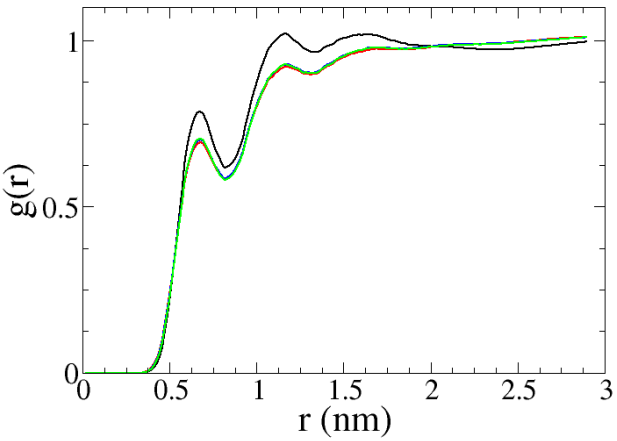
The spectrum obtained from eq. 6.14 contains all the system vibration frequencies



(a) Atomistic resolved monomers.

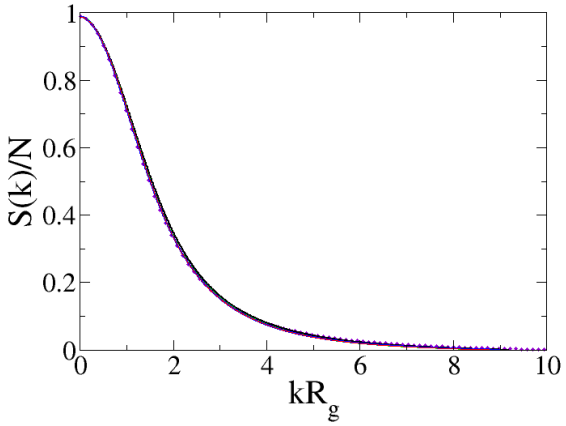


(b) CG resolved monomers.

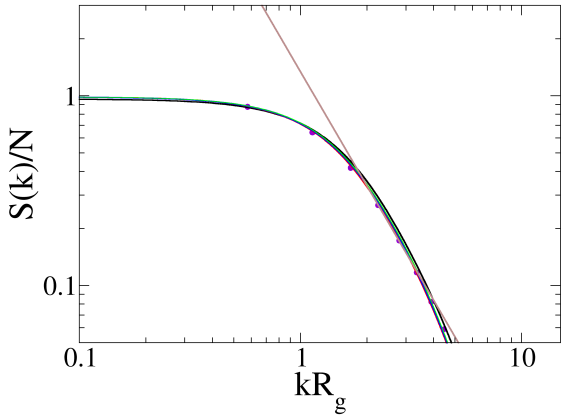


(c) Mixed contributions.

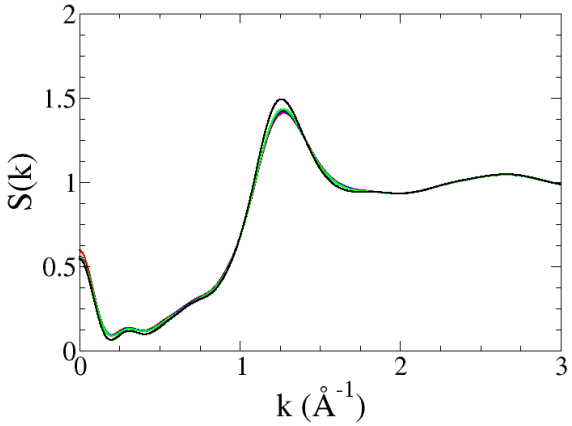
Figure 6.8: Comparison between the RDFs ($g(r)$), calculated between the centers of different components. Color scheme as in Figure 6.2.



(a) The normalized single chain structure factor calculated for the mono-resolved CG model (violet circles), hybrid model simulated without the MTS scheme (black curve) and with the MTS scheme $m = 6$, 8, and 10 applied. Color scheme as in Figure 6.2.



(b) Same as (a) but in log scale. In brown: the theoretical curve with slope 2 (i.e., $nu = 0.5$).



(c) Global structure factor calculated for the hybrid model simulated without the MTS scheme and with the MTS scheme $m = 6$, 8, and 10.

Figure 6.9: Color scheme as in Figure 6.2.

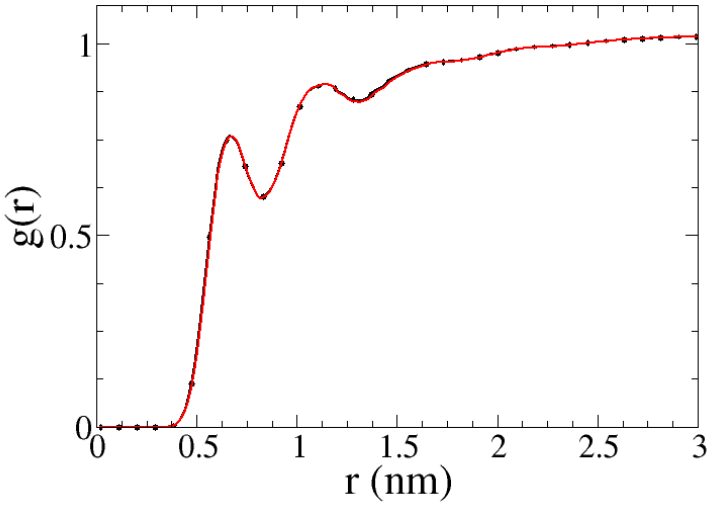


Figure 6.10: Comparison between the intermolecular RDF ($g(r)$) calculated between the center of mass of the beads from the simulation performed on the pure CG model without (black circles) and with the MTS scheme $m = 10$ (solid red line).

(ν) that our MD simulations are able to capture considering that a time step of 1 fs is used (see Fig. 6.11a). Therefore, the $I(\nu)$ obtained for the pure atomistic model shows for example peaks in the frequency range typical of the aromatic C=C stretching (around 1500 cm^{-1}) also visible in the experimental infrared spectra of polystyrene[41] and other peaks at lower ν related to slower motions along the chain. As expected the spectrum obtained for the CG model does not present any peaks a high frequency but only a large one at very low ν. It is interesting to notice that the spectrum of the hybrid model maintains all the peaks at high frequencies typical of the AA model and also that broad one visible in the CG one showing once again that the technique used to merge the AA and CG potential is able to preserve all the features of both models (Fig. 6.11a). Figure 6.11b shows the comparison between the spectral densities obtained from the simulations carried out with the MTS scheme and clearly demonstrates that no alteration in the dynamics of the system is introduced. Finally, the global diffusion of the polymer chains is analyzed in terms of chain selfdiffusion coefficient (D) obtained using the Einstein relation

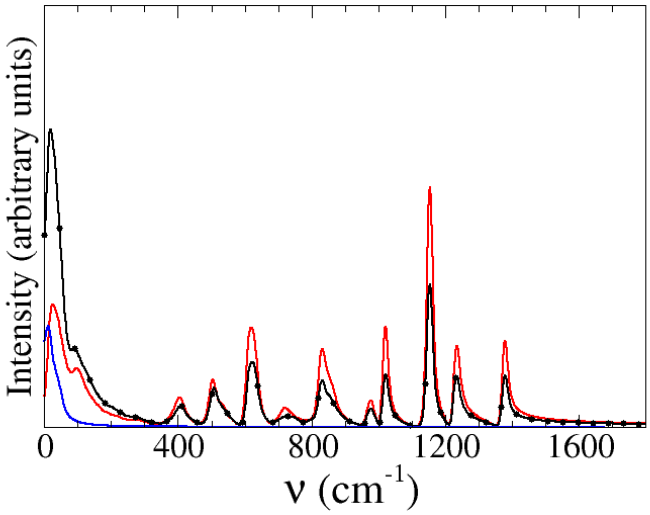
$$D = \lim_{t\to\infty} \frac{(R(t) - R(0))^2}{6t} \tag{6.16}$$

where $(R(t) - R(0))^2$ represents the mean square displacement (MSD) of each chain being R(t) the position of the center of mass of the polymer chain at time t. Figure 6.12 shows the MSDs calculated for three systems simulated with the MTS scheme and compares them with the curve obtained from the simulation performed with the standard LF algorithm (without the MTS applied). From the figure it

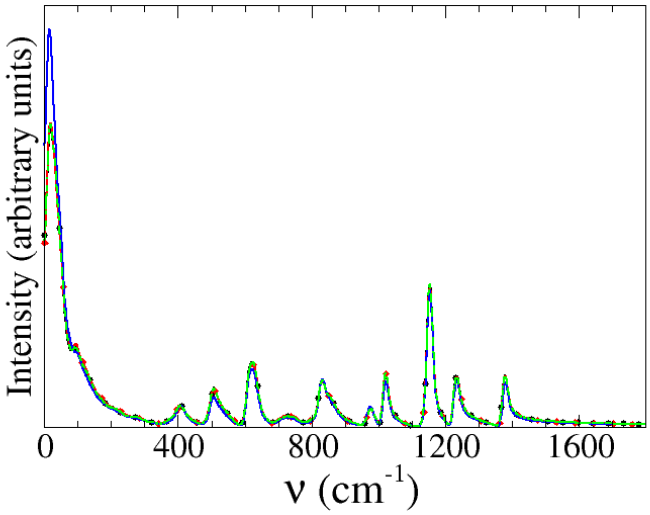
appears clear that all the four models have reached the diffusive regime in which MSD$\sim t$. The resulting diffusion coefficients are 2.0×10^{-6} ($\pm1.8\times10^{-8}$), 2.7×10^{-6} ($\pm4.4\times10^{-8}$), and 2.8×10^{-6} ($\pm2.5\times10^{-8}$) cm^{2}s^{-1} for MTS scheme using m = 6, 8, and 10, respectively. These numbers compare well with the D obtained from the simulation performed with the standard LF algorithm (no-MTS) which is 3.2×10^{-7} ($\pm1.4\times10^{-8}$) cm^{2}s^{-1} although it can be noticed that the difference between the diffusion coefficient obtained from the simulation performed with the standard LF algorithm and those performed with the MTS scheme becomes larger for higher value of m. Figure 6.12 presents the MSDs calculated for the simulations with the MTS scheme applied, shifted by a factor, τ, along the time axis to match the MSD of the reference (no-MTS) simulation. This method shows that a perfect overlap between the curves can be obtained after about 10 ns. However, for very short time (i.e., in subdiffusive regime), the application of the MTS scheme seems to affect the dynamics of the system which shows a higher slope of the MSD than expected (0.91 for m = 6, 0.93 for m = 8, 0.95 for m = 10 to be compared with 0.85 for the system without MTS). Here, it is important to notice that the dynamics of the newly proposed hybrid model of polymer chain needs further investigation to verify whether it reproduces the three diffusive regimes typical of polymer melts.

6.6 Summary and Conclusions

In this manuscript, we propose a simple MTS scheme for simulations performed on multiresolved particle-based models where atoms and CG beads are mixed together. The scheme approximates only the CG beadbead interactions calculating the corresponding forces with their Taylor expansions of order 2. All the other interactions (atomatom and atombead) are calculated every time step (1 fs). The scheme is applied to the simulation of a melt of atactic polystyrene for which the hybrid model has been previously developed[1]. The procedure is able to correctly reproduce the structural properties of the melt and does not alter the spectral density and the free diffusion of the system. In the latter case, however, it must be noticed that, irrespectively whether the MTS scheme is applied or not, in the subdiffusive regime, the mean square displacement of the center of mass of the chain is characterized by a slope which is higher than that predicted by the theory. This result indicates that a detailed analysis of the dynamics of the hybrid model is necessary to understand in which time range the dynamic properties are correctly captured. The number of time steps which can be approximated by the MTS scheme can be estimated by calculating the weighted average fractional deviation between the approximated beadbead forces and their true values through performing a very short preliminary simulation. In calculating this difference, it is, however, necessary to consider that



(a)



(b)

Figure 6.11: (a): Spectral density [$I(\nu)$ see eq. 6.14 in the text] for the monoresolved CG (blue solid line), AA (red solid line) models and hybrid model (dot-solid black line) as obtained from simulations without the usage of the MTS scheme. (b) Comparison between the spectral density obtained for the hybrid model when no MTS scheme is applied (dot-solid black line) and when the MTS scheme with m = 6, 8, and 10 is applied (green, blue, and red lines, respectively).

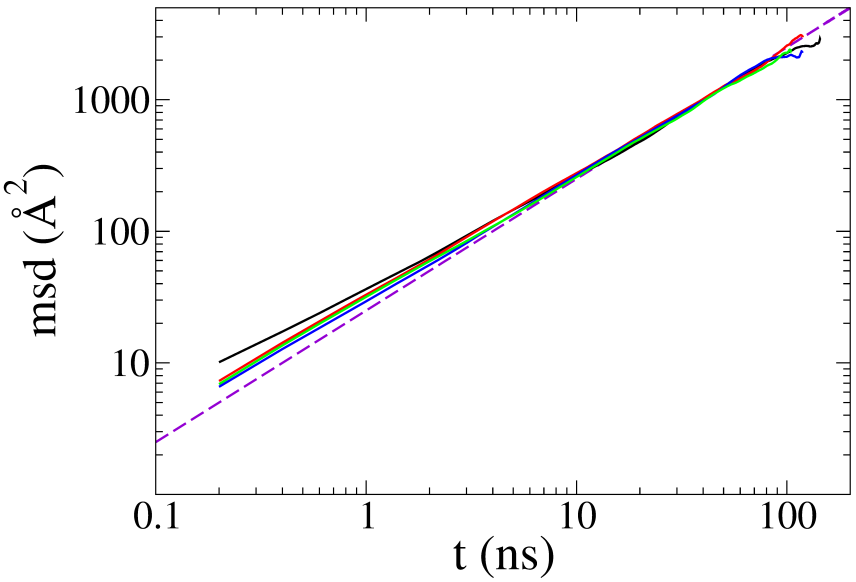


Figure 6.12: Comparison between the mean square displacement obtained for the hybrid model when no MTS scheme is applied (solid black line) and when the MTS scheme with $m = 6$, 8, and 10 is applied (green, blue, and red lines, respectively). The latter curves are shifted of a factor $\tau = 6.3$, 8.4, and 8.8, respectively. The purple dashed line of slope 1 indicates the diffusive regime. The dashed black line has a slope of 0.85 and indicates the subdiffusive regime shown by the hybrid model simulated without the MTS scheme applied.

forces of different magnitude act on the beads and therefore each force needs to be weighted by the total force. The results suggest that for the system under investigation 10 time steps can confidently be approximated corresponding to an error on the force less than 15%. This value is also reasonable as it matches with standard time step values used for CG simulations performed with the IBI method. The computational gain in using the MTS scheme depends on two factors: the number of time steps approximated and the number of CG beads in the model. In the specific case studied here, the computational gain turns out to be around 20%.

Bibliography

[1] N. Di Pasquale, D. Marchisio, and P. Carbone, "Mixing atoms and coarse-grained beads in modelling polymer melts," *The Journal of Chemical Physics*, vol. 137, no. 16, p. 164111, 2012.

[2] S. O. Nielsen, R. E. Bulo, P. B. Moore, and B. Ensing, "Recent progress in adaptive multiscale molecular dynamics simulations of soft matter," *Phys. Chem. Chem. Phys.*, vol. 12, pp. 12401–12414, 2010.

[3] M. Praprotnik, L. Delle Site, and K. Kremer, "Multiscale simulation of soft matter: From scale bridging to adaptive resolution," *ANNUAL REVIEW OF PHYSICAL CHEMISTRY*, vol. 59, pp. 545–571, 2008.

[4] P. Carbone and C. Avendaño, "Coarse-grained methods for polymeric materials: enthalpy- and entropy-driven models," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 4, no. 1, pp. 62–70, 2014.

[5] H. A. Karimi-Varzaneh, N. F. A. van der Vegt, F. Müller-Plathe, and P. Carbone, "How good are coarse-grained polymer models? A comparison for atactic polystyrene," *ChemPhysChem*, vol. 13, no. 15, pp. 3428–3439, 2012.

[6] P. Carbone, F. Negri, and F. Müller-Plathe, "A coarse-grained model for polyphenylene dendrimers: Switching and backfolding of planar three-fold core dendrimers," *Macromolecules*, vol. 40, no. 19, pp. 7044–7055, 2007.

[7] P. Carbone and L. Lue, "Prediction of bulk density and molecular packing in model dendrimers with different chain stiffness," *Macromolecules*, vol. 43, no. 21, pp. 9191–9197, 2010.

[8] P. Carbone, H. A. Karimi-Varzaneh, X. Chen, and F. Müller-Plathe, "Transferability of coarse-grained force fields: The polymer case," *The Journal of Chemical Physics*, vol. 128, no. 6, p. 064904, 2008.

[9] H. A. Karimi-Varzaneh, F. Müller-Plathe, S. Balasubramanian, and P. Carbone, "Studying long-time dynamics of imidazolium-based ionic liquids with

a systematically coarse-grained model," *Phys. Chem. Chem. Phys.*, vol. 12, pp. 4714–4724, 2010.

[10] P. Carbone, H. A. Karimi-Varzaneh, and F. Müller-Plathe, "Fine-graining without coarse-graining: an easy and fast way to equilibrate dense polymer melts," *Faraday Discuss.*, vol. 144, pp. 25–42, 2010.

[11] X. Chen, P. Carbone, G. Santangelo, A. Di Matteo, G. Milano, and F. Müller-Plathe, "Backmapping coarse-grained polymer models under sheared nonequilibrium conditions," *Phys. Chem. Chem. Phys.*, vol. 11, pp. 1977–1988, 2009.

[12] M. Praprotnik, L. Delle Site, and K. Kremer, "Adaptive resolution molecular-dynamics simulation: Changing the degrees of freedom on the fly," *The Journal of Chemical Physics*, vol. 123, no. 22, p. 224106, 2005.

[13] S. O. Nielsen, P. B. Moore, and B. Ensing, "Adaptive multiscale molecular dynamics of macromolecular fluids," *Phys. Rev. Lett.*, vol. 105, p. 237802, 2010.

[14] L. Delle Site, S. Leon, and K. Kremer, "BPA-PC on a Ni(111) surface: The interplay between adsorption energy and conformational entropy for different chain-end modifications," *Journal of the American Chemical Society*, vol. 126, no. 9, pp. 2944–2955, 2004.

[15] M. R. Machado, P. D. Dans, and S. Pantano, "A hybrid all-atom/coarse grain model for multiscale simulations of dna," *Phys. Chem. Chem. Phys.*, vol. 13, pp. 18134–18144, 2011.

[16] G. S. Ayton, E. Lyman, and G. A. Voth, "Hierarchical coarse-graining strategy for protein-membrane systems to access mesoscopic scales," *Faraday Discuss.*, vol. 144, pp. 347–357, 2010.

[17] M. Orsi, W. E. Sanderson, and J. W. Essex, "Permeability of small molecules through a lipid bilayer: A multiscale simulation study," *The Journal of Physical Chemistry B*, vol. 113, no. 35, pp. 12019–12029, 2009.

[18] Q. Shi, S. Izvekov, and G. A. Voth, "Mixed atomistic and coarse-grained molecular dynamics: Simulation of a membrane-bound ion channel," *The Journal of Physical Chemistry B*, vol. 110, no. 31, pp. 15045–15048, 2006.

[19] S. Izvekov and G. A. Voth, "Multiscale coarse-graining of mixed phospholipid/cholesterol bilayers," *Journal of Chemical Theory and Computation*, vol. 2, no. 3, pp. 637–648, 2006.

[20] M. Neri, C. Anselmi, M. Cascella, A. Maritan, and P. Carloni, "Coarse-grained model of proteins incorporating atomistic detail of the active site," *Phys. Rev. Lett.*, vol. 95, p. 218102, 2005.

[21] B. Ensing, S. O. Nielsen, P. B. Moore, M. L. Klein, and M. Parrinello, "Energy conservation in adaptive hybrid atomistic/coarse-grain molecular dynamics," *Journal of Chemical Theory and Computation*, vol. 3, no. 3, pp. 1100–1105, 2007.

[22] P. Español, "Hybrid description of complex molecules," *EPL (Europhysics Letters)*, vol. 88, no. 4, p. 40008, 2009.

[23] S. Matysiak, C. Clementi, M. Praprotnik, K. Kremer, and L. Delle Site, "Modeling diffusive dynamics in adaptive resolution simulation of liquid water," *The Journal of Chemical Physics*, vol. 128, no. 2, p. 024503, 2008.

[24] D. Fritz, K. Koschke, V. A. Harmandaris, N. F. A. van der Vegt, and K. Kremer, "Multiscale modeling of soft matter: scaling of dynamics," *Phys. Chem. Chem. Phys.*, vol. 13, pp. 10412–10420, 2011.

[25] H. Grubmùller, H. Heller, A. Windemuth, and K. Schulten, "Generalized verlet algorithm for efficient molecular dynamics simulations with long-range interactions," *Molecular Simulation*, vol. 6, no. 1-3, pp. 121–142, 1991.

[26] M. E. Tuckerman, G. J. Martyna, and B. J. Berne, "Molecular dynamics algorithm for condensed systems with multiple time scales," *The Journal of Chemical Physics*, vol. 93, no. 2, pp. 1287–1291, 1990.

[27] J. A. Izaguirre, S. Reich, and R. D. Skeel, "Longer time steps for molecular dynamics," *The Journal of Chemical Physics*, vol. 110, no. 20, pp. 9853–9864, 1999.

[28] G. Zhang and T. Schlick, "The Langevin/implicit Euler/normal mode scheme for molecular dynamics at large time steps," *The Journal of Chemical Physics*, vol. 101, no. 6, pp. 4995–5012, 1994.

[29] M. Mandziuk and T. Schlick, "Resonance in the dynamics of chemical systems simulated by the implicit midpoint scheme," *Chemical Physics Letters*, vol. 237, no. 5, pp. 525–535, 1995.

[30] E. Barth and T. Schlick, "Extrapolation versus impulse in multiple-timestepping schemes. II. linear analysis and applications to Newtonian and Langevin dynamics," *The Journal of Chemical Physics*, vol. 109, no. 5, pp. 1633–1642, 1998.

[31] J. A. Morrone, R. Zhou, and B. J. Berne, "Molecular dynamics with multiple time scales: How to avoid pitfalls," *Journal of Chemical Theory and Computation*, vol. 6, no. 6, pp. 1798–1804, 2010.

[32] V. A. Harmandaris, V. G. Mavrantzas, D. N. Theodorou, M. Kröger, J. Ramírez, H. C. Öttinger, and D. Vlassopoulos, "Crossover from the rouse to the entangled polymer melt regime: Signals from long, detailed atomistic molecular dynamics simulations, supported by rheological experiments," *Macromolecules*, vol. 36, no. 4, pp. 1376–1387, 2003.

[33] G. J. Martyna, M. E. Tuckerman, D. J. Tobias, and M. L. Klein, "Explicit reversible integrators for extended systems dynamics," *Molecular Physics*, vol. 87, no. 5, pp. 1117–1157, 1996.

[34] W. Streett, D. Tildesley, and G. Saville, "Multiple time-step methods in molecular dynamics," *Molecular Physics*, vol. 35, no. 3, pp. 639–648, 1978.

[35] H.-J. Qian, P. Carbone, X. Chen, H. A. Karimi-Varzaneh, C. C. Liew, and F. Müller-Plathe, "Temperature-transferable coarse-grained potentials for ethylbenzene, polystyrene, and their mixtures," *Macromolecules*, vol. 41, no. 24, pp. 9919–9929, 2008.

[36] D. Reith, M. Pütz, and F. Müller-Plathe, "Deriving effective mesoscale potentials from atomistic simulations," *Journal of Computational Chemistry*, vol. 24, no. 13, pp. 1624–1636, 2003.

[37] M. G. Martin and J. I. Siepmann, "Transferable potentials for phase equilibria. 1. united-atom description of n-alkanes," *The Journal of Physical Chemistry B*, vol. 102, no. 14, pp. 2569–2577, 1998.

[38] H. A. Karimi-Varzaneh, H.-J. Qian, X. Chen, P. Carbone, and F. Müller-Plathe, "IBIsCO: A molecular dynamics simulation package for coarse-grained simulation," *Journal of Computational Chemistry*, vol. 32, no. 7, pp. 1475–1487, 2011.

[39] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. Di Nola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," *The Journal of Chemical Physics*, vol. 81, no. 8, pp. 3684–3690, 1984.

[40] B. Dünweg and K. Kremer, "Molecular dynamics simulation of a polymer chain in solution," *The Journal of Chemical Physics*, vol. 99, no. 9, pp. 6983–6997, 1993.

[41] C. Y. Liang and S. Krimm, "Infrared spectra of high polymers. VI. Polystyrene," *Journal of Polymer Science*, vol. 27, no. 115, pp. 241–254, 1958.

Chapter 7

Programming Dual scale MD

7.1 Introduction

The basis of the research work presented in this thesis is the development of novel algorithms for MD. An algorithm is a set of instructions that allow a certain procedure to be carried out. In more practical terms, an algorithm is the conversion of mathematical formulae into the code which can be executed by computers. For MD a single algorithm will do a task such as moving particles or calculating force, while the complete collection of connected algorithms is what forms the program as a whole. For the work presented in Chapters 5 and 6, customised code had to be written, and the aim of this Chapter is to explain this customised code and how it was incorporated into an existing MD program.

This Chapter does not aim to reiterate how to write basic simulation code, for which there are already many good resources including Allen and Tildesley[1], Frenkel and Smit[2], and for a more modern approach "The Art of Molecular Dynamics Simulation"[3]. Ultimately however, the most up to date examples will be found in the source code repositories for actively developed projects. At the time of writing, good examples of open source MD programs with active communities include Espresso++[4], Lammps[5] and Gromacs[6].

The starting point for the program used in Chapters 5 and 6 was the IBIsCO simulation program[7]. This was chosen because of its ability to handle tabulated potentials well, necessary for the use of the IBI potentials in dual scale models, and because of the simplicity of the code. Most established MD programs have upwards of 100,000 lines of code, representing the multitude of various options in these programs. The IBIsCO code however started at approximately 5,000 lines of code, making it much more accessible as a starting point for experimenting with new methods.

This Chapter is arranged in three sections, Section 7.2 details how the data

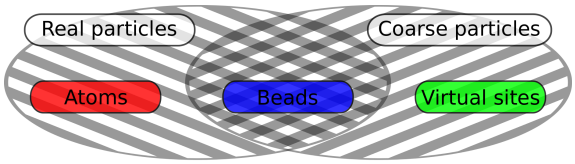


Figure 7.1: Venn diagram of the different types of particles in dual scale simulations

structures in the program were modified to accomodate the dual scale methodology, Section 7.3 then explains the flow of the program as a whole and finally Section 7.4 shows how this was expanded to implement the multiple time step scheme. For completeness, the full listing of the code in the version of IBIsCO used in this Thesis is given in Appendix D. The code presented is written in the Fortran 95 language, for which good sources on include References [8] and [9].

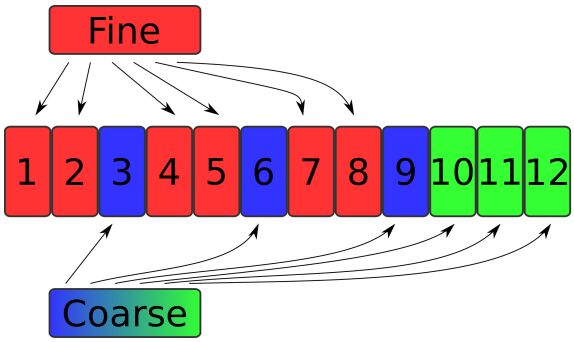
7.2 Data structures for dual scale MD

As described in Chapter 5, within the dual scale simulations there are three distinct sets of particles: atoms, beads and virtual sites (VS). As shown in Figure 7.1 these can be grouped into two further sets, the atoms and beads are "real" particles, and the beads and VS are coarse particles.

For all of these particles, various data exist, such as position, velocity, force and other data related to the particles, all of which are stored in many arrays collectively known as the data structure. It will be necessary to iterate over the data for all of these sets individually during the various MD algorithms. Therefore the data structures within the MD program need to be designed in a way to allow easy iteration over any set.

The solution used in this work is shown in Figure 7.2a. At the start of the programs execution the data of the real particles is loaded into the data structures and then the VS are appended to this. Two further arrays called FINE and COARSE are created, containing a series of addresses to every particle in their set. These arrays are referred to as pointer arrays as on their own they hold no information, but instead give directions to the locations of other data.

For example, for the data in Figure 7.2a, the FINE array would contain the values $\{1, 2, 4, 5, 7, 8\}$. To iterate over all the data for the atom set, one must use a do loop iterating over the number of atoms, then use the corresponding value in the FINE array to get the address to the atom. This is shown in Figure 7.2b, with the index of atom I being found in the pointer array at FINE(I).



(a) Visualising the pointer arrays to access fine and coarse particles

```
DO I=1, NATOMS
    J = FINE(I)
    ATOM_POSITION = RXYZ(J)
    ! Now calculate on the Ith Atom
END DO
```

(b) Using a pointer array to iterate over a subset.

Figure 7.2: The use of pointer arrays to access subsets of particles

7.3 The dual scale MD loop

With our data structures set up, we can turn our attention to calculating Equation 7.1 as originally defined in Chapter 4.

$$\mathbf{F}_i = \sum_{\beta} \left(\sum_{j \in \beta} \lambda_{\alpha\beta} \mathbf{F}_{ij}^{atomistic}(\mathbf{r}_{ij}) + \frac{m_i}{M_\alpha}(1 - \lambda_{\alpha\beta}) \mathbf{F}_{\alpha\beta}^{CG}(\mathbf{R}_{\alpha\beta}) \right) \tag{7.1}$$

We can perform this summation in any order, and therefore choose the most computationally efficient route. Rather than calculate $\mathbf{F}_{\alpha\beta}$ for each atom and transfer a fraction of this to the atom, we can instead calculate all pairwise VS forces and then distribute only the net force on VS onto the underlying atoms.

Figure 7.3 shows a comparison of simplified versions of a single scale and dual scale simulation MD loop. Using the pointer arrays we can easily treat the atomistic and CG nonbonded interactions separately, which has two main benefits. Firstly, these can use different settings within their respective subroutines, for things such as update frequency and cutoff range. This allows the fine tuning of these algorithms. Secondly, by using the same subroutines but with different arguments, the code is more easily maintained and debugged.

7.3.1 Working with virtual sites

Within the modified MD loop presented, there are also various algorithms for manipulating the properties of the VS in the system, namely defining their position

```
DO STEP = 1, NSTEP

   ! Update the neighbourlist for atoms?
   IF (MOD(STEP, ATOM_NEBUPDATE) .EQ. 0) THEN
      CALL UPDATE_NEIGHBOURLIST(NATOMS, FINE, &
                                LIST_ATOM, RCUT_ATOM)
   END IF



   ! Calculate nonbonded force on all atoms
   CALL NONBONDED_FORCE(NATOMS, FINE, &
                        LIST_ATOM, RCUT_ATOM)



   CALL BONDED_FORCE() ! Calculate bonded force

   CALL MOVE()         ! Move all atoms
END DO
```

(a) Standard MD loop

```
DO STEP = 1, NSTEP
   CALL VIRTUAL_DEF() ! Define the positions of VS
   ! Update the neighbourlist of each set?
   IF (MOD(STEP, ATOM_NEBUPDATE) .EQ. 0) THEN
      CALL UPDATE_NEIGHBOURLIST(NATOMS, FINE, &
                                LIST_ATOM, RCUT_ATOM)
   END IF
   IF (MOD(STEP, BEAD_NEBUPDATE) .EQ. 0) THEN
      CALL UPDATE_NEIGHBOURLIST(NCOARSE, COARSE, &
                                LIST_BEAD, RCUT_BEAD)
   END IF
   ! Calculate nonbonded force on each set
   CALL NONBONDED_FORCE(NATOMS, FINE, &
                        LIST_ATOM, RCUT_ATOM)
   CALL NONBONDED_FORCE(NCOARSE, COARSE, &
                        LIST_BEAD, RCUT_BEAD)
   CALL DISTRIBUTE_VSFORCE() ! Move the nonbonded force off VS
   CALL BONDED_FORCE()       ! Calculate bonded force
   CALL DISTRIBUTE_VSFORCE() ! Move the bonded force off VS
   CALL MOVE()               ! Move all real particles
END DO
```

(b) Dual scale MD loop

Figure 7.3: A comparison of the execution of a regular and dual scale MD program.

```
DO I=1, NVIRTA
  VS_POS = I + NREAL
  TI = ITYPE(VS_POS)
  SUMTOTXYZ(:) = 0.0   ! 3D COM
  DO A=1, VIRT_NUMATOMS(TI)   ! Loop over all the atoms within this VS
    J = VIRT_ATM_IND(I, A)
    TJ = ITYPE(J)
    SUMTOTXYZ(:) = SUMTOTXYZ(:) + VIRT_MASSCOEFF(TI, TJ) * RXYZ(:, J)
  END DO
  RXYZ(:, VS_POS) = SUMTOTXYZ(:)
END DO
```

(a) A subroutine for defining the positions of virtual sites.

```
DO I=1, NVIRTA  ! Loop over all VS
  VS_POS = NREAL + I  ! VS are appended, so find the VS here
  TI = ITYPE(VS_POS)
  DO A=1, VIRT_NUMATOMS(TI)  ! Loop over atoms in this VS
    J = VIRT_ATM_IND(I, A)
    TJ = ITYPE(J)
    FXYZ(:, J) = FXYZ(:, J) + VIRT_MASSCOEFF(TI, TJ) * FXYZ(:, VS_POS)
  END DO
  !Reset force on virtual site to 0 once distributed
  FXYZ(:, VS_POS) = 0.0
END DO
```

(b) A subroutine to distribute force from virtual sites to atoms.

Figure 7.4: Example subroutines for dealing with Virtual Sites

and reassigning the net force on them. These are shown in Figure 7.4. The VS are iterated over using the fact that they were appended onto the real particles, so the Ith VS is found at (NREAL + I) where NREAL is the number of real particles. Each VS keeps a record of the atoms it contains, here called VIRT_ATM_IND.

An example subroutine for calculating the position of virtual sites is given in Figure 7.4a, and is an implementation of Equation 7.2.

$$\mathbf{R}_\alpha = \frac{1}{M_\alpha} \sum_{i \in \alpha} m_i \mathbf{r}_i \tag{7.2a}$$

$$M_\alpha \equiv \sum_{i \in \alpha} m_i \tag{7.2b}$$

In this, the position of each VS is defined as the center of mass of its constituent atoms, the factor m_i/M_α is stored as the variable VIRT_MASSCOEFF. It is important to make sure that the positions of the atoms used to calculate the center of mass are all within the same periodic image as each other. This subroutine needs to be called before any calculations involving the VS, but after the atoms have moved to their new position for this timestep. In Figure 7.3 this is done immediately at the

start of each step in the subroutine VIRTUAL_DEF.

Figure 7.4b shows a subroutine for distributing the force on a VS onto the underlying atoms, which happens on a mass weighted basis. This algorithm performs the second term of the right hand side in Equation 7.1 once the net force on VS has been calculated.

7.4 Implementing a multiple timestep scheme

The multiple timestep (MTS) scheme described in Chapter 6 was built upon the dual scale MD loop described in Section 7.3. This Section outlines the workings of this scheme, with the full code given in Appendix D.7. The existing design of the dual scale MD loop is ideal as the CG and atomistic nonbonded force calculation steps are already separated. This allows the approximation of the CG nonbonded force in some steps through wrapping the subroutine in a SELECT CASE statement, shown in Figure 7.5.

For this code block, the user defines a variable called NMTS which states how many steps to approximate the CG force for. The choice of whether to approximate forces is then controlled by feeding the result of a MOD function into the SELECT CASE construct. When the remainder of the time step number (called STEPNO) divided by (NMTS + 3) is 1, 2 or 3 the nonbonded force is calculated explicitly, and for all other values an approximation is used. It can be seen that if NMTS is zero, the original dual scale MD loop is recovered, as the explicit step branch is always entered.

In the explicitly calculated steps, the force is calculated as normal and then stored in an intermediate array, called MTS_FXYZ, which is sized to hold the force vector from every CG particle for three steps. In the approximated steps, the for-

```
MTS_MOD = MOD(STEPNO, (3 + NMTS))
SELECT CASE(MTS_MOD)
    CASE(1, 2, 3) ! Explicit step branch
        CALL NONBONDED_FORCE(NCOARSE, BEAD, MAXNAB_BEAD, &
                             LIST_BEAD, RCUT_BEAD, RCUTSQ_BEAD)
        CALL MTS_SAVEFORCE(MTS_MOD, BEAD, FXYZ, NITEMS, &
                           MTS_FXYZ, NCOARSE)
        CALL MTS_SAVEAUXS(MTS_MOD)
    CASE DEFAULT ! Approximation branch
        CALL MTS_APPROX(MTS_MOD, BEAD, FXYZ, NITEMS, MTS_FXYZ, NCOARSE)
        CALL MTS_LOADAUXS(MTS_MOD)
END SELECT
```

Figure 7.5: Algorithm for switching between saving and approximating CG forces in an MTS scheme.

ward approximation is generated in the subroutine MTS_APPROX which replaces the usual, and much more computationally intensive, nonbonded force calculation. Various other variables such as the pressure tensor can also be stored and approximated in a similar fashion, here done in subroutines called MTS_SAVEAUXS and MTS_LOADAUXS.

The subroutines for recording and approximating the forces are shown in Figure 7.6. These turn out to be very simple through the use of the pointer arrays explained in Section 7.2. Overall, this formulation of the MTS scheme allows for the number of approximated steps to be varied easily as was presented in the paper, but also for different order approximation techniques to be tried.

7.5 Conclusion

As has been shown, with a modern modular design to the original code, it is easy to expand this to handle dual scale MD. The algorithms presented are not designed for optimal performance and could be improved once this is required. It is important to remember that any such optimisations may well make the detection of errors and further modifications to the methodology more difficult, and therefore should only be pursued once it becomes necessary.

```
SUBROUTINE MTS_SAVEFORCE(I, BEAD, FXYZ, NITEMS, &
                         MTS_FXYZ, NCOARSE)
  IMPLICIT NONE

  INTEGER, INTENT(IN) :: I !< Mod of current MTS step
  INTEGER, INTENT(IN) :: NITEMS, NCOARSE
  REAL*4, DIMENSION(3, NITEMS), INTENT(IN) :: FXYZ
  REAL*4, DIMENSION(3, 3, NCOARSE), INTENT(INOUT) :: MTS_FXYZ
  INTEGER, DIMENSION(NCOARSE), INTENT(IN) :: BEAD
  INTEGER :: A, ATOM_ID

  DO A = 1, NCOARSE
    ATOM_ID = BEAD(A)
    MTS_FXYZ(:,I,A) = FXYZ(:,ATOM_ID)
  END DO

  RETURN
END SUBROUTINE MTS_SAVEFORCE
```

(a) A subroutine for saving the CG forces

```
SUBROUTINE MTS_APPROX(I, BEAD, FXYZ, NITEMS, MTS_FXYZ, NCOARSE)
  IMPLICIT NONE

  INTEGER, INTENT(IN) :: I ! Mod of current MTS step
  INTEGER, INTENT(IN) :: NITEMS, NCOARSE
  REAL*4, DIMENSION(3, NITEMS), INTENT(INOUT) :: FXYZ
  REAL*4, DIMENSION(3, 3, NCOARSE), INTENT(IN) :: MTS_FXYZ
  INTEGER, DIMENSION(NCOARSE), INTENT(IN) :: BEAD
  INTEGER :: A, ATOM_ID

  DO A = 1, NCOARSE
    ATOM_ID = BEAD(A)
    ! Forwards approx
    FXYZ(:,ATOM_ID) = MTS_FXYZ(:,3,A) + &
          (I-3) * 0.5 * (MTS_FXYZ(:,3,A) - MTS_FXYZ(:,1,A))
  END DO

  RETURN
END SUBROUTINE MTS_APPROX
```

(b) A subroutine for approximating CG forces

Figure 7.6: Example subroutines for saving and approximating forces in an MTS scheme.

Bibliography

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*. New York, NY, USA: Clarendon Press, 1989.

[2] D. Frenkel and B. Smit, *Understanding Molecular Simulation*. Orlando, FL, USA: Academic Press, Inc., 2nd ed., 2001.

[3] D. C. Rapaport, *The Art of Molecular Dynamics Simulation*. New York, NY, USA: Cambridge University Press, 2nd ed., 2004.

[4] J. D. Halverson, T. Brandes, O. Lenz, A. Arnold, S. Bevc, V. Starchenko, K. Kremer, T. Stuehn, and D. Reith, "ESPResSo++: A modern multiscale simulation package for soft matter systems," *Computer Physics Communications*, vol. 184, no. 4, pp. 1129–1149, 2013.

[5] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1 – 19, 1995.

[6] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 12, pp. 19–25, 2015.

[7] H. A. Karimi-Varzaneh, H.-J. Qian, X. Chen, P. Carbone, and F. Müller-Plathe, "IBIsCO: A molecular dynamics simulation package for coarse-grained simulation," *Journal of Computational Chemistry*, vol. 32, no. 7, pp. 1475–1487, 2011.

[8] S. J. Chapman, *Fortran 95/2003 for scientists and engineers*. Boston [u.a.]: McGraw-Hill, 3. ed. ed., 2008.

[9] M. Metcalf, J. K. Reid, and M. Cohen, *Fortran 95/2003 explained*. Numerical mathematics and scientific computation, Oxford, New York: Oxford University Press, 2004.

Chapter 8

Parallel programming in MD

8.1 Introduction

As has already been highlighted numerous times, the quality of results from computational research such as molecular dynamics can always be improved through further sampling. On a more practical level, research is always limited by the resources that are available including both computer time for performing simulation and deadlines for when research must be completed. It is therefore the responsibility of the computational researcher to maximise the usage of the available computational resources. For this, research software must react to innovations in hardware and software and the subject of this Chapter is one such innovation which has transformed scientific computing over the last fifteen years, parallel programming.

In order to understand parallel programming first the terminology for the hardware involved must be briefly explained. Each single computer is called a node and many of these can be connected on a network to form a cluster. Calculations on a computer are performed by the CPU core, and on a single node there are one or more CPU cores, from up to 4 on a modern desktop computer to up to 24 on the highest performance computational nodes. Finally, a node has a certain amount of memory shared between all cores on a node for temporarily storing data while programs are in operation.

Whilst computer hardware is constantly evolving to become more powerful, to properly utilise this power we need to appreciate how this power is made available. Early advances in computer power until around the year 2000 were governed by Moore's Law[1], which predicted the doubling of transistor counts on a single computer core every 18 months. More recently however, physical limitations on the miniaturisation of electronics has made further progress infeasible and forced manufacturers to improve computers by allowing many cores to interconnect in a single node[2]. This means that rather than their ability to sequentially solve many

calculations quickly, computers have become more powerful through their ability to solve many calculations simultaneously. Parallel programming therefore refers to the application of this new hardware to performing algorithms in parallel, decreasing the total runtime of a program.

8.1.1 Amdahls Law

The theoretical speed-up, S, that is possible when executing a program using n_c cores is given by Amdahls law, Equation 8.1:

$$S\left(n_c\right) = \frac{1}{(1-p) + \frac{p}{n_c}} \tag{8.1}$$

Where p is the fraction of the program's runtime in serial that is executed in parallel. This can be reformulated to reveal what is the maximum possible speed up for a given program given enough computational resources.

$$\lim_{n_c \to \infty} S = \frac{1}{1-p} \tag{8.2}$$

This shows that it is important for all algorithms in a program be parallelisable, as they will quickly become bottlenecks. Historically, this has led to algorithms falling out of favour, for example, the SHAKE algorithm for constraining bond lengths was unsuited to parallelism[3] which led to the p–LINCS algorithm becoming used instead[4].

8.2 Parallel software libraries

The introduction of parallel hardware has then given rise to various new software technologies which are designed to allow this new hardware to be used. Programming languages are designed to allow the expression of how an algorithm is to be performed by a computer. Extensions to these languages, called libraries, then allow the programmer to express not just how an algorithm functions, but how it can be solved in parallel. In the rest of this Chapter, three of these libraries, OpenMP, MPI and CUDA, will be summarised and their applications to MD programs reviewed. The examples given in this Chapter are written in the Fortran language, however these libraries are available in most similar languages including C and C++.

8.2.1 OpenMP

OpenMP is an extension to programming languages that allows portions of code to be marked as parallel[5]. In Fortran this is done through bracketing DO loops with

```
!$OMP PARALLEL DO
DO I = 1, N
    CALL FUNCTION(I)
END DO
!$OMP END PARALLEL DO
```

Figure 8.1: Example OpenMP directives around a DO loop

!$OMP markers which are read as special instructions to the compiler: this is shown in Figure 8.1. In this example, the program will proceed in serial until the start of the loop, at which point more cores will join in. The different iterations of the loop will be completed by many cores, and once all iterations have been completed the program reverts to being serial. This is called the fork-join model as the number of cores at work changes throughout the program.

OpenMP is a shared memory paradigm, meaning that once more cores have been added they have access to the same piece of memory, so they share the same view of the current state of the program. This means that all cores must physically be on the same node putting an upper limit on the number of cores that can be used with this method.

MD code is full of loops over the particles, and therefore is very well suited to using OpenMP[6]. As many cores are working on the same piece of memory, it is possible that their memory access conflicts with one another causing numerical errors. This is called race conditions, as different cores can be thought to race to access a piece of memory. However it is possible to rewrite algorithms so that no conflicts occur, with minimal performance penalties[7].

As the modifications to the code are technically comments, it is possible to compile the code for serial usage without changing the code. This makes debugging any issues caused by parallelisation trivial, as a serial version of the program can be generated to compare against.

8.2.2 MPI

Another language extension for parallelisation is message passing interface or MPI[8]. MPI works by creating various instances of the same program, known as tasks, running on different cores. MPI is a distributed memory paradigm, so each task's memory is private and they each have their own unique set of data to operate on. Tasks can only communicate with each other through explicit calls in the code.

With reference to Amdahls law, MPI allows the entirety of code to run in parallel rather than just a section. As each MPI task has its own memory, it can take place on hardware in different physical locations meaning that there is no upper limit on the number of cores an MPI program can use.

```
CALL FUNCTION(A)

CALL MPI_SEND(A, 100, MPI_DOUBLE_PRECISION, DEST, TAG, COMM, IERR)

CALL MPI_RECV(B, 100, MPI_DOUBLE_PRECISION, DEST, TAG, COMM, IERR)

CALL FUNCTION(B)
```

Figure 8.2: An example usage of MPI to send and receive data. In this example, each task

The most common usage of MPI for MD is domain decomposition, where the system volume is divided between MPI tasks. Communication is required when particles move between domains and to communicate particles that are on the edge of each domain to the neighbouring domains[9, 10]. Apart from the extra communication, all other algorithms can work in the same way as the serial versions. In addition to domain decomposition, it is possible to divide the calculation of different parts of the force field between different MPI tasks. For example the calculation of electrostatic interactions can be performed independently of the dispersive forces[11]. As the execution of the program can only go as fast as the slowest part, it is important to try and balance the computational work between different tasks. Dynamic load balancing algorithms have been created which look to reassign the division of work on the fly to address this problem[12].

8.2.3 CUDA

A graphics processing unit (GPU) is an additional piece of hardware for a computer which specialises in parallel processing. These were originally designed for the 3D graphics in video games, however more recently they have become popular for high performance computing in scientific programming. CUDA is a adaptation of the C programming language designed for writing code specifically for GPUs[13].

When writing code with CUDA, some subroutines can be written to be executed on the GPU, allowing the CPU to pass work to the GPU to do. Programming for GPUs is a mixture of the two previously explained technologies, with a mixture of shared and distributed memory available for use on a GPU. GPU hardware is structured vastly different to CPU hardware, most notably by the number of individual processors available to perform mathematical operations. Whilst a single CPU node contains up to 24 cores, a single GPU will have 1000s of processors available. This means to tailor an algorithm for a GPU, it needs to be heavily rewritten to split the work into many more separate pieces than is required for CPUs.

For the purposes of MD, GPUs have been identified as an extremely powerful option for the future[14], and have become increasing common in the last ten years

with implementations in most major MD packages. A single GPU has been shown to have the power of hundreds of CPU cores[15], and have also been shown to be very cost efficient compared to CPUs[16]. The performance of GPUs is highly volatile however, and MD algorithms have to be tuned to the specific hardware that is used to get the best performance[16]. With GPU technology relatively new and continually advancing, this makes finding the best performance a moving target.

Unlike the other options mentioned so far CUDA is not an open standard, it is developed by the Nvidia corporation specifically for hardware that they produce and sell. Therefore with any code developed in the CUDA language there is the risk of vendor lock in, making moving away from using CUDA difficult. There are open source alternative such as OpenCL[17], however these are not as well supported. Finally, debugging code written in CUDA is very difficult as the code is executed on a separate piece of hardware, and tracing errors that occur on the GPU is difficult.

8.2.4 Mixed approaches

All of the options reviewed so far have their strengths and weaknesses, generally caused by limitations in the hardware. It then follows that various combinations of the technologies have been tried together to circumvent these limitations. OpenMP offers the most efficient use of CPU cores, but is limited to a single node by the shared memory approach. Therefore MPI has been used to bridge across nodes while using OpenMP within a single node, to extract the maximum possible efficiency[18, 19] The number of GPUs that can be attached onto a single node is limited to 4, therefore MPI has again been used to bridge nodes and allow many GPUs to be used[20]. Finally, executing code on a GPU does not prevent the CPU from doing any work in the meantime. The asynchronous operation of CPU and GPU can be exploited to allow them to work in parallel[21, 22].

8.3 Application to IBIsCO

Ultimately, the work presented in Chapters 5 and 6 was done using an OpenMP solution, with a full listing given in Appendix D. Although CUDA would provide the best performance and value for money, it was decided to not pursue this option for two reasons. Firstly, writing CUDA code is much more complex than OpenMP code, and as this research was based on rapidly prototyping different ideas, it is important that code can be written quickly. Secondly, as some code is executed on a separate piece of hardware, it is much more difficult to debug and find where errors have occured. This is important because the work involves dealing with potentially unstable algorithms. MPI was not chosen because of similar debugging issues as

CUDA, where data would be spread over many nodes, and because of the extra complexity the communication would cause. As the position of VS particles in a dual scale model are determined by the atoms, domain decomposition would require two separate instances of communication, once to communicate the positions of all atoms, and again to communicate the calculated positions of VS. Finally, an OpenMP solution could make use of all the resources that were available, making a more complicated solution unnecessary.

Bibliography

[1] G. E. Moore, "Readings in computer architecture," ch. Cramming More Components Onto Integrated Circuits, pp. 56–59, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000.

[2] J. D. Meindl, "Beyond Moore's Law: The interconnect era," *Computing in Science & Engineering*, vol. 5, no. 1, pp. 20–24, 2003.

[3] R. Elber, A. P. Ruymgaart, and B. Hess, "SHAKE parallelization," *The European Physical Journal Special Topics*, vol. 200, no. 1, 2011.

[4] B. Hess, "P-LINCS: a parallel linear constraint solver for molecular simulation," *Journal of Chemical Theory and Computation*, vol. 4, no. 1, pp. 116–122, 2008.

[5] OpenMP Architecture Review Board, "OpenMP application program interface version 3.0," 2008.

[6] R. Couturier and C. Chipot, "Parallel molecular dynamics using OpenMP on a shared memory machine," *Computer Physics Communications*, vol. 124, no. 1, pp. 49–59, 2000.

[7] K. B. Tarmyshov and F. Müller-Plathe, "Parallelizing a molecular dynamics algorithm on a multiprocessor workstation using OpenMP," *Journal of Chemical Information and Modeling*, vol. 45, no. 6, pp. 1943–1952, 2005.

[8] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, (Budapest, Hungary), pp. 97–104, 2004.

[9] H. A. Karimi-Varzaneh, H.-J. Qian, X. Chen, P. Carbone, and F. Müller-Plathe, "IBIsCO: A molecular dynamics simulation package for coarse-grained simulation," *Journal of Computational Chemistry*, vol. 32, no. 7, pp. 1475–1487, 2011.

[10] I. T. Todorov, W. Smith, K. Trachenko, and M. T. Dove, "DL_POLY_3: new dimensions in molecular dynamics simulations via massive parallelism," *J. Mater. Chem.*, vol. 16, pp. 1911–1918, 2006.

[11] S. Páll, M. J. Abraham, C. Kutzner, B. Hess, and E. Lindahl, *Solving Software Challenges for Exascale: International Conference on Exascale Applications and Software, EASC 2014, Stockholm, Sweden, April 2-3, 2014, Revised Selected Papers*, ch. Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS, pp. 3–27. Cham: Springer International Publishing, 2015.

[12] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, "GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation," *Journal of Chemical Theory and Computation*, vol. 4, no. 3, pp. 435–447, 2008.

[13] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[14] J. E. Stone, D. J. Hardy, I. S. Ufimtsev, and K. Schulten, "Gpu-accelerated molecular modeling coming of age," *Journal of Molecular Graphics and Modelling*, vol. 29, no. 2, pp. 116–125, 2010.

[15] J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten, "Accelerating molecular modeling applications with graphics processors," *Journal of Computational Chemistry*, vol. 28, no. 16, pp. 2618–2640, 2007.

[16] J. A. Anderson, C. D. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *Journal of Computational Physics*, vol. 227, no. 10, pp. 5342–5359, 2008.

[17] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *IEEE Des. Test*, vol. 12, no. 3, pp. 66–73, 2010.

[18] A. Pal, A. Agarwala, S. Raha, and B. Bhattacharya, "Performance metrics in a hybrid mpiopenmp based molecular dynamics simulation with short-range interactions," *Journal of Parallel and Distributed Computing*, vol. 74, no. 3, pp. 2203–2214, 2014.

[19] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 12, pp. 19–25, 2015.

[20] J. Glaser, T. D. Nguyen, J. A. Anderson, P. Lui, F. Spiga, J. A. Millan, D. C. Morse, and S. C. Glotzer, "Strong scaling of general-purpose molecular dynamics simulations on GPUs," *Computer Physics Communications*, vol. 192, pp. 97–107, 2015.

[21] W. M. Brown, P. Wang, S. J. Plimpton, and A. N. Tharrington, "Implementing molecular dynamics on hybrid high performance computers short range forces," *Computer Physics Communications*, vol. 182, no. 4, pp. 898–911, 2011.

[22] W. M. Brown, A. Kohlmeyer, S. J. Plimpton, and A. N. Tharrington, "Implementing molecular dynamics on hybrid high performance computers particleparticle particle-mesh," *Computer Physics Communications*, vol. 183, no. 3, pp. 449–459, 2012.

Chapter 9

Revisiting the dual scale methodology: Octanol

9.1 Preface

This chapter represents the final paper in this thesis which is currently in the final stages of preparation for publication. One of the main motivations with this piece of work was to make performing this type of simulation more accessible. Previous work presented in this thesis used customised simulation code out of necessity, as the work focussed on experimentation on new algorithms. Instead with this work the aim was to see what was possible using only unmodified simulation code.

Whilst the simulation code used is standard the input files for these simulations are complicated and creating them is tedious. Therefore to simplify creating these and equally impotant, to allow these to be reproducible, a Python package was created to automate this process, the code for which is listed in Appendix F. This allows the conversion from a set of input files for an atomistic system along with an associated definition of the hybrid simulation, to the input files for a hybrid scale simulation, as shown in Figure 9.1.

A large departure from the rest of the work presented in this thesis is the use of

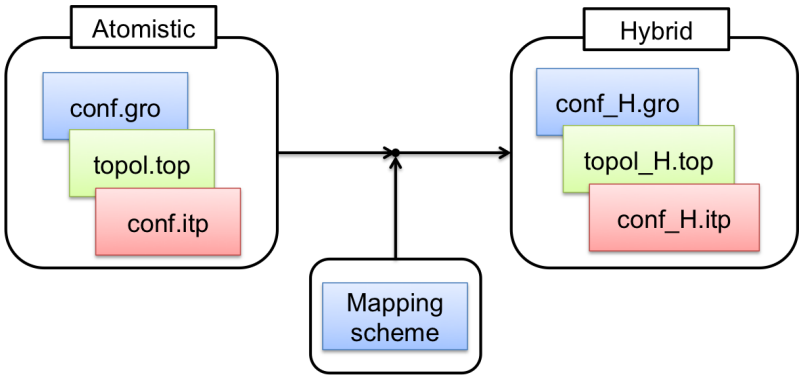


Figure 9.1: Automating the creation of hybrid input files

octanol as the system of interest, rather than a polymer. This was driven by another motivation for this work: to investigate the effectiveness of CG force fields derived specifically for hybrid models. Previous work had taken preexisting force fields and combined them to create the hybrid model, while in this work the CG force field is derived inside the hybrid model.

The polymers presented in previous chapters despite being massively important from both a research and industrial standpoint had the downside that due to their large size they have large characteristic times for rearrangement. This means that any feedback from changes in the force field is slow and therefore are unsuited for this work.

Octanol fulfills many criteria that polyamide met in the previous chapter, firstly it has hydrogen bonding as well as large aliphatic sections which require different levels of detail in the model. Compared to smaller alcohols, it is also large enough that the internal structure has enough degrees of freedom that there is room for error to occur should the new force field not be correct. Finally, as a freely moving liquid it will react to change in force field in a more timely manner.

The rest of this Chapter is presented in a paper format, but has not yet been published. The supporting information for this Chapter is given in Appendix C.

A systematic approach to dual scale models

Richard J. Gowers and Paola Carbone
School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester

Abstract

We present a methodology for constructing hybrid scale models where the intramolecular connectivity is modelled at atomistic detail while the majority of nonbonded interactions are modelled at the coarse level, with selective details retained. An automated tool for constructing the input files for such models is introduced, and then applied to both an all atom and a united atom model of octan-1-ol. The atomistic details in the OH head groups are retained at an atomistic level of detail while an iterative Boltzmann inversion CG forcefield is derived within the hybrid model to handle the coarser degrees of freedom.

The hybrid versions of each model are able to retain the hydrogen bonding behaviour, with the UA model showing slightly better results. Finally, the current computational performance of such a model is explored and compared against the potential theoretical performance.

9.2 Introduction

The use of coarse-graining (CG) techniques to reduce the degrees of freedom of a simulation model to access longer length and time scales in simulation has become an established and powerful tool in the modern simulator's toolbox. There are two schools of thought in how to derive parameters for these models. The first of these is using a "bottom up" approach, which comes from the well known result that there exists a unique potential to reproduce a given pairwise distribution[1]. These methods therefore try to optimise the resulting CG properties to those taken from reference atomistic simulations. A set of reference atomistic probability distributions can be optimised using the inverse Boltzmann iteration scheme[2, 3] and the inverse Monte Carlo method[4], Alternatively the potential of mean force between CG sites can directly be targetted using the force matching method[5]. The complimentary

approach to this is the "top down" approach, where it is instead attempted to try and match macroscopic quantities. Again a variety of target quantities have been optimised around, including density and interfacial tension[6], partition coefficients using Martini[7], and the predicted equation of state using SAFT-γ[8].

Many of these methods have matured to the stage that in recent years freely distributed software tools have been created to automate the process of generating the potentials, including such tools for Martini[9, 10], force matching[11], IBI[11, 12] and Inverse Monte Carlo[13]. These are incredibly valuable because of their ability to make using these methods easily available and reproduceable.

Despite the power of CG simulations, they are not without drawbacks. Charges within the system are often localised onto single atoms, making modelling them using CG beads difficult[14]. In particular the lack of hydrogen bonding has been identified as a deficiency in CG models[15, 16]. Secondly, CG approaches require that the resulting CG bonded degrees of freedom are independent from each other, and it is known to be difficult to find a mapping scheme where this is strictly true[17, 18]. The result of this is that often the configurational space explored by a molecule is often incorrect as while each individual angle is correct, when plotted against each other the two dimensional distribution is flattened[19, 20]. Finally, the results of any CG simulation inherently lack any atomistic details, and the process of reintroducing these details, known as backmapping, is a fundamentally challenging problem for even simple molecules[21, 22].

To try and build upon the strengths of CG models, various approaches have been proposed for trying to reincorporate some atomistic details back into coarser models. One approach is modelling the CG sites as both a position and a vector to indicate the orientation of the charge within the site, such as in the ELBA water model[23, 24]. Others have used a mixed description of both atoms and beads simulataneously to selectively choose which details to preserve, these models are usually called either hybrid or dual resolution models.

With two distinct levels of resolution in the system, interactions need to be define that cross this resolution boundary. This can be achieved through introducing extra forcefield parameters between the atoms and beads[25], or through the use of virtual sites (VS) to mediate any atom–bead interactions[26, 27]. In particular treating the electrostatics at an atomisic level in a hybrid model has been proved to be able to retain atomistic details[28] including hydrogen bonds[29].

9.3 Hybrid scale models

In this paper we describe a methodology for performing molecular dynamics (MD) simulations with molecules represented simultaneously at both fine-grained and

coarse-grained level of detail, with the overall objective being to accurately represent the system while minimising the computational cost. We will now briefly set out the theory behind such a model, present our automated system for creating these models, and then apply this to a system of octanol to test its effectiveness.

Performing a molecular dynamics simulation requires the iterative numerical calculation of the Hamiltonian[30]. The Hamiltonian for a system of N particles in the absence of any external field, can be defined according to equation 9.1[31]

$$\mathcal{H}\left(\mathbf{r}^N, \mathbf{p}^N\right) = \mathcal{K}\left(\mathbf{p}^N\right) + \mathcal{V}\left(\mathbf{r}^N\right) \tag{9.1}$$

Where $\mathbf{r}$ and $\mathbf{p}$ refers to the positions and momenta of the N particles, which give rise to $\mathcal{K}$ and $\mathcal{V}$, the kinetic and potential energy contributions respectively. The potential energy contribution can be expressed as the sum of nonbonded (V_{nb}) and bonded (V_{bonded}) contributions:

$$\mathcal{V} = V_{nb}\left(\mathbf{r}^N\right) + V_{bonded}\left(\mathbf{r}^N\right) \tag{9.2}$$

The most computationally expensive part of any MD simulation is the evaluation of nonbonded interactions[32, 33], usually causing around 90% of the simulation runtime. The computational cost of calculating V_{nb} scales with the number of interactions that must be calculated, meaning that not only large systems are more expensive to simulate, but also systems which have a high number density of particles.

A traditional approach to reducing the computational cost of simulation has been therefore to reduce the number of particles required to represent a given system. As a first step this can be done through combining hydrogens into the atoms they are bonded to, to create United-Atom (UA) models[34]. Going further, many heavy atoms can be combined into a single particle representation creating CG models. To do this, all atoms in the system must be mapped into a coarse-grained representation, allowing a description of the system in coarse-grained coordinates $\mathbf{R}^{N_C}$, where $N_C \ll N$. The center of the new coarse site is generally the center of mass of the constituent atoms[5], although other schemes have been proposed such as using the center of charge[35].

Typically the process of coarse-graining a model has been done before the start of the simulation, resulting in an irreversible loss in detail to the model. Instead, by reducing the degrees of freedom on the fly, the detailed representation is not permanently lost allowing the model to operate in either coarse-grained or fine-grained space. Reducing the degrees of freedom on the fly creates so called hybrid or dual-scale models. These feature a description of the molecular model in atomistic space and CG space at the same time, where again the positions in $\mathbf{R}$ space is

determined by a mapping scheme from the atoms in **r** space.

The particles in CG space are referred to as virtual sites (VS), and act as a proxy for allowing two groups of atoms to interact through a single pairwise potential[36]. Now the quantity V_{nb} can be solved using a mixture of coarse and fine descriptions. Importantly to avoid double counting of interactions, each VS can only interact with each other VS at a single level of resolution, so if a VS interacts with another VS at the coarse level then no fine-grained interactions may be computed between atoms belonging to the VS and vice versa. None of the bonded degrees of freedom have been coarse-grained as they are typically fast to calculate, but contain a large amount of structural information.

The potential energy in Equation 9.2 can now be represented with its nonbonded potential energy split into two contributions, one from coarse-grained space interactions and the other from fine-grained space interactions, shown in Equation 9.3

$$\mathcal{V} = V_{nb,fine}(\mathbf{r}^N) + V_{nb,coarse}(\mathbf{R}^{N_C}) + V_{bonded}(\mathbf{r}^N) \tag{9.3}$$

The kinetic energy contribution to the Hamiltonian is unchanged. As VS never themselves move, they have neither velocity nor mass.

9.3.1 Automating the workflow

Creating the input files to follow this scheme is non–trivial, and so to facilitate the process an automated tool has been created which takes a set of Gromacs input files for an atomistic system, along with a definition of the hybrid mapping, and generates new input files to run the hybrid scale simulation. The definition of the hybrid mapping is the only new file required for this process, and an example of this is given in Figure 9.2.

This uses the same syntax as other Gromacs inputs, and requires the beadtypes and hybriddef directives, followed by a mapping for each molecule type. The mapping scheme for each molecule can also be defined using the same mapping xml files as VOTCA[11], which are easily made using the online STOCK tool[37].

9.3.2 Hybrid octanol

The previously outlined methodoloy is now applied to the solvent octan-1-ol. Two separate atomistic models of octanol were chosen to serve as reference models, an all-atom (AA) model and a UA model.

Apart from the number of particles in each of these models, the main difference between them is the use of partial charges on the atoms. On the AA model, every atom has a small partial charge, whilst in the UA model only the oxygen and the

```
[ beadtypes ]
; Similar to Atomtypes, define the types of VS
X
Y
Z

[ hybriddef ]
; Define the interaction level between different
; types of beads
default coarse
X X     atomistic

[ OcOH ]
; Index, Type, Name, Indices
  1       X      X1     1   2   3   4   5   6   7   8
  2       Y      Y2     9  10  11  12  13  14  15  16  17
  3       Z      Z3    18  19  20  21  22  23  24  25  26  27
```

Figure 9.2: Example mapping file

hydrogen and carbon bonded to it carry any partial charge, leaving the majority of the molecule uncharged.

To define the VS the same mapping scheme was employed for both atomistic models, with the nine heavy atoms in octanol split into three equally sized CG sites. This mapping scheme is shown in Figure 9.3, with the three VS being named X, Y and Z, with X being the OH end of the octanol.

With each of the three VS in the molecule having different types, there are six different pairs of VS interactions. Of these, it was chosen to model X-X interactions at the fine level (ie atomistic), while all others were done at the coarse level. The final effect of this scheme is shown in Figure 9.3, where only the head group of each octanol molecule interact via many atomistic potentials, while all other intermolecular interactions are at the coarse level.

With all the intermolecular interactions defined, our attention turns to intramolecular interactions. Previous attempts at using bonds directly between fine and coarse particles gave poor results in preserving the structural integrity of the molecules[29]. Instead, and because bonded degrees of freedom are computationally cheap to calculate, the bonded degrees of freedom in the hybrid models will remain unchanged from the atomistic model.

Nonbonded interactions are required between particles still belonging to the same molecule but not directly bonded by either a bond, angle or torsion, and the coarse interactions from VS are unsuitable for these short ranged interactions as they are generally quite soft. Instead, atomistic level nonbonded interactions from the underlying atoms are used for all intramolecular interactions.

The result of this formulation of the force field means that in addition to an atom-

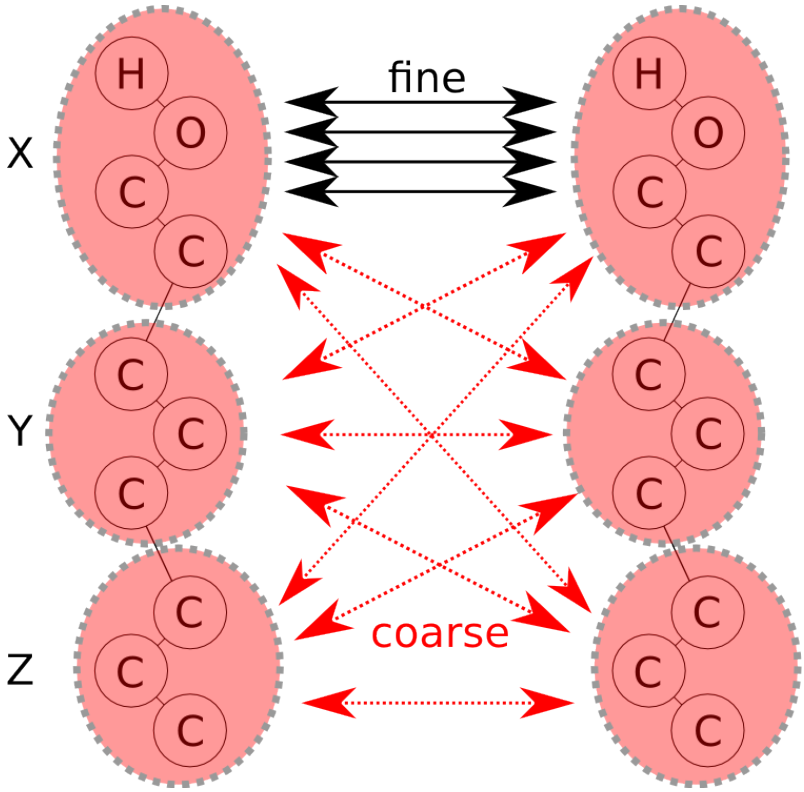


Figure 9.3: Mapping scheme for octanol and visualising the level of interaction between two adjacent octanol molecules. Atomistic sites and interactions are shown in solid black lines while coarse are shown in red and dashed.

istic force field, the only extra parameters required are any coarse-grained nonbonded interactions. These can be sourced from preexisting libraries of CG parameters, or as in this work, these parameters can be derived alongside the hybrid model.

9.3.3 Deriving CG potentials

In total five nonbonded potentials are required for the virtual sites, one for each VS pairing excluding X-X. These were derived using the iterative Boltzmann inversion (IBI) method[2]. In this method, the pairwise nonbonded potentials (V) are initially guessed according to a Boltzmann inversion of the pair correlation function $g_{ref}(r)$, shown in Equation 9.4.

$$V(r) \approx -k_B T \log\left(g(r)_{ref}\right) \tag{9.4}$$

Where k_B and T refer to the Boltzmann constant and temperature respectively. Note that the use of Boltzmann inversion makes the resulting potential tied to a particular thermodynamic point, however this is not always the case[38].

The structures generated from these initial potentials are generally quite poor, however they are refined by adjusting the potentials according to the difference between the reference pair distributions and those produced by the previous simulation

($g_i(r)$), to produce a new set of pairwise potentials (V_{i+1}), shown in Equation 9.5.

$$V_{i+1}(r) = V_i(r) + \alpha k_B T \log\left(\frac{g_i(r)}{g_{ref}(r)}\right) \tag{9.5}$$

Where α is a heuristic prefactor to moderate the change to subsequent potentials; in this work we used a value of $\alpha = 0.3$. If this prefactor is not used, the corrections to the potential are too large and the resulting potentials oscillate around the solution. The potentials must be iterated multiple times using this method, with the correction factor approaching zero as the potential converge.

The potentials generated from this method will not reproduce the correct pressure, therefore once the forcefield starts to converge on the target distributions, a pressure correct step is added. The pressure correction to the coarse potentials was a linear ramp, shown in Equation 9.6, as originally described by Reith and coworkers[2].

$$\Delta V_i(r) = \left(1 - \frac{r}{r_{cut}}\right) A_i \tag{9.6}$$

The parameter A was parameterised by considering the virial expansion of pressure (P), as shown in Equation 9.7, for both the current iteration i and the target pressure, as was originally done by Han and coworkers[39]. This adjustment to the potential corrected the pressure within a few iterations, but also disturbed the convergence of the pair potentials, requiring extra iterations to be performed.

$$P_i = \frac{\rho}{k_B T} - \frac{2\pi\rho^2}{3}\int_0^\infty r^3 \frac{dV(r)}{dr} g_i(r) dr \tag{9.7a}$$

$$P_{target} \approx \frac{\rho}{k_B T} - \frac{2\pi\rho^2}{3}\int_0^\infty r^3 \frac{d}{dr}\left(V(r) + \Delta V(r)\right) g_i(r) dr \tag{9.7b}$$

$$P_i - P_{target} = -A_i \left[\frac{2\pi\rho^2}{3 r_{cut}}\int_0^{r_{cut}} r^3 g_i(r) dr\right] \tag{9.7c}$$

Where ρ is the number density.

9.3.4 Simulation details

All simulations were conducted using version 5.0.4 of Gromacs[40]. The system size was 4096 molecules of octanol, simulated inside a cubic volume with length approximately 10 nm with periodic boundary conditions enforced on all sides.

Atomistic reference data was generated using the atomistic force fields detailed above. Parameters for the reference atomistic UA model came from Reference [41] while parameters for the reference AA model came from the OPLS-AA forcefield[34].

The parameters for these were found using Lipidbook[42]. The parameters for the CG part of the hybrid version of each model was derived using the IBI method detailed above.

A nonbonded interaction cutoff of 0.9 nm was used for the AA atomistic model, 1.2 nm for the UA atomistic model and 1.4 nm for the CG potentials in both models. In all models electrostatics were calculated using the particle-mesh Ewald (PME) method[43].

Production runs were conducted in an NPT ensemble, with the pressure maintained at 1 bar using an isotropic Berendsen barostat[44] with a compressibility of 4.5×10^{-5} bar^{-1} and a coupling time of 2 ps. The temperature was maintained at 298 K using the velocity rescale thermostat[45] with a coupling time of 0.1 ps. An integration timestep of 1 fs was used for all simulations, with 50 ns of simulated time.

9.4 Results

Unless otherwise state, all data was analysed using a combination of the MDAnalysis[46] and mdsynthesis[47] Python packages. Plots were created using matplotlib[48] and seaborn[49]

9.4.1 Structural results

Overall good agreement was able to be found using IBI after approximately 25 iterations. The final pair distributions for all VS combinations are shown in Figure 9.4, and are considered good results from an IBI procedure. For all pairings excluding X-X, the good agreement is a product of the IBI method where, however an important result from this is that the IBI procedure can still find a solution when used in a hybrid model.

It is important to notice the interactions between the X-X pairing come from the atomistic forcefield, and therefore these potentials were not optimised in any way, meaning that the good agreement shown is a result of the atomistic forcefield working well alongside the IBI potentials. In the AA version of the hybrid model, the agreement between the reference and final structure is worse, most notably at a distance of 0.4 nm where the reference peak was lower. This could be attributed to a lack of electrostatic screening caused by the removal of the partial charges in the coarse-grained tail, necessary as these degrees of freedom were removed.

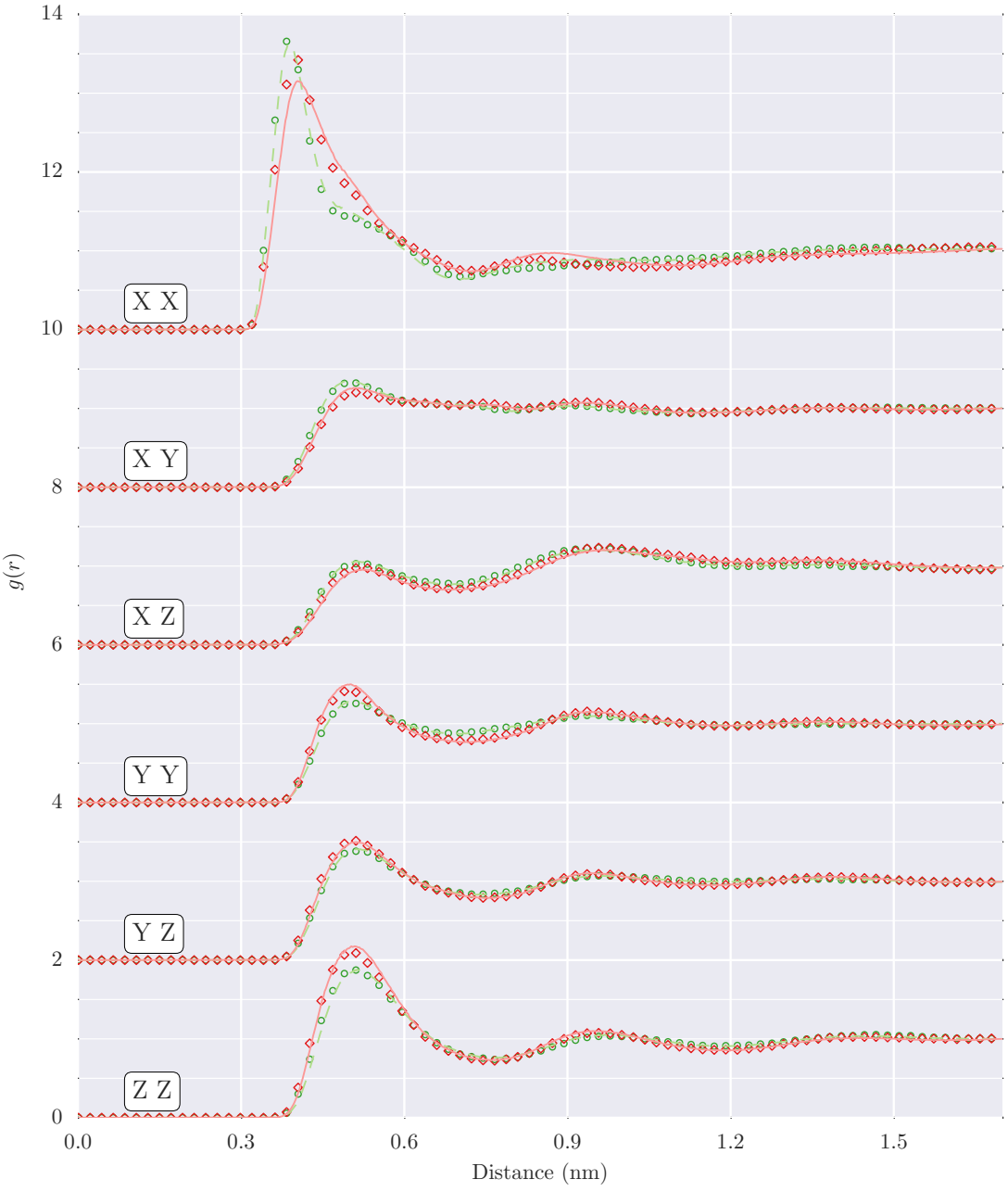


Figure 9.4: Comparison of pair distributions across the different models tested. Different pair distributions have been shifted by 2 units in the y axis. atomistic UA - dashed green line, hybrid UA - green circular markers, atomistic AA - solid red line, hybrid AA - red diamond markers

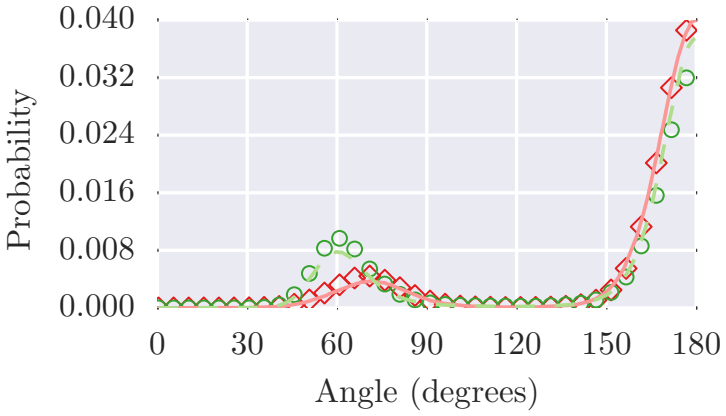


Figure 9.5: Comparison of the backbone dihedral angles in both models. 180° represents a trans configuration. Uses same legend as Figure 9.4.

Bonded parameters

To compare the efficacy of the atomistic bonds in the hybrid versions of each model, the average C–C–C–C dihedral angle was measured, shown in Figure 9.5. Both the hybrid models show good agreement with the atomistic model results, with the UA model being the worse of the two. The UA model has a slightly lower population of the trans configuration, indicating that the molecules are slightly more coiled compared to the reference atomistic version.

Hydrogen bonding details

The ability for the hybrid model to correctly represent atomistic details is paramount in assessing its performance, and the most important atomistic detail present in the system is the hydrogen bonding. Hydrogen bonds in the system can form between OH groups in different molecules, with a donor oxygen atom sharing its bonded hydrogen atom with an acceptor oxygen atom. These bonds can be characterised by their hydrogen-acceptor distance (r_{OH}) and the O–H–O angle (θ_{OHO}), giving a probability as a function of these two dimensions: P(r_{OH}, θ). To directly compare the atomistic and hybrid versions of each model directly, the probability of the hybrid model can be subtracted from the atomistic model, giving the difference in two dimensions. These probabilities are plotted in Figure 9.6.

Clearly visible in all models is the population of hydrogen bonds at just under 0.2 nm and around 170 degrees. At distances larger than 0.4 nm, most clearly in the AA model, there is a repeating chevron pattern, which is evidence of the extended structure in the system caused by hydrogen bonding. Considering the differences between the hybrid and the atomistic versions of each model, we can clearly see that the hybrid version of the AA model has slightly shorter hydrogen bonds. This

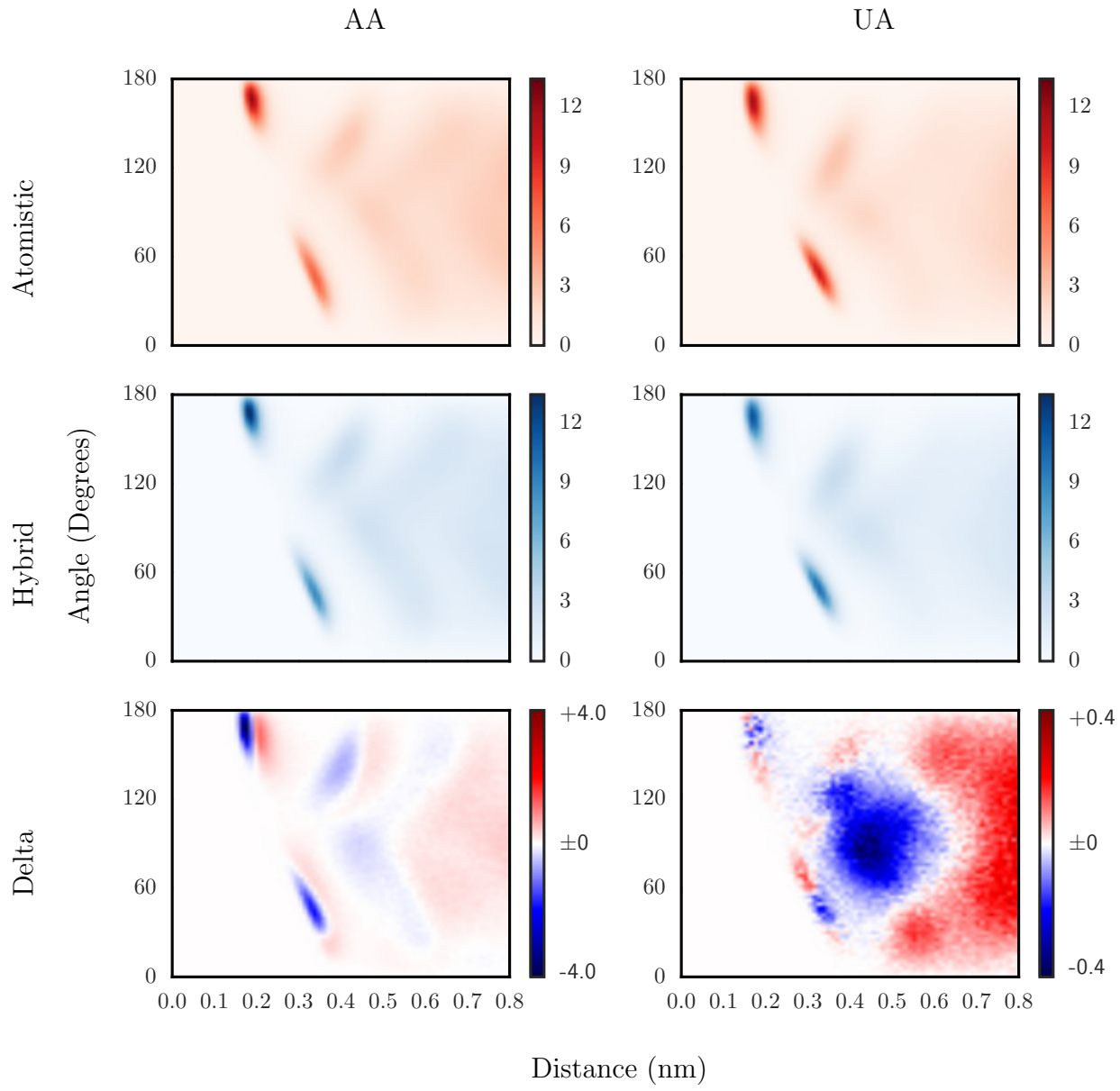


Figure 9.6: Comparison of the angle and length of potential hydrogen bonds in hybrid and atomistic models. The top panels show the atomistic results in red, the hybrid results are below in blue, while the bottom panels show the difference between these two. Red areas indicate a higher probability in the atomistic model while blue areas indicate a higher probability in the hybrid model. Note that the scale of the UA map is a tenth of the AA map.

effect is seen in the chevron pattern at larger distances, indicating that the entire hydrogen bond network is measurably different. For the UA version of the model, the differences in probability are 3% at their maximum, indicating that the UA hybrid model more accurately reproduces the hydrogen bond network than the AA hybrid model.

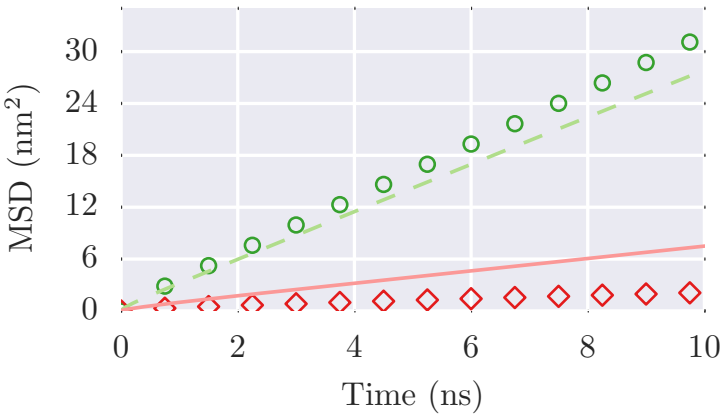


Figure 9.7: Mean squared displacement for all four models. Uses same legend as Figure 9.4

9.4.2 Dynamics results

Bulk diffusion

The diffusion constant of molecules, as defined according to the Einstein relation in Equation 9.8 was measured using the g_msd tool from Gromacs and is plotted in Figure 9.7.

$$D = \lim_{t\to\infty} \frac{\langle (\mathbf{r}(t) - \mathbf{r}(0))^2 \rangle}{6t} \tag{9.8}$$

The measured diffusivity was 0.1199, 0.0281, 0.4593 and 0.5324 $\times 10^{-5}$cm^{2}s^{-1} for the atomistic AA, hybrid AA, atomistic UA and hybrid UA models respectively. The result for the atomistic AA agrees well with previously published results as does the factor of nearly 4 between the UA and AA[50].

Typically CG models exhibit much faster diffusion, often by an order of magnitude, generally attributed to the flatter potential energy landscape the particles are moving through. These results however show that the hybrid UA model shows only a modest increase in its diffusion, while the hybrid AA model is diffusing nearly 5 times slower.

In optimising around target structures, the IBI coarse-graining procedure is attempting to numerically find the potential of mean force (PMF) between different VS centers, which should include the effects of these partial charges. Indeed, when the distribution of the magnitude of the force on each atom is compared between the atomistic and hybrid models, both the mean and also the distribution around the mean is perfectly matched, this is shown in the Supporting Information. This indicates that the CG force field is subjecting atoms to similar magnitudes of force, and this isn't the cause of the slowed diffusion.

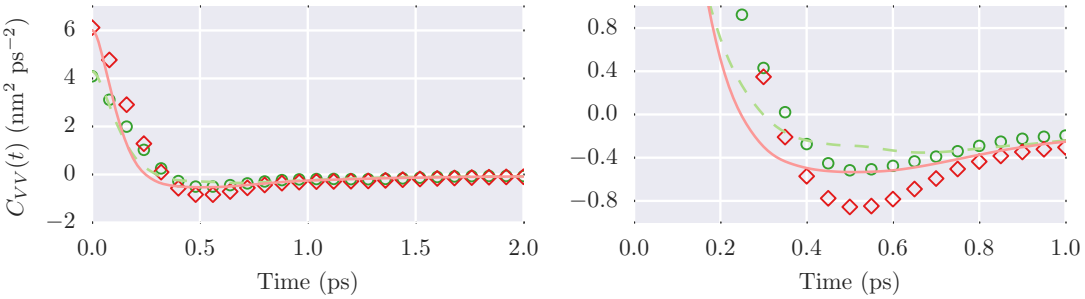


Figure 9.8: Molecule velocity autocorrelation. The right panel shows the detail of the backscattering area. Uses same legend as Figure 9.4.

Instead, we can look at the rate at which the velocity of molecules changes using the velocity autocorrelation function, shown in Equation 9.9. This was calculated for the velocity of each molecule, and is shown in Figure 9.8

$$\mathbf{v}_{CM}(t) = \frac{\sum m_i \mathbf{v}_i(t)}{\sum m_i} \tag{9.9a}$$

$$C_{VV}(t) = \langle \mathbf{v}_{CM}(t_0 + t) \cdot \mathbf{v}_{CM}(t_0) \rangle \tag{9.9b}$$

Where the angular brackets refer to an average over many starting times (t_0) and all molecules.

Noticeable is the larger negative well for the hybrid AA model at around 0.5 ps, caused by collisions with neighbouring molecules in the system. The deeper well indicates that when molecules in this model collide, their velocities are more reversed afterwards, i.e. collisions were less glancing in nature. The largest difference between the atomistic and hybrid AA models, besides the use of VS, is the removal of partial charges in the tail of the molecule. The removal of these relatively long range and pervasive partial charges could have caused the potential energy landscape to be sharper, decreasing the mobility of molecules. Indeed, the difference in diffusion does seem to be caused by the lack of partial charges; when the atomistic AA model is ran without partial charges, its diffusion falls to 0.0410 $\times 10^{-5}$ cm^{2}s^{-1}.

Hydrogen bond lifetimes

To evaluate the dynamics on a shorter lengthscale, the rate at which hydrogen bonds break can be measured. The time autocorrelation of the hydrogen bonds can be measured by monitoring the population of hydrogen bonds over time ($h_{ij}(t)$), shown in Equation 9.10. For the continuous definition of lifetime, once a given hydrogen bond has broken it cannot later reform. Hydrogen bonds were geometrically defined between a donor oxygen, hydrogen and an acceptor oxygen as having a hydrogen

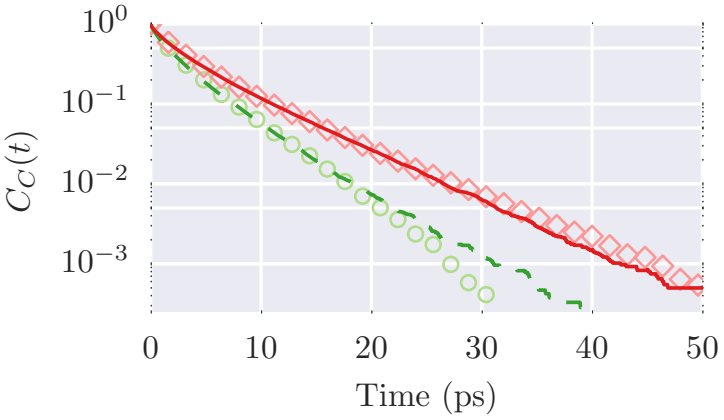


Figure 9.9: Comparison of the continuous hydrogen bond lifetime correlation. Note that the y axis is log scaled. Uses same legend as Figure 9.4.

acceptor distance of less than 0.3 nm and an O–H–O angle of greater than 130 degrees[51].

$$C_C(t) = \left\langle \frac{\sum h_{ij}(t_0) h_{ij}(t_0 + t)}{\sum h_{ij}(t_0)^2} \right\rangle \tag{9.10}$$

Simulations were performed under identical conditions to previously, but only for 250 ps, with trajectory frames being saved every 0.01 ps. The lifetimes of hydrogen bonds were then measured using the HydrogenBondAutoCorrel analysis module from MDAnalysis, this is plotted in Figure 9.9 The hydrogen bond lifetime can then be calculated through numerically integrating C_C, shown in Equation 9.11.

$$\tau_C = \int_0^\infty \left(C_C\left(t\right) - \left\langle C_C\left(t = \infty\right) \right\rangle \right) dt \tag{9.11}$$

This yields lifetimes of 4.26 and 4.32 ps for the atomistic and hybrid versions of the AA model, and 2.93 and 2.97 ps for the UA model. Unlike in previous work[29] where both the small and large scale dynamics had been affected, here the lifetimes are practically identical.

9.5 Computational cost of hybrid models

9.5.1 Theoretical performance

Now that the ability of these models to recreate atomistic results has been discussed, attention can now turn to the computational speed of these models. As stated in the introduction of this paper, one of the main motivations of hybrid models is to reduce the computational cost of performing simulation, allowing data to be collected at a faster rate. The total number of pairwise interactions that must be evaluated can

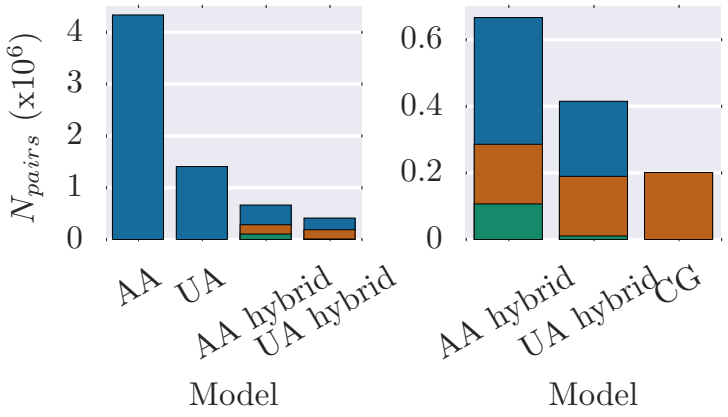


Figure 9.10: Comparison of the estimated number of nonbonded pairs in each model. Atomistic interactions are shown on top in blue, coarse interactions below in orange and finally intramolecular atomistic pairs in green.

be estimated using Equation 9.12, which assumes that the distribution of particles in the system is uniform. With pairwise interactions taking the majority of time in performing simulations[32], N_{pairs} can act as a rough estimate of the computational cost of a model.

$$N_{pairs} \approx \frac{4}{3}\pi \left(N_{atoms} \times \rho_{atoms} r_{cut,atoms}^3 + N_{VS} \times \rho_{VS} r_{cut,VS}^3 \right) + N_{intra} \tag{9.12}$$

Where N refers to the number of particles of each scale in the system, ρ refers to the number density, and r_{cut} refers to the interaction cutoff, and N_{intra} refers to the number of intramolecular pairwise interactions as described in Section 9.3.

The result of this is shown in Figure 9.10, and immediately apparently is the reduced computational cost of the hybrid versions of both the AA and UA models. The intramolecular pairs are only visible in the AA hybrid model, and represent 6.2% and 2.6% of the total number of pairs for the AA and UA models respectively. Finally, the right panel makes a comparison against a pure CG model, showing that the AA and UA models are roughly 2–3× the computational cost, despite accurately modelling the hydrogen bonding in the system.

9.5.2 Benchmarks of performance

The actual performance of running each of the models was measured using the built in benchmarking in version 5.0.6 of Gromacs[40]. Simulations were ran in serial to exclude any effects of the parallelisaton scheme. The results of this are shown in Figure 9.11, and it can be seen that the hybrid versions of each model are substantially slower than their fully atomistic counterparts.

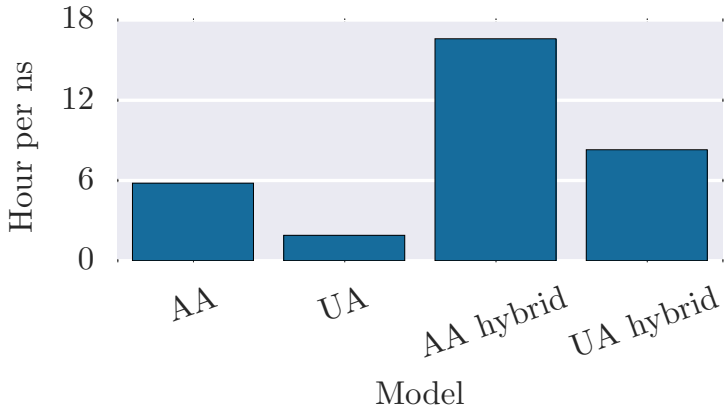


Figure 9.11: Comparison of the time required to calculate 1 ns of simulation. Higher values indicate slower execution time.

This discrepancy can be attributed to a number of compromises that have to be made in order to perform the simulations. Firstly, all nonbonded interactions had to use the larger coarse-grained potential cutoff due to limitations of the software, meaning that all atoms would have calculated far more nonbonded interactions than was strictly necessary. Following the approach by Wassenaar and coworkers[28] these were nullified through using a custom LJ table which was zeroed after the cutoff. The interactions from atoms inside a VS were "switched off" through putting their parameters to zero, however from the accounting of Flops/step it appears that these were still calculated, again leading to many meaningless calculations. Finally, as IBI potentials are a numerical potential, tabulated potentials had to be used, these are using an outdated "group" neighbour scheme in Gromacs rather than the newer verlet method.

Although the current performance of the hybrid model is actually slower than the fully atomistic versions, it is believed that after optimisations are made it should be possible to meet the theoretical performance outlined previously. Using a previous generation of a similar hybrid model running on bespoke code, we have shown that the theoretical scaling behaviour of hybrid models is possible[29]. In fact, once the position of the virtual sites has been determined, the pairwise interactions in the fine and coarse domains can be solved independently. This opens up extra possibilities for parallelisation through solving the two domains on different compute nodes, similar to how PME electrostatics are treated[40]. Some care is required however, as depending on the relative numbers of coarse and fine interactions, this may result in load imbalance between the two domains.

It should be remembered however that the CG model can use a larger integration timestep as the removal of the atomistic bonds greatly reduces the fastest frequency in the system. This means that directly comparing the computational cost per time

step is misleading as each step of the CG system represents more simulated time. One possibility for mitigating this factor slightly is the use of a multiple time step scheme[52, 53], where the interactions between VS could be evaluated at a reduced frequency, and indeed this has been shown to be possible for hybrid models[54].

9.6 Conclusion

We have presented an automated framework for preparing dual scale simulations for the Gromacs molecular dynamics package. This was applied to two atomistic models, both UA and AA, of octanol to create a hybrid model and an IBI force field was parameterised specifically for each hybrid model. The IBI procedure was conducted including the pressure correction iterations within the hybrid model and found a solution for both models without any problems.

Overall the resulting structural agreement between the hybrid and atomistic versions of each model were good. The hydrogen bonding structures, a product of the parts of the molecule which weren't included in the IBI fitting, were also quite well represented in the hybrid models, with the UA hybrid model showing better agreement. The bonded degrees of freedom, which were modelled using their original atomistic parameters inside the hybrid model, again showed good agreement against the atomistic models. Overall the structural results were very promising, and showed that the combination of IBI and atomistic parameters can work together well.

Short scale dynamic behaviour was correctly captured in both the hybrid models, however bulk diffusion was much slower in the hybrid version of the AA model, whilst approximately equal in the UA model. This discrepency in the AA model was attributed to the removal of partial charges from the aliphatical tail in the hybrid model. As has been identified by other research groups[28] the treatment of electrostatics in hybrid scale models remains an ongoing problem. In this case, despite good structural agreement, the dynamic behaviour was greatly changed by the removal of slight partial charges. It has been shown that applying a stochastic friction term can bring a CG model's dynamics into line with an atomistic model[55], and this approach could be used here in order to iron out these differences in dynamics.

With an automated system for creating such simulations, it is intended to try testing the limits of such models with larger and more complicated molecules. In particular, understanding how electrostatics can be treated in mixed resolution models is a key priority.

Acknowledgements

This work was funded by BBSRC grant BB/J014478/1. The authors would like to acknowledge the assistance given by IT Services and the use of the Computational Shared Facility at The University of Manchester.

Bibliography

[1] R. L. Henderson, "A uniqueness theorem for fluid pair correlation functions," *Physics Letters A*, vol. 49, no. 3, pp. 197–198, 1974.

[2] D. Reith, M. Pütz, and F. Müller-Plathe, "Deriving effective mesoscale potentials from atomistic simulations," *Journal of Computational Chemistry*, vol. 24, no. 13, pp. 1624–1636, 2003.

[3] T. C. Moore, C. R. Iacovella, and C. McCabe, "Derivation of coarse-grained potentials via multistate iterative boltzmann inversion," *The Journal of chemical physics*, vol. 140, no. 22, p. 224104, 2014.

[4] A. P. Lyubartsev, M. Karttunen, I. Vattulainen, and A. Laaksonen, "On coarse-graining by the inverse monte carlo method: Dissipative particle dynamics simulations made to a precise tool in soft matter modeling," *Soft Materials*, vol. 1, no. 1, pp. 121–137, 2002.

[5] W. G. Noid, J.-W. Chu, G. S. Ayton, V. Krishna, S. Izvekov, G. A. Voth, A. Das, and H. C. Andersen, "The multiscale coarse-graining method. i. a rigorous bridge between atomistic and coarse-grained models," *The Journal of Chemical Physics*, vol. 128, no. 24, p. 244114, 2008.

[6] W. Shinoda, R. Devane, and M. L. Klein, "Multi-property fitting and parameterization of a coarse grained model for aqueous surfactants," *Molecular Simulation*, vol. 33, no. 1-2, pp. 27–36, 2007.

[7] S. J. Marrink, A. H. de Vries, and A. E. Mark, "Coarse grained model for semi-quantitative lipid simulations," *The Journal of Physical Chemistry B*, vol. 108, no. 2, pp. 750–760, 2004.

[8] C. Avendaño, T. Lafitte, A. Galindo, C. S. Adjiman, G. Jackson, and E. A. Müller, "Saft-γ force field for the simulation of molecular fluids. 1. A single-site coarse grained model of carbon dioxide," *The Journal of Physical Chemistry B*, vol. 115, no. 38, pp. 11154–11169, 2011.

[9] T. Bereau and K. Kremer, "Automated parametrization of the coarse-grained martini force field for small organic molecules," *J Chem Theory Comput*, vol. 11, no. 6, pp. 2783–2791, 2015.

[10] Y. Qi, H. I. Ingólfsson, X. Cheng, J. Lee, S. J. Marrink, and W. Im, "CHARMM-GUI martini maker for coarse-grained simulations with the martini force field," *Journal of Chemical Theory and Computation*, vol. 11, no. 9, pp. 4486–4494, 2015.

[11] V. Rühle and C. Junghans, "Hybrid approaches to coarse-graining using the VOTCA package: Liquid hexane," *Macromolecular Theory and Simulations*, vol. 20, no. 7, pp. 472–477, 2011.

[12] T. C. Moore, C. R. Iacovella, and C. McCabe, "Derivation of coarse-grained potentials via multistate iterative boltzmann inversion," *The Journal of Chemical Physics*, vol. 140, no. 22, 2014.

[13] A. Mirzoev and A. P. Lyubartsev, "MagiC: Software package for multiscale modeling," *Journal of Chemical Theory and Computation*, vol. 9, no. 3, pp. 1512–1520, 2013.

[14] P. A. Golubkov, J. C. Wu, and P. Ren, "A transferable coarse-grained model for hydrogen-bonding liquids," *Physical Chemistry Chemical Physics*, vol. 10, pp. 2050–2057, 2008.

[15] T. Shen and S. Gnanakaran, "The stability of cellulose: a statistical perspective from a coarse-grained model of hydrogen-bond networks.," *Biophysical Journal*, vol. 96, no. 8, pp. 3032–3040, 2009.

[16] A. Milani, M. Casalegno, C. Castiglioni, and G. Raos, "Coarse-grained simulations of model polymer nanofibres," *Macromolecular Theory and Simulations*, vol. 20, no. 5, pp. 305–319, 2011.

[17] C. Peter and K. Kremer, "Multiscale simulation of soft matter systems - from the atomistic to the coarse-grained level and back," *Soft Matter*, vol. 5, pp. 4357–4366, 2009.

[18] M. Bulacu, N. Goga, W. Zhao, G. Rossi, L. Monticelli, X. Periole, D. P. Tieleman, and S. J. Marrink, "Improved angle potentials for coarse-grained molecular dynamics simulations," *Journal of Chemical Theory and Computation*, vol. 9, no. 8, pp. 3282–3292, 2013.

[19] V. A. Harmandaris, N. P. Adhikari, N. F. A. van der Vegt, and K. Kremer, "Hierarchical modeling of polystyrene: From atomistic to coarse-grained simulations," *Macromolecules*, vol. 39, no. 19, pp. 6708–6719, 2006.

[20] O. Bezkorovaynaya, A. Lukyanov, K. Kremer, and C. Peter, "Multiscale simulation of small peptides: Consistent conformational sampling in atomistic and coarse-grained models," *Journal of Computational Chemistry*, vol. 33, no. 9, pp. 937–949, 2012.

[21] A. J. Rzepiela, L. V. Schäfer, N. Goga, H. J. Risselada, A. H. De Vries, and S. J. Marrink, "Reconstruction of atomistic details from coarse-grained structures," 2010.

[22] X. Chen, P. Carbone, G. Santangelo, A. Di Matteo, G. Milano, and F. Müller-Plathe, "Backmapping coarse-grained polymer models under sheared nonequilibrium conditions," *Phys. Chem. Chem. Phys.*, vol. 11, pp. 1977–1988, 2009.

[23] M. Orsi, W. Ding, and M. Palaiokostas, "Direct mixing of atomistic solutes and coarse-grained water," *Journal of Chemical Theory and Computation*, vol. 10, no. 10, pp. 4684–4693, 2014.

[24] M. Orsi, "Comparative assessment of the elba coarse-grained model for water," *Molecular Physics*, vol. 112, no. 11, pp. 1566–1576, 2014.

[25] Q. Shi, S. Izvekov, and G. A. Voth, "Mixed atomistic and coarse-grained molecular dynamics: Simulation of a membrane-bound ion channel," *The Journal of Physical Chemistry B*, vol. 110, no. 31, pp. 15045–15048, 2006.

[26] A. J. Rzepiela, M. Louhivuori, C. Peter, and S. J. Marrink, "Hybrid simulations: combining atomistic and coarse-grained force fields using virtual sites," *Physical Chemistry Chemical Physics*, vol. 13, pp. 10437–10448, 2011.

[27] N. Di Pasquale, D. Marchisio, and P. Carbone, "Mixing atoms and coarse-grained beads in modelling polymer melts," *The Journal of Chemical Physics*, vol. 137, no. 16, p. 164111, 2012.

[28] T. A. Wassenaar, H. I. Ingólfsson, M. Prieß, S. J. Marrink, and L. V. Schäfer, "Mixing MARTINI: Electrostatic coupling in hybrid atomisticcoarse-grained biomolecular simulations," *The Journal of Physical Chemistry B*, vol. 117, no. 13, pp. 3516–3530, 2013.

[29] R. J. Gowers and P. Carbone, "A multiscale approach to model hydrogen bonding: The case of polyamide," *The Journal of Chemical Physics*, vol. 142, no. 22, p. 224907, 2015.

[30] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*. New York, NY, USA: Clarendon Press, 1989.

[31] J. P. Hansen and I. McDonald, *Theory of Simple Liquids*. London: Academic, 1990.

[32] K. B. Tarmyshov and F. Müller-Plathe, "Parallelizing a molecular dynamics algorithm on a multiprocessor workstation using openmp," *Journal of Chemical Information and Modeling*, vol. 45, no. 6, pp. 1943–1952, 2005.

[33] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995.

[34] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, "Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids," *Journal of the American Chemical Society*, vol. 118, no. 45, pp. 11225–11236, 1996.

[35] Z. Cao and G. A. Voth, "The multiscale coarse-graining method. XI. Accurate interactions based on the centers of charge of coarse-grained sites," *The Journal of Chemical Physics*, vol. 143, no. 24, p. 243116, 2015.

[36] A. J. Rzepiela, M. Louhivuori, C. Peter, and S. J. Marrink, "Hybrid simulations: combining atomistic and coarse-grained force fields using virtual sites," *Phys. Chem. Chem. Phys.*, vol. 13, pp. 10437–10448, 2011.

[37] S. Bevc, C. Junghans, and M. Praprotnik, "STOCK: Structure mapper and online coarse-graining kit for molecular simulations," *Journal of Computational Chemistry*, vol. 36, no. 7, pp. 467–477, 2015.

[38] P. Carbone, H. A. K. Varzaneh, X. Chen, and F. Müller-Plathe, "Transferability of coarse-grained force fields: The polymer case," *The Journal of Chemical Physics*, vol. 128, no. 6, p. 064904, 2008.

[39] H. Wang, C. Junghans, and K. Kremer, "Comparative atomistic and coarse-grained study of water: What do we lose by coarse-graining?," *The European Physical Journal E*, vol. 28, no. 2, pp. 221–229, 2009.

[40] S. Páll, M. Abraham, C. Kutzner, B. Hess, and E. Lindahl, "Tackling exascale software challenges in molecular dynamics simulations with gromacs," in *Solving Software Challenges for Exascale* (S. Markidis and E. Laure, eds.), vol. 8759 of *Lecture Notes in Computer Science*, pp. 3–27, Springer International Publishing, 2015.

[41] M. Siwko, *Disturb or stabilise? Effects of different molecules on biological membranes*. PhD thesis, University of Groningen, 2008.

[42] J. Domański, P. J. Stansfeld, M. S. P. Sansom, and O. Beckstein, "Lipidbook: A public repository for force-field parameters used in membrane simulations," *The Journal of Membrane Biology*, vol. 236, no. 3, pp. 255–258, 2010.

[43] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, "A smooth particle mesh ewald method," *The Journal of chemical physics*, vol. 103, no. 19, pp. 8577–8593, 1995.

[44] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," *The Journal of Chemical Physics*, vol. 81, no. 8, pp. 3684–3690, 1984.

[45] G. Bussi, D. Donadio, and M. Parrinello, "Canonical sampling through velocity rescaling," *The Journal of Chemical Physics*, vol. 126, no. 1, p. 014101, 2007.

[46] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, "MDAnalysis: A toolkit for the analysis of molecular dynamics simulations," *Journal of Computational Chemistry*, vol. 32, no. 10, pp. 2319–2327, 2011.

[47] D. Dotson, R. J. Gowers, and O. Beckstein, "MDSynthesis: Alpha release: 0.5.0," 2015.

[48] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[49] M. Waskom, O. Botvinnik, P. Hobson, J. Warmenhoven, J. B. Cole, Y. Halchenko, J. Vanderplas, S. Hoyer, S. Villalba, E. Quintero, A. Miles, T. Augspurger, T. Yarkoni, C. Evans, D. Wehner, L. Rocher, T. Megies, L. P. Coelho, E. Ziegler, T. Hoppe, S. Seabold, S. Pascual, P. Cloud, M. Koskinen, C. Hausler, kjemmett, D. Milajevs, A. Qalieh, D. Allan, and K. Meyer, "seaborn: v0.6.0 (June 2015)," 2015.

[50] T. Kulschewski and J. Pleiss, "A molecular dynamics study of liquid aliphatic alcohols: simulation of density and self-diffusion coefficient using a modified OPLS force field," *Molecular Simulation*, vol. 39, no. 9, pp. 754–767, 2013.

[51] D. C. Rapaport, "Hydrogen bonds in water," *Molecular Physics*, vol. 50, no. 5, pp. 1151–1162, 1983.

[52] D. D. Humphreys, R. A. Friesner, and B. J. Berne, "A multiple-time-step molecular dynamics algorithm for macromolecules," *The Journal of Physical Chemistry*, vol. 98, no. 27, pp. 6885–6892, 1994.

[53] H. Grubmüller and P. Tavan, "Multiple time step algorithms for molecular dynamics simulations of proteins: How good are they?," *Journal of computational chemistry*, vol. 19, no. 13, pp. 1534–1552, 1998.

[54] N. Di Pasquale, R. J. Gowers, and P. Carbone, "A multiple time step scheme for multiresolved models of macromolecules," *Journal of Computational Chemistry*, vol. 35, no. 16, pp. 1199–1207, 2014.

[55] S. Izvekov and G. A. Voth, "Modeling real dynamics in the coarse-grained representation of condensed phase systems," *The Journal of Chemical Physics*, vol. 125, no. 15, p. 151101, 2006.

Chapter 10

Conclusion and outlook

The research presented in this thesis aimed to explore the potential of dual scale models for molecular dynamics (MD). These models exist at two levels of resolution, atomistic and coarse-grained (CG), and aim to find a compromise between the accuracy of atomistic models and the low computational cost of CG models. Targeted sections of the molecule can be represented in full detail, while other sections use a simplified description, requiring two force fields to be blended together to work in harmony. As the scope of computer simulation is limited by the computational cost of running simulations, these models will allow a wider range of potential applications.

In this work, two generations of a methodology for creating dual scale models were presented. The first generation, in which atoms and beads were directly bonded, was applied to different polymer melt systems, a polyamide system in Chapter 5 and polystyrene and polyethene systems in Chapter 6. The hydrogen bonding in the polyamide was examined in detail and found to provide an accurate representation. A multiple time step (MTS) scheme was tested with the polyethene and polystyene systems, and found to have no effect on the predicted structures and little effect on the dynamics.

The second generation of this methodology where all bonds were atomistic and all CG particles virtual, was applied to an octanol system in Chapter 9. Two independent octanol models were created based on an all-atomistic (AA) and united atom (UA) reference atomistic model respectively. This iteration of the methodology showed an improvement in the ability to model intramolecular structure, which was previously a weakness. Overall the dual scale models and algorithms presented were successful in recreating the targeted properties.

Section 10.1 outlines the lessons learned in how to construct dual scale models, their strengths and weaknesses and future directions of research. Section 10.3 highlights potential applications of dual scale models.

10.1 Findings on dual scale models

10.1.1 Choice of CG force field

Iterative Boltzmann inversion (IBI) force fields were used exclusively for the CG parts of the models presented. These were able to fulfill their purpose of recreating the same pairwise distributions as the atomistic reference systems without requiring any extra modifications. In the case of the polymer systems preexisting force field parameteres were used without modification, while in the octanol system the parameters were derived in situ, showing that either approach is feasible. Ultimately however the choice of what type of CG force field to use in dual scale models is still an open choice, and while IBI solutions work well they are not the only option[1].

10.1.2 Electrostatics

Electrostatic forces between partial charges in molecular models play a large role in their resulting behaviour. In the dual scale models presented these were retained in the atomistic resolution sections and both the reaction-field and particle-mesh Ewald (PME) approaches were successfully used to good effect in Chapters 5 and 9 respectively. Hydrogen bonding was used as a key metric for evaluating any changes to the electrostatic behaviour and in both cases very little effect was seen.

In Chapter 9 the partial charges on the AA octanol model were entirely removed and replaced with a CG representation. In this case, although the structures, and by extension the potential of mean force, were correctly reproduced by the CG potential introduced, the resulting diffusive behaviour was much slower. As has also been identified by others[2], the role of electrostatic forces in dual scale models needs further investigation as their treatment has wide ranging implications on the resulting models behaviour.

10.1.3 Intramolecular structures

The computational gain from dual scale models comes from the reduced number of nonbonded interactions, while the bonded structure was recognised as being cheap to simulate. The first generation model for polyamide had atoms and beads directly bonded to each other along the polymer backbone, necessitating potentials for bonds and angles with a mixture of atoms and beads. With CG potentials generally being much softer than atomistic potentials, it proved difficult to find potentials which performed well. This lead to these structural elements being poorly modelled in the dual scale model, which had a knock on effect in the overall structure of the polymer chains.

This deficiency was one of the motivations for the new methodology presented in Chapter 9, where all bonded interactions were modelled at the atomistic level. Using the original atomistic parameters the bonded structure was accurately reproduced in both the produced models of octanol, with the added bonus that this approach required less parameterisation. The calculation of all bonds at an atomistic level comes at a negligible computational and so there is little reason not to use them. While it is true that these interactions often dictate the integration time step in simulation, it is possible to work around this through either making the bonds rigid or using a multiple time step scheme as presented in Chapter 6.

10.1.4 Dynamics

The IBI force fields used are optimised around reference structures, with the model's dynamical behaviour not included as part of the process. A common result of CG IBI models is that their diffusion is orders of magnitude faster because of the reduced fluctuations in force caused by removing the degrees of freedom, but with a mixture of atomistic and CG forcefields, the expected dynamics are unclear.

For the dual scale polyamide system in Chapter 5, the measured self diffusion of the polymer chains was thirty times faster than the atomistic system, compared to 150× for the pure CG system, suggesting that the dynamics are somewhere between those of an atomistic and CG system. However at the smaller length scale of hydrogen bonds, it was observed that these were forming and breaking at around twice the rate of the atomistic system. This mixture of speed ups relative to the atomistic system indicate that the dynamics of the system had been unevenly affected. In the UA octanol model in Chapter 9 the dynamics of the atomistic and dual scale models were nearly matched, while the AA version showed the previously mentioned strange discrepency, where the bulk diffusion was slower in the dual scale model. The reasons for this behaviour are currently unclear, although it was theorised that the removal of charges created a sharper potential energy landscape causing molecules to become trapped in their surroundings more often.

For sampling static properties of the system, this uneven affect on dynamics is not a problem as each individual still represents a point in phase space that can be averaged over. However one of the key advantages of MD over other computational methods is its ability to explicitly describe the time evolution of events. With the rates of individual processes affected in a non uniform way the value of this time description is less clear.

10.1.5 Algorithm development and computational performance

With fewer degrees of freedom which must be calculated, dual scale models are intended to be computationally cheaper to simulate than atomistic models. When the models were ran in the well established Gromacs MD program[3] in Chapter 9, the performance was much slower than expected and worse than even the reference atomistic system. This is a disappointing outcome, but is not entirely surprising as the program was not designed with dual scale simulations in mind.

The earlier work presented in Chapters 5 and 6 used an MD program specially developed for dual scale MD and the scheme for implementing dual scale MD was set out in Chapter 7. The implementation requires significant changes to the internal structure of a program, but the individual algorithms used are relatively simple. Extending this implementation to include optimisations such as a multiple time step scheme was straightforward and the algorithms used are amenable to parallelisation and are therefore suitable for high performance computation. This potential performance was realised using the implementation detailed in Chapter 7, with simulation speed scaling correctly with the number of particles. The cost of using VS was shown to be minimal compared to other parts of the progam. This performance was pushed even further through the application of a multiple timestep (MTS) algorithm to take advantage of the lower characteristic frequency of the CG force field.

10.2 Recommendations for future work

10.2.1 Dynamics improvements

The typical approach to recover dynamics from a CG simulation is to rescale the simulated time to match a known scaling law or quantity[4]. In this approach, after simulating 10 ns of time, the trajectory can be stretched to represent 100 ns instead. However this is not possible when the dynamics for different length scales need rescaling by different amounts.

Instead a much better approach would be to apply friction to the CG particles during the simulation to bring them in line with the atomistic dynamics. This friction adds random noise to the forces between CG particles in order to replicate the lost degrees of freedom. This method has already been used in CG systems where a random force based on the Langevin equation was added to slow down the dynamics[5, 6]. Applying this to only the CG parts of a dual scale system should allow the dynamics to be matched to the atomistic system. Unfortunately this approach loses one of the benefits of CG simulation where phase space is traversed

faster per simulated time, however the alternative of speeding up the atoms to match the CG dynamics is not possible.

10.2.2 Software

Now that the potential of dual scale models has been proven both in this work and by other researchers, the next step in the adoption of these models is to try and make them much more accessible for a wider audience. Steps towards this were started in Chapter 9 with the introduction of a framework for defining dual scale simulations and automating the creation of the input files for them.

However it is clear that a dedicated high performance platform for performing these simulations is required. Whilst programs like those presented in Chapter 7 show the correct scaling, a great deal more work is required for them to be as fast as more established programs. The limitations in Gromacs could be resolved, however it is more likely that using a more flexible program could provide a better platform for running these simulations. Examples of such modern programs include Espresso++[7] which already has support for adaptive resolution simulations and OpenMM[8] which has been designed from the outset to be used as a library.

10.3 Possible applications of dual scale models

Based on the findings presented in this work, a wide variety of possibilities for using dual scale models exist of which two will be listed here.

10.3.1 Biomolecular simulations

A recurring motif in the systems studied so far is that they all feature molecules where a high level of detail is required for only specific groups, eg hydrogen donor and acceptor groups. Biomolecular systems features many molcules of this nature, for example lipid bilayer systems, which model the walls of a cell. The generic structure of a lipid consists of a hydrophilic head group with many partial charges and a hydrophobic tail which is chemically simpler.

The number of particles that can be simulated limits the study of these structures to simplified geometries that neglect their three dimensional shape. Coarse-grained representations have previously been used[9, 10], however these models rely on averaging out the behaviour of hydrogen bonding. In some cases, such as when assessing the permeation through the skin barrier[11], the role of hydrogen bonding has been shown to contribute greatly to the structural integrity of the bilayer. Therefore retaining the hydrogen bonding detail may be fundamental to reproducing realistic behaviour in this system.

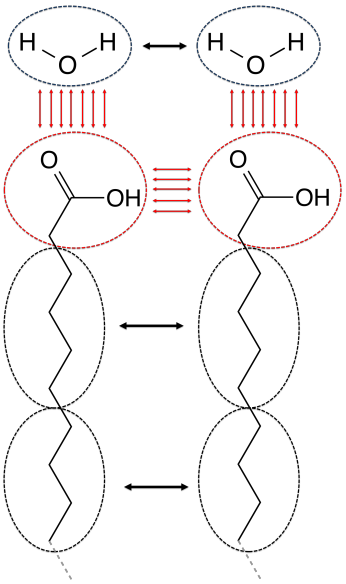


Figure 10.1: A potential dual scale model for lipids solvated in water. Proposed atomistic level interactions are shown as many red arrows, while single black arrows show CG interactions.

A possible dual scale model for a lipid molecule is shown in Figure 10.1, with only the detail on the headgroup retained. These systems are solvated with water, and so another challenge in designing a dual scale model for these systems would be to determine what level of detail is necessary to retain between the lipid headgroups and the water molecules. With a more computationally efficient model such as a dual scale model, larger systems could be simulated which would provide a more accurate representation of the underlying physics of the real system.

10.3.2 Conjugated polymers

Whilst the dual scale models examined in this work have all featured a mixture of atomistic and CG interactions in their nonbonded force field, this does not have to be the case. An interesting possibility is to construct a dual scale model which solves all nonbonded interactions in CG space and all bonded interactions in atomistic space.

One application for this methodology would be in highly conjugated polymers which feature many double bonds and aromatic rings along their length. These features highly restrict the conformational space that a polymer can inhabit, and modelling this has been shown to be difficult using a CG representation[12]. These polymers are used in photovoltaics, and their properties have been shown to be sensitive to the configuration of atomistic bonds[13]. Therefore a dual scale model employed here would allow the slow polymer dynamics to be sampled in a more timely manner, while still respecting the restrictions imposed by the double bonds using the atomistic level bonding. This formulation would be complimented well with a MTS scheme to allow the bonded interactions to be calculated more frequently

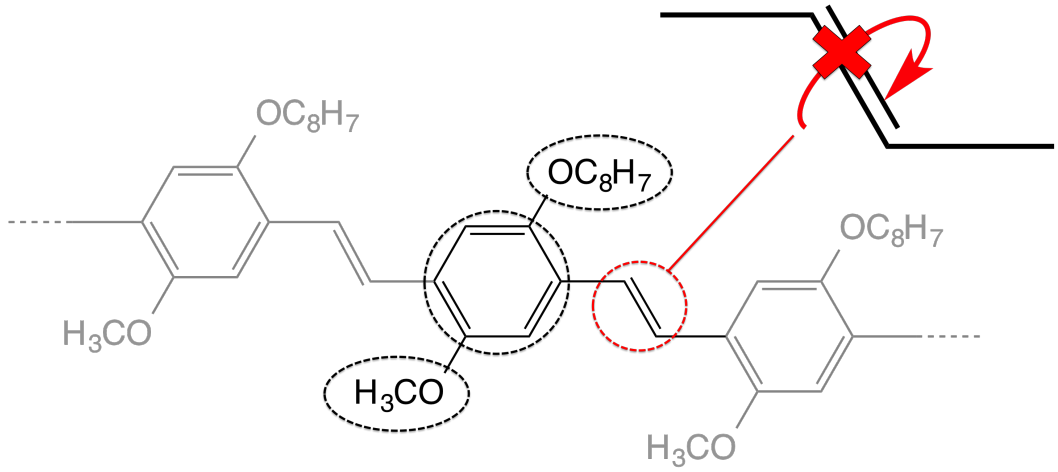


Figure 10.2: A hypothetical dual scale model for PPV. The restricted rotation around the double bonds is shown in inset.

than the nonbonded interactions.

10.3.3 Monte Carlo simulations

All of the molecular simulations presented until this point have been conducted using MD, however using dual scale models for Monte Carlo (MC) simulations presents an interesting alternative. MC simulations are similar to MD in that they feature a simulation volume filled with molecular models designed to sample a particular thermodynamic ensemble. In MC however, the system evolves through making random moves, such as displacing or rotating a molecule. These moves are accepted or rejected based upon the difference in energy that the move causes, with the probability, P, a move is accepted given by Equation 10.1[14].

$$P\left(o \to n\right) = \begin{cases} \exp\left[-\frac{1}{k_B T}\left(V\left(n\right) - V\left(o\right)\right)\right] & V\left(n\right) \geq V\left(o\right) \\ 1 & V\left(n\right) < V\left(o\right) \end{cases} \tag{10.1}$$

Where o is the state of the system before the move and n the state of the system after. If the move is accepted the system continues from that state n while if the move is rejected the system reverts to state o and a different move is attempted instead.

As the energy of the system, V, is calculated in the same way as in MD (Equation 2.2), it would be possible to again make this process computationally cheaper through using a dual scale model. Unlike MD, the results of a MC simulation are not a description of the system over time, but are instead just a series of points in phase space. This means that the previously mentioned issues with the dynamics of the system are no longer an issue.

Bibliography

[1] N. Goga, M. N. Melo, A. J. Rzepiela, A. H. de Vries, A. Hadar, S. J. Marrink, and H. J. C. Berendsen, "Benchmark of schemes for multiscale molecular dynamics simulations," *Journal of Chemical Theory and Computation*, vol. 11, no. 4, pp. 1389–1398, 2015.

[2] T. A. Wassenaar, H. I. Ingólfsson, M. Prieß, S. J. Marrink, and L. V. Schäfer, "Mixing MARTINI: Electrostatic coupling in hybrid atomisticcoarse-grained biomolecular simulations," *The Journal of Physical Chemistry B*, vol. 117, no. 13, pp. 3516–3530, 2013.

[3] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1–2, pp. 19–25, 2015.

[4] C. Peter and K. Kremer, "Multiscale simulation of soft matter systems," *Faraday Discuss.*, vol. 144, pp. 9–24, 2010.

[5] S. Izvekov and G. A. Voth, "Modeling real dynamics in the coarse-grained representation of condensed phase systems," *The Journal of Chemical Physics*, vol. 125, no. 15, p. 151101, 2006.

[6] C. Junghans, M. Praprotnik, and K. Kremer, "Transport properties controlled by a thermostat: An extended dissipative particle dynamics thermostat," *Soft Matter*, vol. 4, pp. 156–161, 2008.

[7] J. D. Halverson, T. Brandes, O. Lenz, A. Arnold, S. Bevc, V. Starchenko, K. Kremer, T. Stuehn, and D. Reith, "ESPResSo++: A modern multiscale simulation package for soft matter systems," *Computer Physics Communications*, vol. 184, no. 4, pp. 1129–1149, 2013.

[8] P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J. P. Ku, K. A. Beauchamp, T. J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston,

T. Stich, C. Klein, M. R. Shirts, and V. S. Pande, "OpenMM 4: A reusable, extensible, hardware independent library for high performance molecular simulation," *Journal of Chemical Theory and Computation*, vol. 9, no. 1, pp. 461–469, 2013.

[9] K. R. Hadley and C. McCabe, "A simulation study of the self-assembly of coarse-grained skin lipids," *Soft Matter*, vol. 8, pp. 4802–4814, 2012.

[10] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, "The MARTINI force field: Coarse grained model for biomolecular simulations," *The Journal of Physical Chemistry B*, vol. 111, no. 27, pp. 7812–7824, 2007.

[11] R. Notman and J. Anwar, "Breaching the skin barrier – insights from molecular simulation of model membranes," *Advanced Drug Delivery Reviews*, vol. 65, no. 2, pp. 237–250, 2013.

[12] V. A. Harmandaris, N. P. Adhikari, N. F. A. van der Vegt, and K. Kremer, "Hierarchical modeling of polystyrene: From atomistic to coarse-grained simulations," *Macromolecules*, vol. 39, no. 19, pp. 6708–6719, 2006.

[13] T. Qin and A. Troisi, "Relation between structure and electronic properties of amorphous MEH-PPV polymers," *Journal of the American Chemical Society*, vol. 135, no. 30, pp. 11247–11256, 2013.

[14] D. Frenkel and B. Smit, *Understanding Molecular Simulation*. Orlando, FL, USA: Academic Press, Inc., 2nd ed., 2001.

Part III

Appendices

Appendix A

Cover reprints

A.1 CG review

Encyclopedia of Nanotechnology
DOI 10.1007/978-94-007-6178-0_100940-1

Coarse-Grained and Hybrid Simulations of Nanostructures

Richard Gowers* and Paola Carbone
School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, UK

Synonyms

Mesoscopic simulations; Multiscale simulations

Definition

In computational chemistry coarse-grained (CG) models are defined as molecular models where some details (i.e., degrees of freedom) of the original chemical structure have been removed. The resulting models are a coarser description of the chemical systems compared with the original ones and can then be used to perform either molecular dynamics or Monte Carlo simulations [1]. The reduction of the models' degrees of freedom enables the simulation of systems whose size is comparable with that of the experimental ones and the timescale spanned by these simulations can reach microseconds.

Overview

Computer modeling is a powerful technique to gain molecular level details of chemical systems under different physical conditions and enables to relate macroscopic observations with changes in the chemical and physical state of the system. However, all modeling techniques rely on computer hardware, and therefore their use is limited by the available computer power. State-of-art simulations can nowadays reach the size of few millions of atoms, but if standard high-performance computers are used, the system size usually does not exceed few hundred thousand particles. Indeed, during a molecular simulation, the number of interatomic interactions that must be computed every iteration is proportional to N^2, where N is the total number of system particles [1]. This heavy use of the CPUs limits not only the size of molecular models but also the timescale the system can be simulated for.

One way to circumvent this problem is to reduce the number of interacting particles (N) in the systems, simplifying the models and modifying the original interacting parameters to include in an implicit way the neglecting details. The simplest example of such coarse graining is the development of united-atom (UA) force fields [2] where the hydrogen atoms and the aliphatic carbon to which they are covalently bonded are modeled as a single entity. The assumption underlying the development of such force fields is that the physics of the model is not affected by neglecting the explicit interactions involving the aliphatic hydrogen atoms. A similar decision on how many and which atomistic details can be neglected in a molecular model (procedure known as mapping scheme) is the key decision that must be carefully made every time a new coarse-grained model is developed if some of the system chemical and physical features have to be maintained. It has been indeed shown that mapping schemes which retain different features of the original molecule perform differently depending on the property analyzed [3].

*Email: richard.gowers@postgrad.manchester.ac.uk

A.2 PA Paper

THE JOURNAL OF CHEMICAL PHYSICS **142**, 224907 (2015)

A multiscale approach to model hydrogen bonding: The case of polyamide

Richard J. Gowers[a] and Paola Carbone[b]
School of Chemical Engineering and Analytical Science, University of Manchester, Manchester M13 9PL, United Kingdom

(Received 5 February 2015; accepted 2 June 2015; published online 12 June 2015)

We present a simple multiscale model for polymer chains in which it is possible to selectively remove degrees of freedom. The model integrates all-atom and coarse-grained potentials in a simple and systematic way and allows a fast sampling of the complex conformational energy surface typical of polymers whilst maintaining a realistic description of selected atomistic interactions. In particular, we show that it is possible to simultaneously reproduce the structure of highly directional non-bonded interactions such as hydrogen bonds and efficiently explore the large number of conformations accessible to the polymer chain. We apply the method to a melt of polyamide removing from the model only the degrees of freedom associated to the aliphatic segments and keeping at atomistic resolution the amide groups involved in the formation of the hydrogen bonds. The results show that the multiscale model produces structural properties that are comparable with the fully atomistic model despite being five times faster to simulate. © *2015 AIP Publishing LLC.* [http://dx.doi.org/10.1063/1.4922445]

I. INTRODUCTION

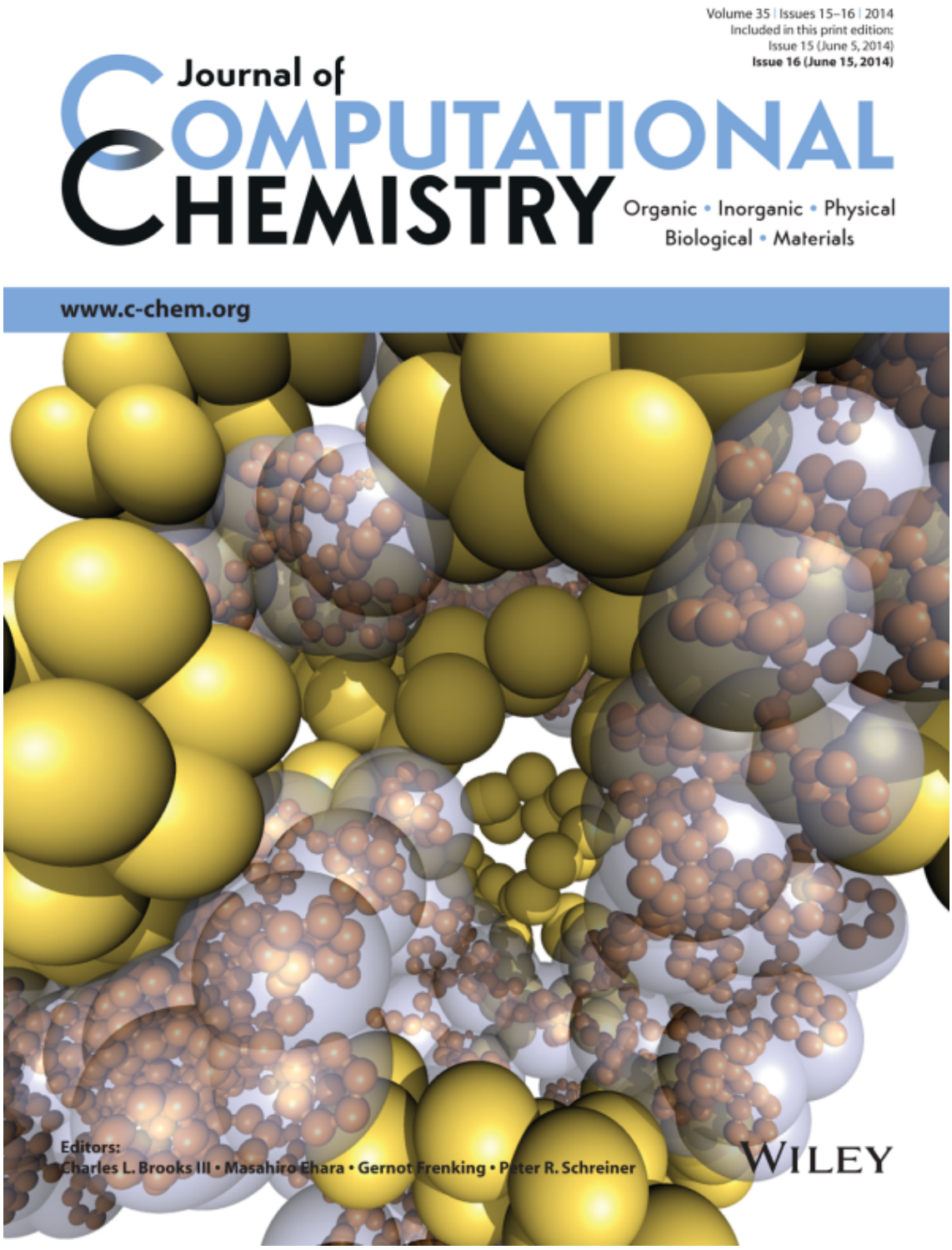
The use of chemical-specific coarse-grained (CG) models to simulate soft materials such as polymers, surfactants, or high viscous liquids has become very popular since the early pioneering attempts.[1–3] This is because despite the computational power available, the modeling of soft, slow relaxing materials is still a challenge when using all-atom (AA) force fields. Specific coarse-grained models can overcome the problem of the slow sampling of complex energy surfaces, reducing the number of degrees of freedom of the molecular system and in some cases, flattening out the energy landscape.[4,5] In the past years, several approaches to the coarse-graining of soft matter have been proposed. Most of them use a multiscale approach where data obtained from detailed atomistic simulations are targeted and the CG force field parameters are refined until they match the target properties. Over the years, different target properties for the CG models have been proposed including using structural data,[6] mechanical properties,[7] and thermodynamic data such as density values,[8] partition coefficients,[9] inter-particle forces,[10] configurational entropy,[11] or potential of mean force.[12] Using CG models, phenomena not accessible before via molecular dynamics can now be modelled; for example, it is now possible to gain a molecular understanding of complicated self-assembling processes involving polymers, surfactants, or ionic liquids.[13–16]

Despite the obvious advantages in using CG models, there still exist some drawbacks. For example, it is sometimes complicated to understand how to model charged systems such as polyelectrolytes or ionic liquids: the charge is indeed delocalised on several atoms and the use of a point charge located on a single bead might not be the appropriate way to model it.[17] Another limit of a CG approach is the lack of a proper description of the hydrogen bonds (HBs).[18–20] The atoms involved in these highly directional non-bonded interactions are grouped into single (or different) CG beads and the interaction is averaged away with other non-bonded interactions. In the case of biopolymers where the hydrogen bond network is responsible for their three dimensional structure, this problem has been solved developing specific CG force field parameters for each aminoacid or DNA base;[21–23] however, such highly targeted approaches cannot be used for other synthetic materials whose 3D structure is not known *a priori*. The formation and disruption of the hydrogen bonding network are also intrinsically related to the system dynamics which can be altered when the model lacks an explicit treatment of it.[24] We have shown that this is indeed the case for structural-based CG models which implicitly take into account the HBs interactions in their effective potentials obtained from radial distribution functions.[24,25] The dynamics of the CG model in this case correlates with the dynamics of the HBs network. At high temperature, the latter has a negligible effect on the former as the HBs network is very weak, but lowering the temperature, the importance of the presence of HBs interactions between the amide groups becomes dominant.

One possible solution to this problem is to develop a multiresolved model, where the molecular system is modelled at two different levels of resolution. These two differently resolved models can be linked by the use of replica exchange[26–28] or be used simultaneously in a single simulation. The latter approach uses atoms and CG beads simultaneously and has been employed to simulate simple fluids,[29] polymers,[30–32] and selected biological systems.[33–37] In particular, we have shown that it is possible to seamlessly integrate an AA force field with a structural based CG force field in modeling single polymer chains.[31] Recently, we have also developed a simple multiple time step scheme which can take advantage of the intrinsic division between different length and time scales in the dynamics of these kind of models.[38]

[a]Electronic mail: richard.gowers@manchester.ac.uk
[b]Electronic mail: paola.carbone@manchester.ac.uk

A.3 MTS Cover page



A.4 MTS Paper

Journal of COMPUTATIONAL CHEMISTRY WWW.C-CHEM.ORG FULL PAPER

A Multiple Time Step Scheme for Multiresolved Models of Macromolecules

Nicodemo Di Pasquale,[a] Richard J. Gowers,[b] and Paola Carbone*[b]

In hybrid particle models where coarse-grained beads and atoms are used simultaneously, two clearly separate time scales are mixed. If such models are used in molecular dynamics simulations, a multiple time step (MTS) scheme can therefore be used. In this manuscript, we propose a simple MTS algorithm which approximates for a specific number of integration steps the slow coarse-grained bead–bead interactions with a Taylor series approximation while the atom–atom ones are integrated every time step. The procedure is applied to a previously developed hybrid model of a melt of atactic polystyrene (di Pasquale, Marchisio, and Carbone, *J. Chem. Phys*. 2012, 137, 164111). The results show that structure, local dynamics, and free diffusion of the model are not altered by the application of the integration scheme which can confidently be used to simulate multiresolved models of polymer melts. © 2014 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.23594

Introduction

In recent years, the development of novel computational techniques which combine different levels of resolution of molecular models has become an exciting new field in computational chemistry and physics.[1] These multiresolved models could in fact be used to overcome one of the most challenging problems in molecular dynamics (MD) simulations namely the difficulty in sampling large conformational spaces with models which retain atomistic details. Commonly the problem of the reduced length scale of MD simulations is solved by resorting to coarse-grained (CG) models where the degrees of freedom of the molecular system are reduced and the energy surface to sample is consequently flattened out. Although these techniques[2] have proved to be extremely powerful in predicting structural,[3] thermodynamic,[4] and dynamical[5] properties of complex and slow relaxing systems, the simplification of the chemical structure of the molecular system prevents gaining a detailed picture of its atomistic interactions unless atoms are reinserted in the coarse model during a second stage of the simulation.[6] One of the solutions to this problem is the development of molecular models which combine two levels of resolution one of which is at the atomistic level. In literature, there are already a few examples of such models developed for simple fluids,[7] polymers,[8,9] and selected biological systems.[10] The development of such models poses of course several fundamental questions related to their thermodynamics.[1,11] In particular, in cases where atoms are combined with beads, two particle-based models characterized by different dynamics are mixed.[12] In most of the cases it has been shown that the dynamics of a CG model is the same as the corresponding atomistic one except that the former is some orders of magnitude faster than the latter.[5,13] Additionally, in a CG model, the collapse of several atoms into a single bead and consequently the removal of the fast oscillations of the atom–atom bond interactions allow the use of a time step (Δt) to integrate the equation of motion which can be considerably larger than that used for all-atoms (AA) models. In CG MD simulations, a Δt between 10 and 20 fs is considered appropriate, whereas in an AA simulation, a Δt of 1 or 2 fs (depending on whether bonds involving the light hydrogen atoms are allowed to oscillate) is normally used. Therefore, when two models whose dynamics can be sampled at two different time scales are mixed together, a multiple time step (MTS) scheme might be in principle applied. The use of a MTS algorithm would in fact fully exploit the advantage of using a dually resolved model and would speed up the simulation time even further by reducing considerably (as a function of the number of beads in the system) the computational time. In the past years, several MTS schemes have been proposed and validated for fully atomistic models. They include: generalized Verlet integration,[14] RESPA[15] and its derivation MOLLY,[16] and Langevin normal mode methods.[17] These methods are all based on the fact that already in an atomistic resolved model, it is not always necessary to sample all the atom pair interactions with the same frequency and, because of the existence of different time and spatial scales in the interactions, every component of the interaction could be considered with its own specific time. This approach in theory increases the efficiency of the simulation because of the computational time saved not computing interactions that can be sampled less frequently. For an AA model, however, the computational time saved is limited and the different time step lengths must be chosen with some criteria. In particular, due to the resonance instability,[18] the interval between the sampling of slow

[a] *N. Di Pasquale*
Dipartimento di Scienza dei Materiali e Ingegneria Chimica, Politecnico di Torino, C.so Duca degli Abruzzi 24, Torino 10129, Italy
[b] *R. J. Gowers, P. Carbone*
School of Chemical Engineering and Analytical Science, The University of Manchester, Oxford Road, Manchester M13 9PL, United Kingdom
E-mail: paola.carbone@manchester.ac.uk
Contract/grant sponsor: BBSRC; Contract/grant number: BB/J014478/1

Appendix B

Supporting information for: A multiscale approach to model hydrogen bonds: The case of polyamide

B.1 Forcefield details

B.1.1 Atom-Bead bonds

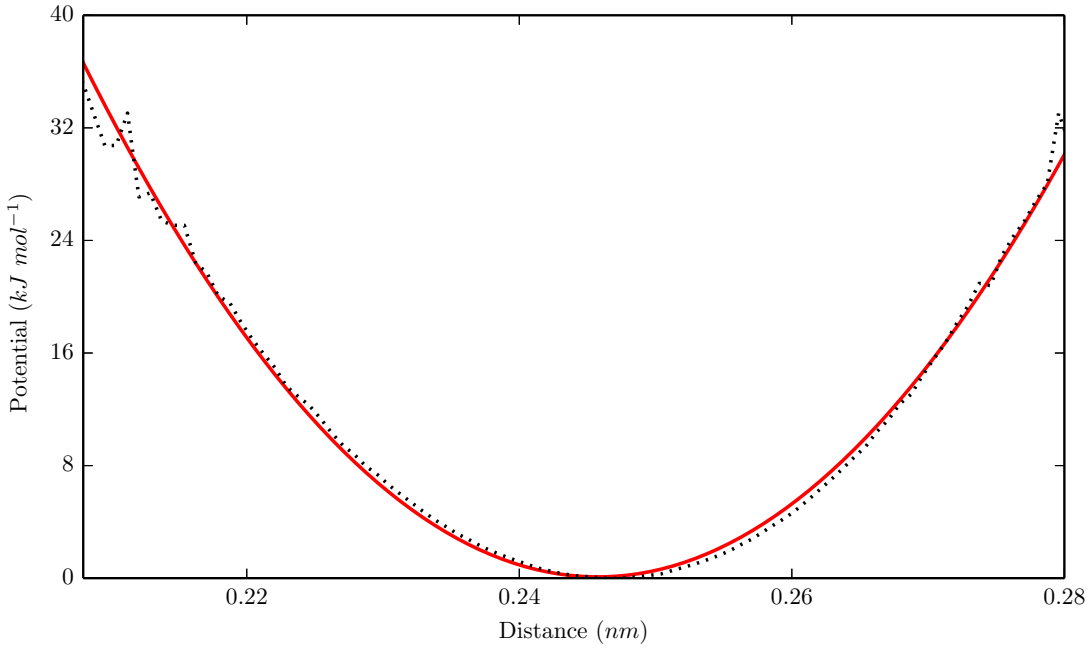
The use of a harmonic function to model the bond potential is justified in B.1, with a comparison of the potential described using a Boltzmann inversion and the analytical form, shown in equation B.1.

B.1 details the parameters used to model these bonds.

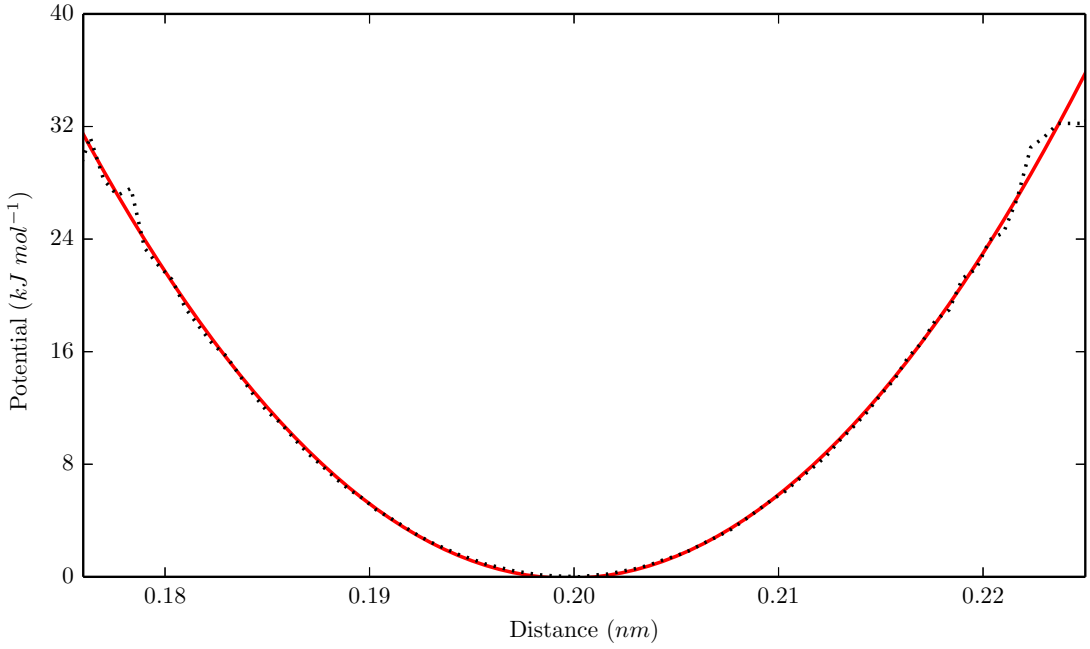
$$V_{bond} = k_{bond} \left(r_{ij} - r_{ij\ 0} \right)^2 \tag{B.1}$$

Bond	k_{bond} (kJ mol^{-1})	$r_{ij\ 0}$ (nm)
N – M3	25600	0.245
C – M2	56100	0.222

Table B.1: Derived parameters for atom – bead bonds used in the hybrid forcefield.



(a) N – M3



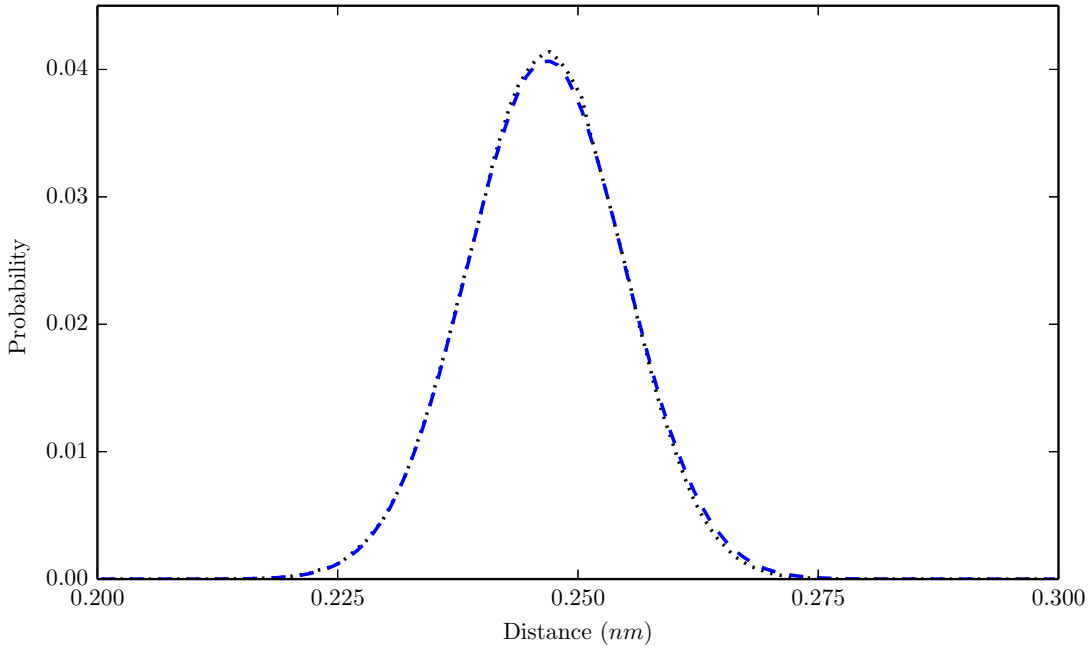
(b) C – M2

Figure B.1: Comparison of the Boltzmann inversion of atomistic probabilities, in black, and the harmonic function used to model it, in red.

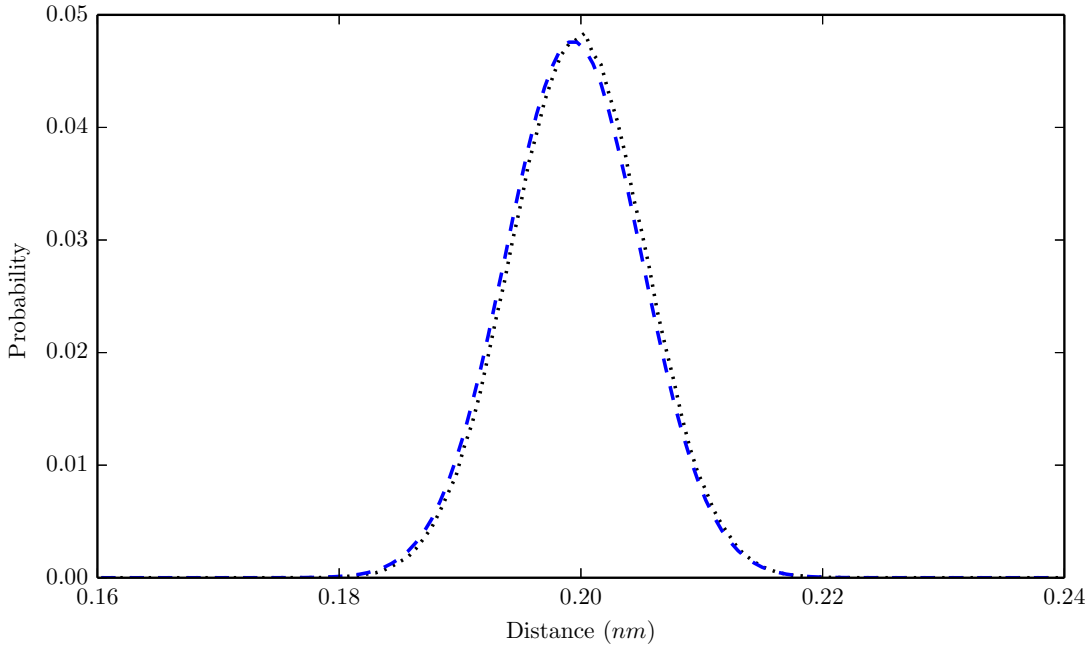
B.2 Structural results

B.2.1 Atom-Bead bonds

B.2 shows bond length distributions for bonds between atoms and beads in the atomistic and hybrid scale models. All distributions are based on 200 ns of NPT simulation at T=400 K and P=1 atm.



(a) N – M3



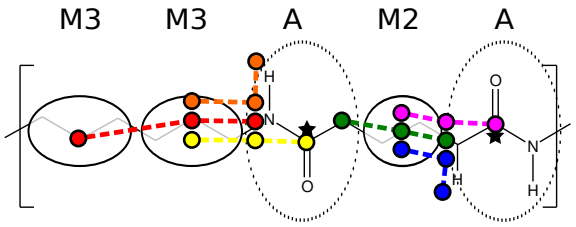
(b) C – M2

Figure B.2: Atom – Bead bond length distributions. Black-Atomistic, Blue-Hybrid.

B.2.2 Atom - Bead angles

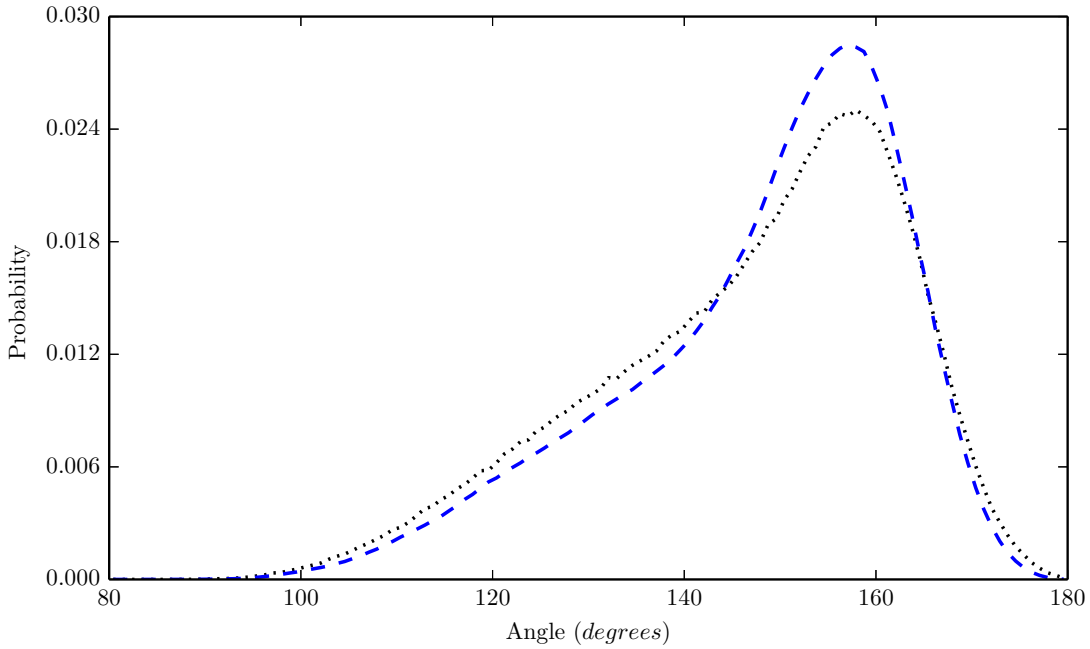
There are six different atom-bead angles which are required in the hybrid scale model, In B.4, angle probability distributions around all atom-bead angles in the forcefield. All distributions are based on 200 ns of NPT simulation at T=400 K and P=1 atm. In black, the reference atomistic distribution. To calculate these,

the atomistic trajectory was coarse-grained to the same level of detail as the hybrid model. The potentials for the hybrid model were constructed from this atomistic distributions using a single Boltzmann inversion. The resulting angle distribution attained in the hybrid scale model is then given in blue.

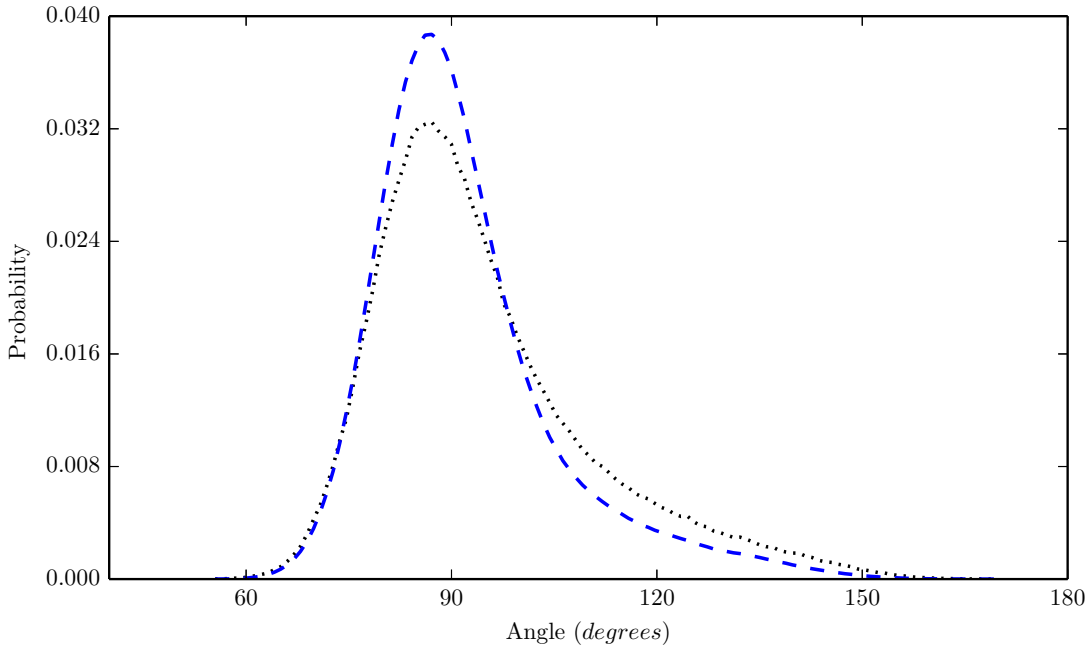


Colour	Angle
Red	M3 – M3 – N
Orange	M3 – N – H_N
Yellow	M3 – N – C_α
Green	C – M2 – C
Blue	M2 – C – H
Magenta	M2 – C – C_α

Figure B.3: Diagram of the different atom – bead angles used in the hybrid scale model

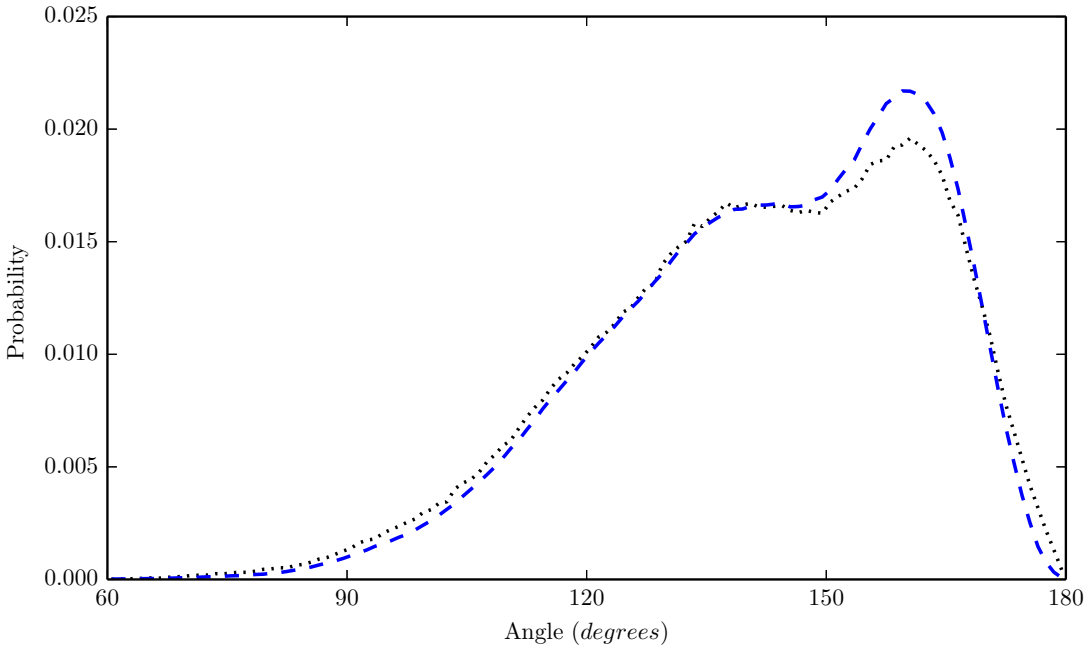


(a) C_α – N – M3

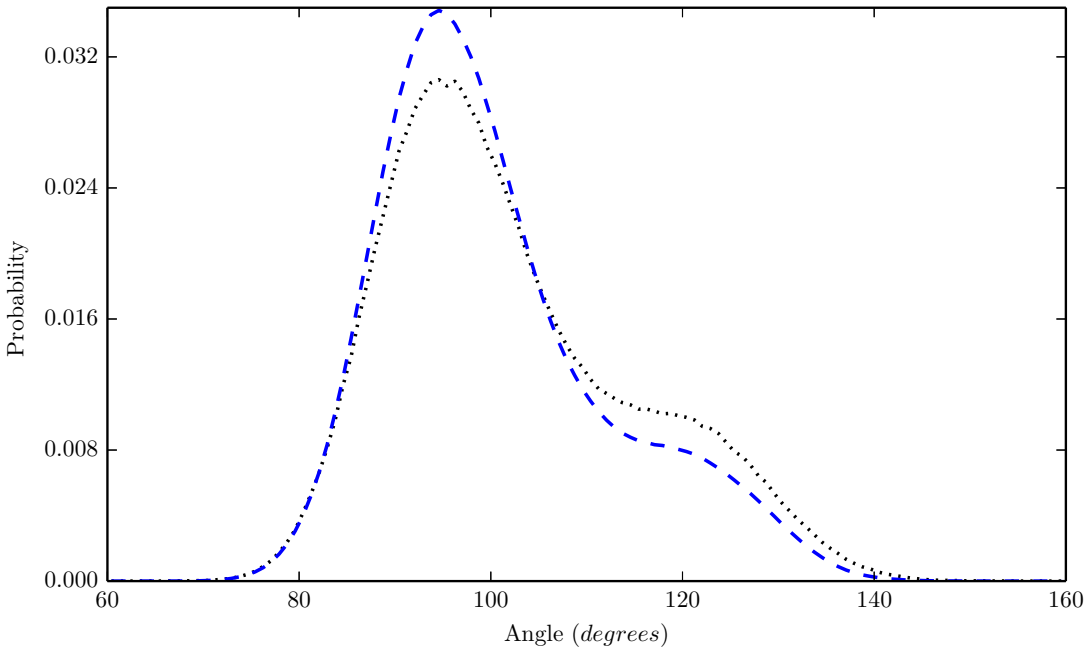


(b) H_N – N – M3

Figure B.4: Atom–Bead angle distributions. Black-Atomistic, Blue-Hybrid

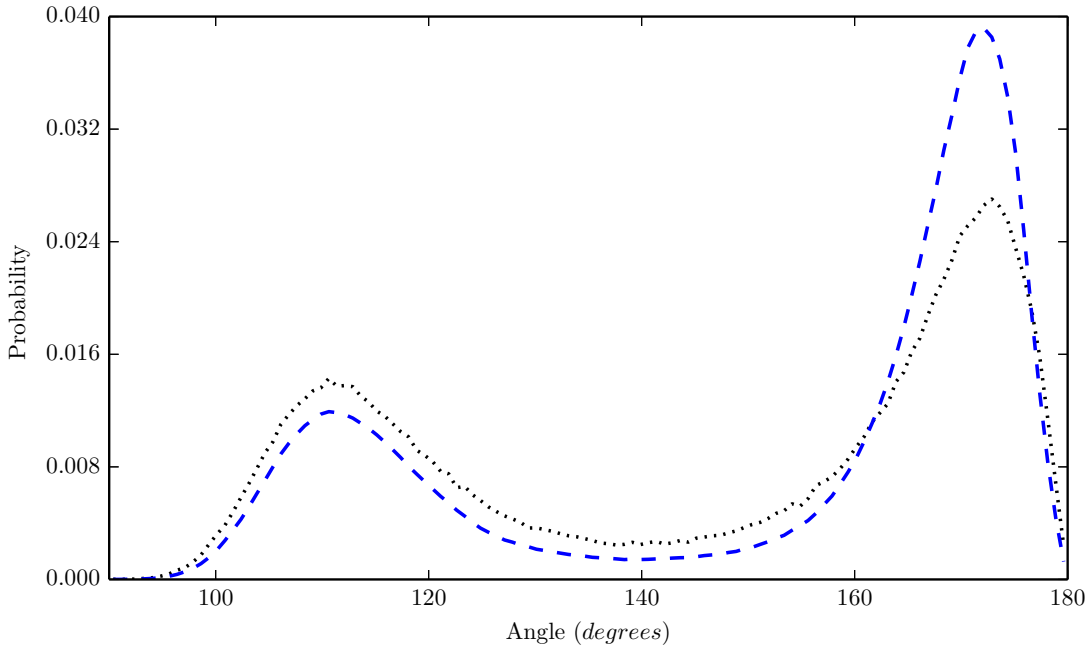


(c) N – M3 – M3

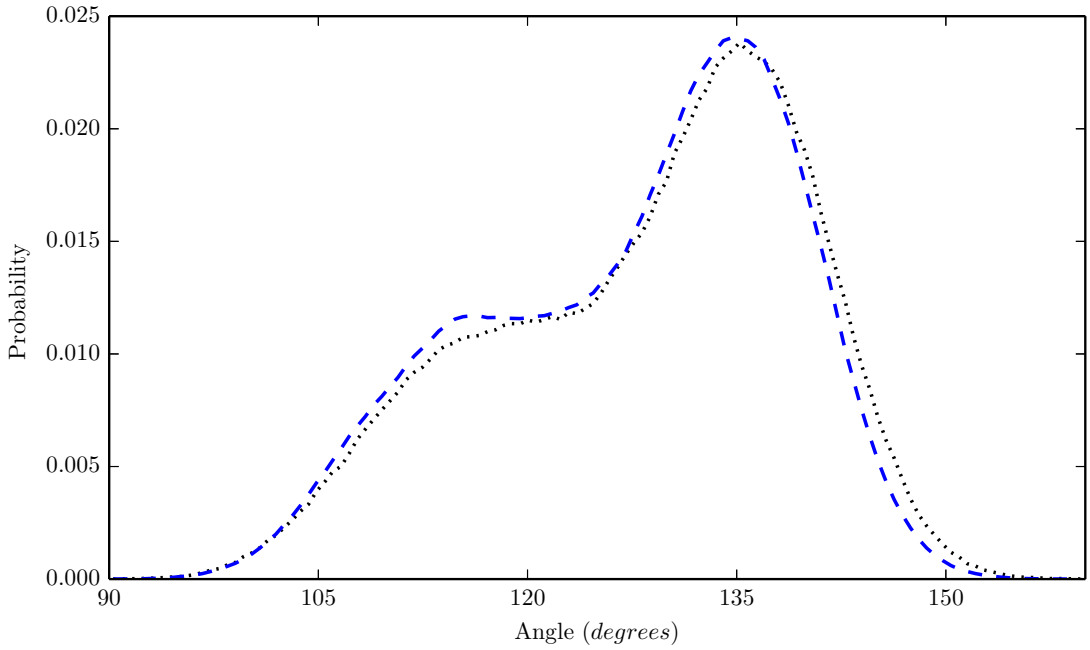


(d) H – C – M2

Figure B.4: Atom–Bead angle distributions. Black-Atomistic, Blue-Hybrid



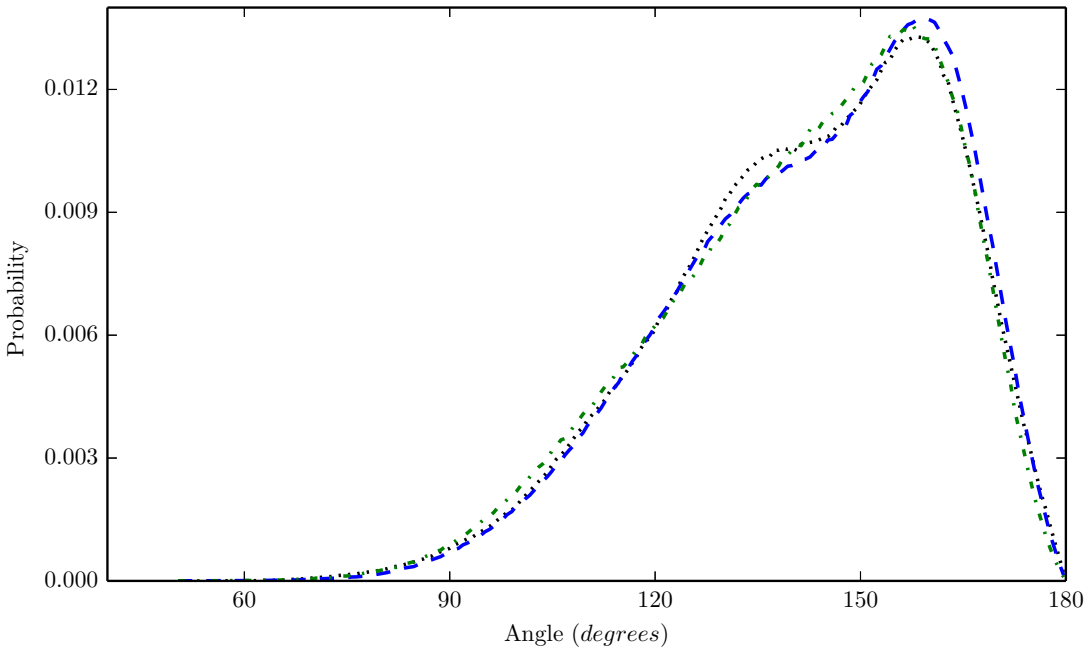
(e) C – M2 – C



(f) C_α – C – M2

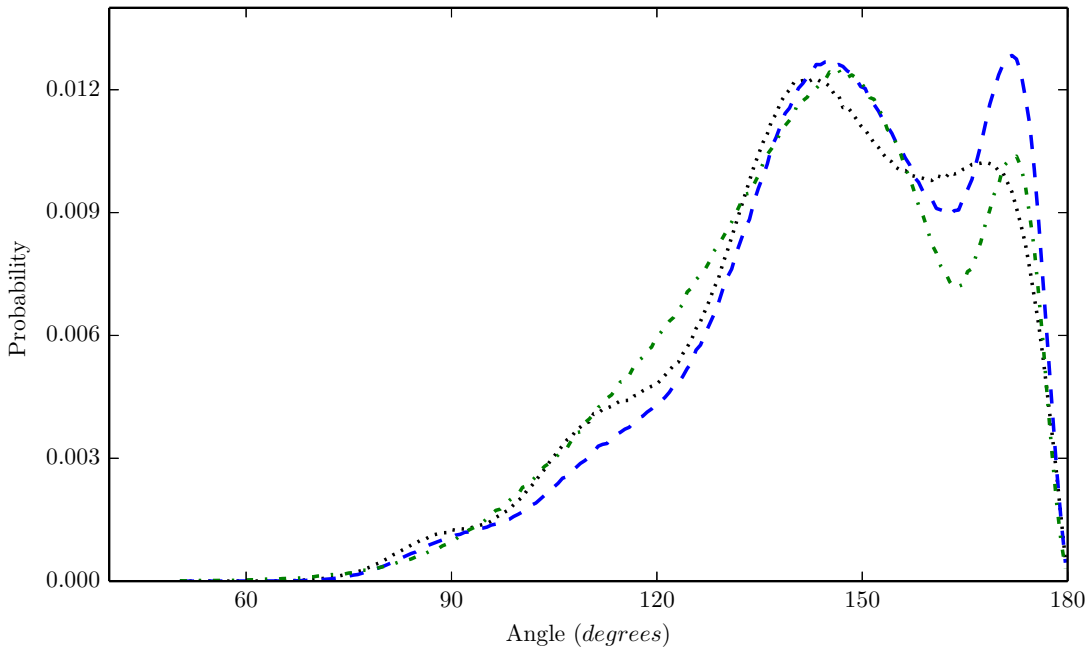
Figure B.4: Atom–Bead angle distributions. Black-Atomistic, Blue-Hybrid

B.2.3 Bead - Virtual Site angles

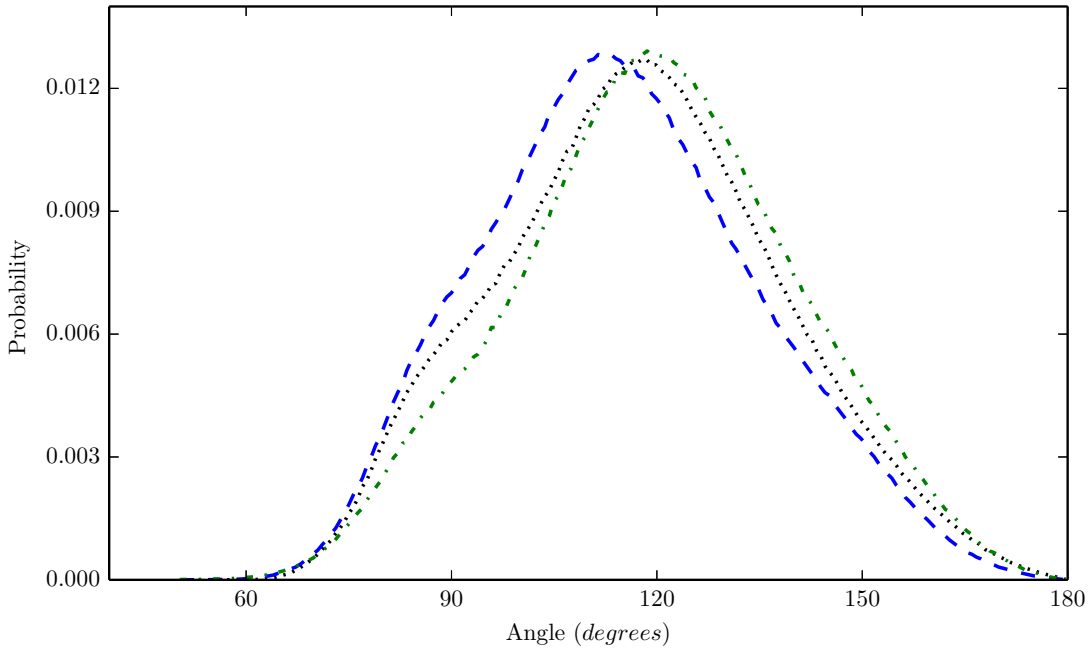


(a) A - M3 - M3

Figure B.5: Bead – Virtual site angle distributions. Black-Atomistic, Blue-Hybrid, Green-CG



(b) A - M2 - A



(c) M3 - A - M2

Figure B.5: Bead – Virtual site angle distributions. Black-Atomistic, Blue-Hybrid, Green-CG

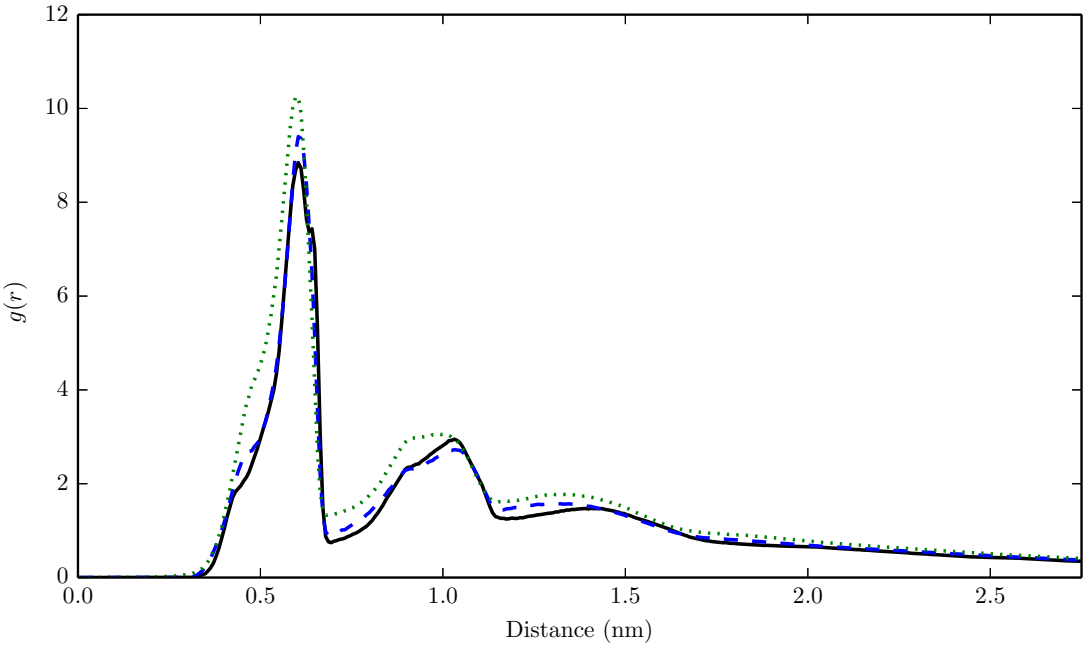
B.2.4 Intramolecular rdfs

Intramolecular RDFs are used to view the pair distribution of species strictly in the same polymer chain. When calculating these, the number density (ρ) of the particles is based on the number of the species found in a single chain (N).

$$g_{intra}(r) = \langle \frac{n(r)}{\rho V_{shell}} \rangle \tag{B.2a}$$

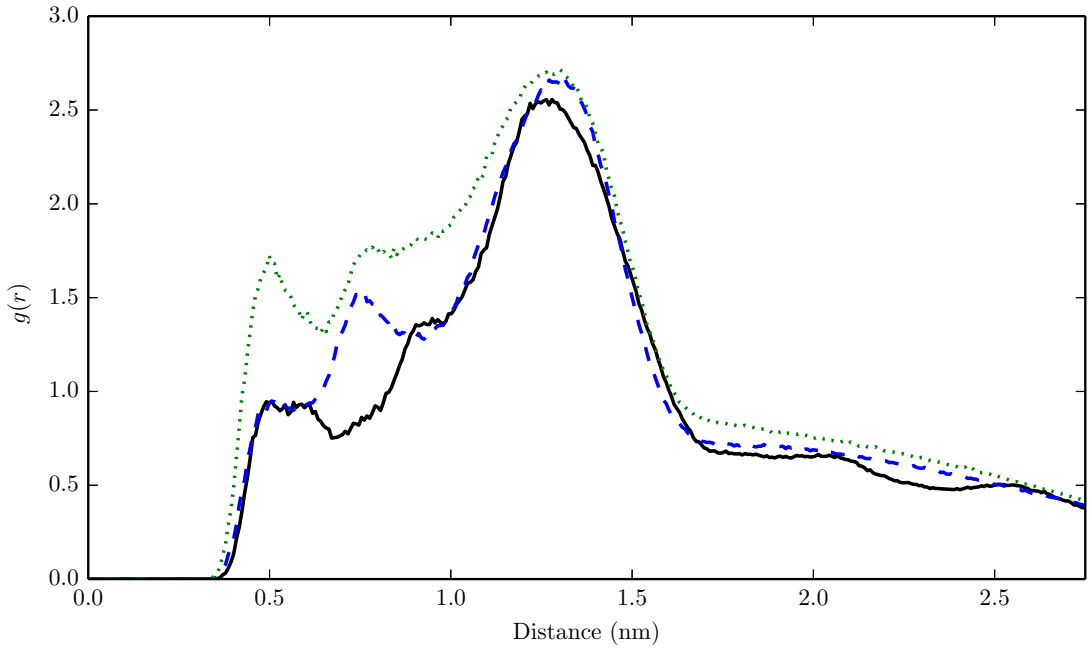
$$\rho = \frac{(N-1)}{V_{box}} \tag{B.2b}$$

$$V_{shell} = \frac{4}{3}\pi \left[\left(r + \frac{1}{2}\Delta r \right)^3 - \left(r - \frac{1}{2}\Delta r \right)^3 \right] \tag{B.2c}$$

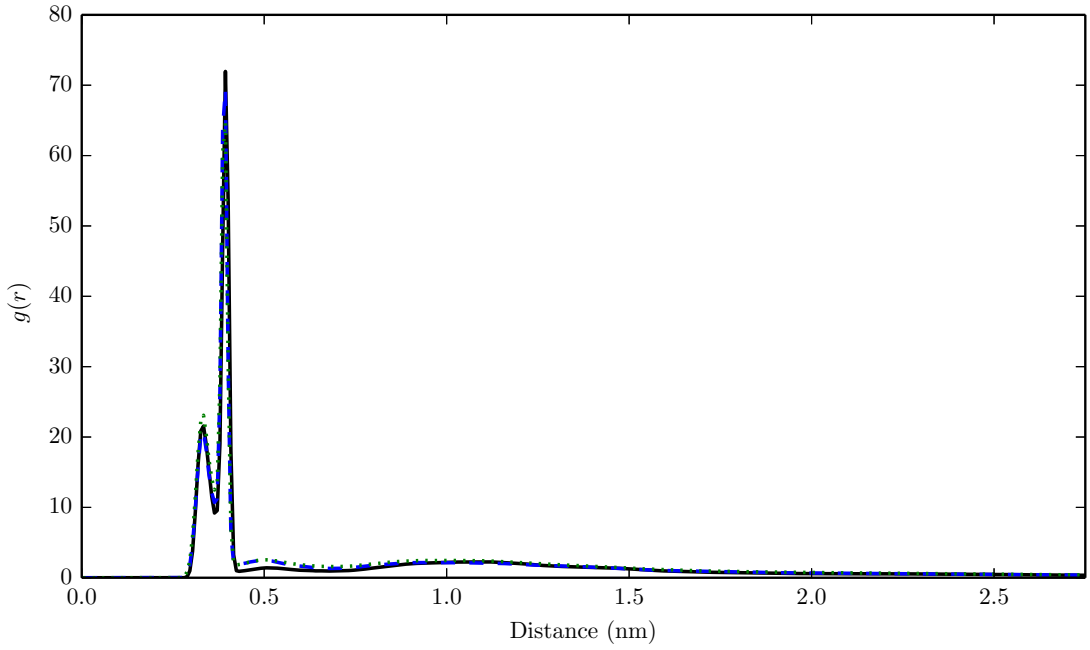


(a) A - A

Figure B.6: Intramolecular RDFs compared across different scale models. Colours: Black-Atomistic, Blue-Hybrid, Green-Coarse



(b) M2 - M2

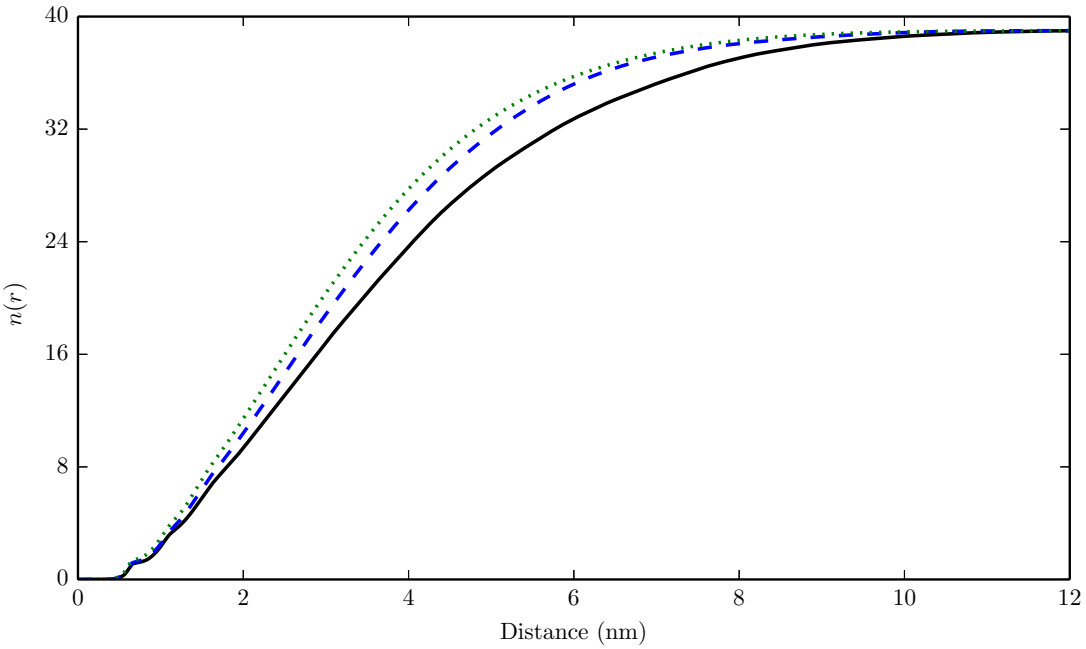


(c) M3 - M3

Figure B.6: Intramolecular RDFs compared across different scale models. Colours: Black-Atomistic, Blue-Hybrid, Green-Coarse

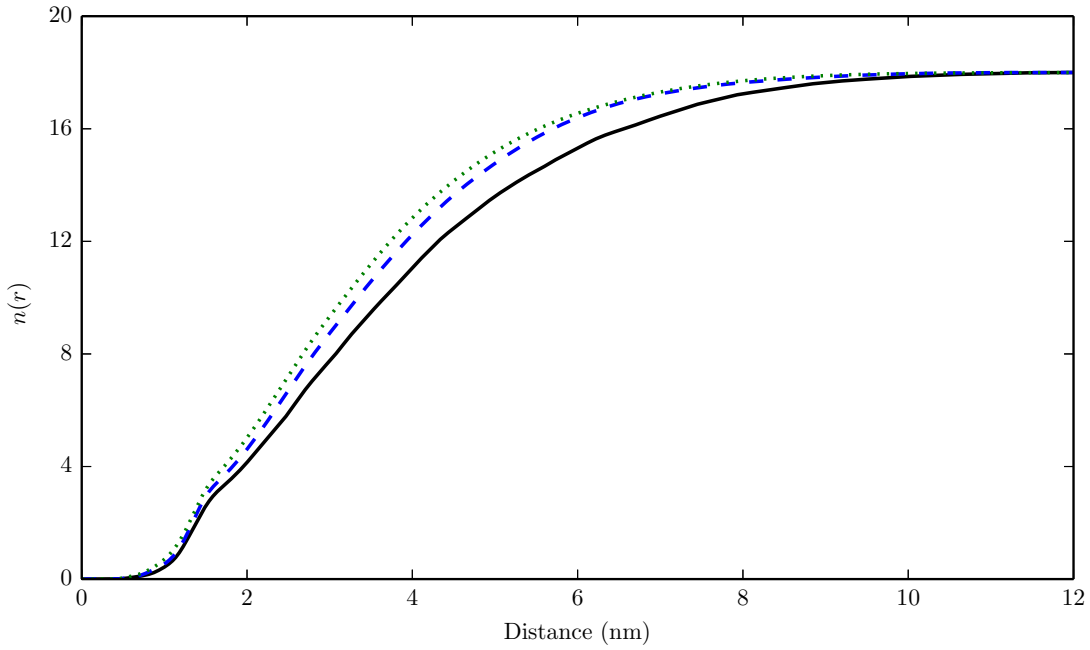
In addition, the cumulative sum of the number of particles as a function of radial distance can be plotted to better visualise the chain conformations. These converge to the total number of a given species in a chain, 39 for A and M3 and 18 for M2. The more tightly coiled conformations of the hybrid and coarse models are shown

by their quicker convergence to these values.

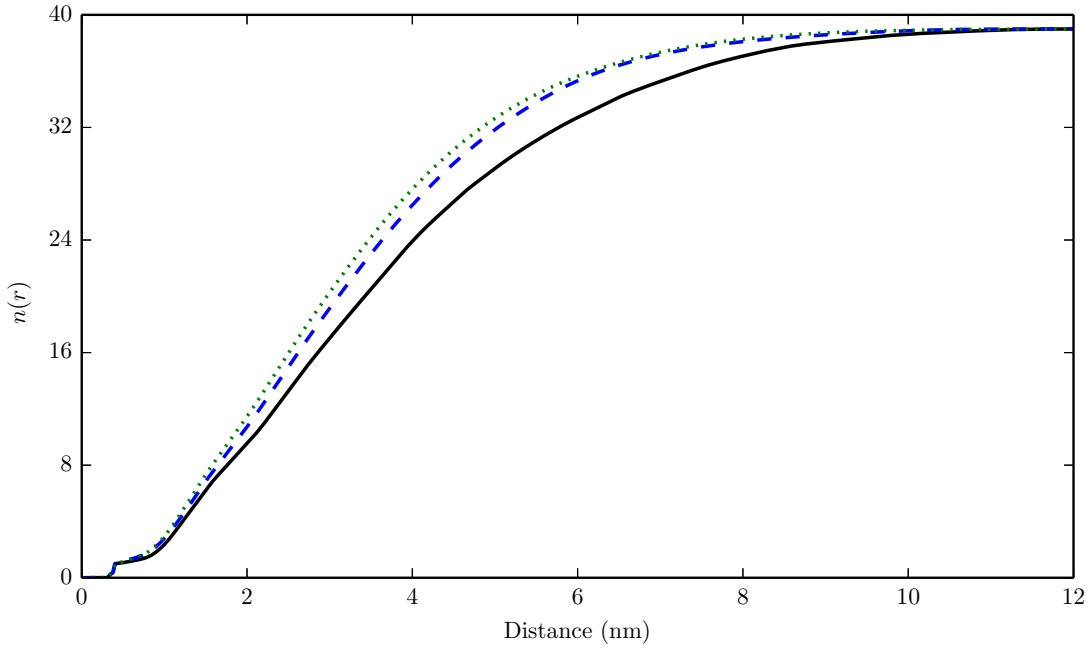


(a) A - A

Figure B.7: Cumulative sum of the number of particles. Colours as previously.



(b) M2 - M2



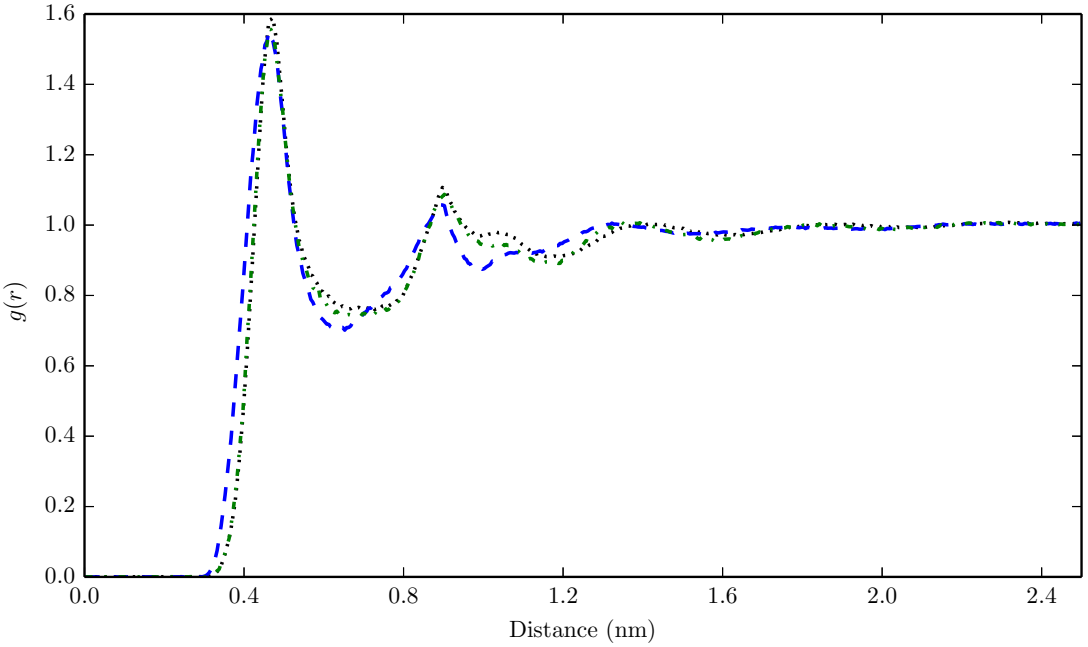
(c) M3 - M3

Figure B.7: Cumulative sum of the number of particles.
Colours as previously.

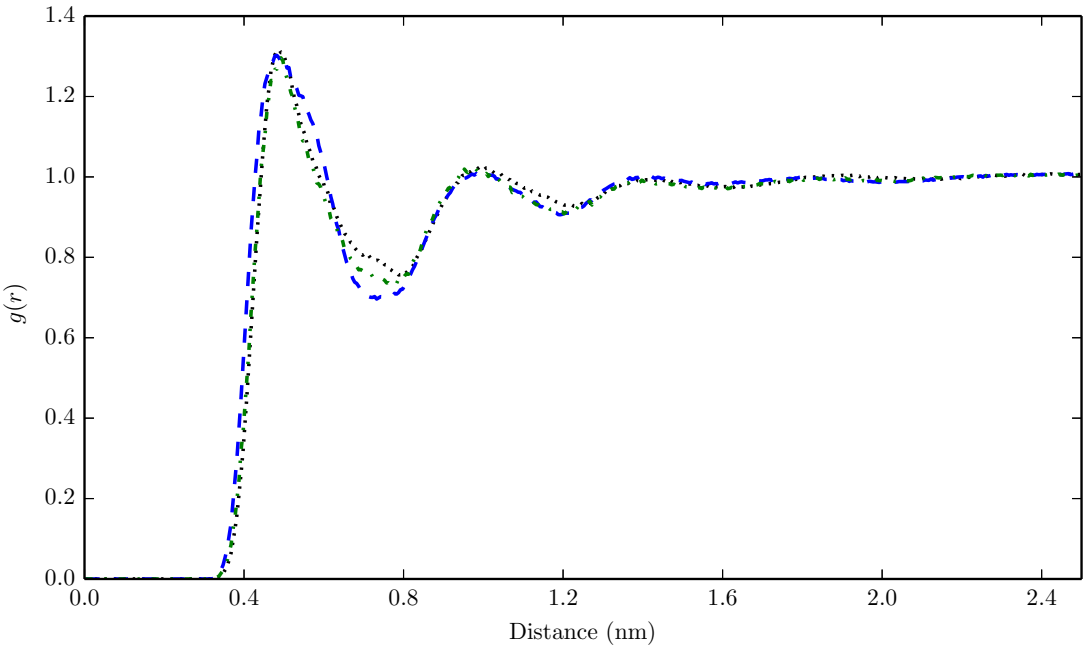
B.2.5 Intermolecular rdfs

Intermolecular radial distribution functions between all combinations beads, compared across the coarse-grained, hybrid and atomistic scale models. All distributions are based on 200 ns of NPT simulation at T=400 K and P=1 atm. In the case of the

atomistic and hybrid scale models, where some or all of these beads are not present, the trajectory is first coarse-grained to the same level as the coarse-grained before the calculations.

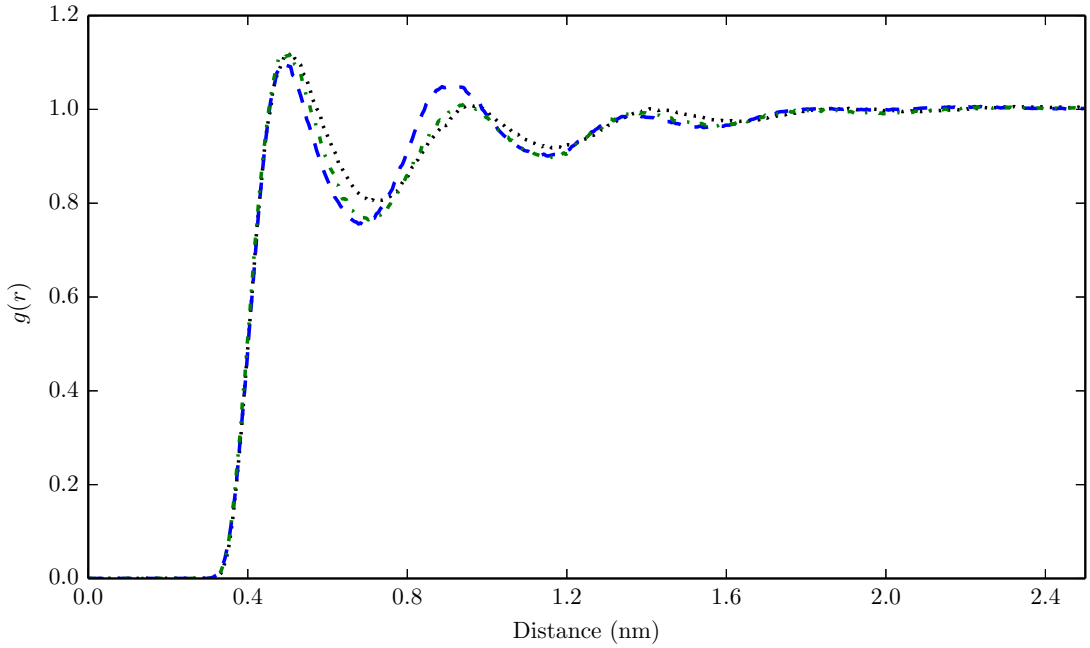


(a) A - A

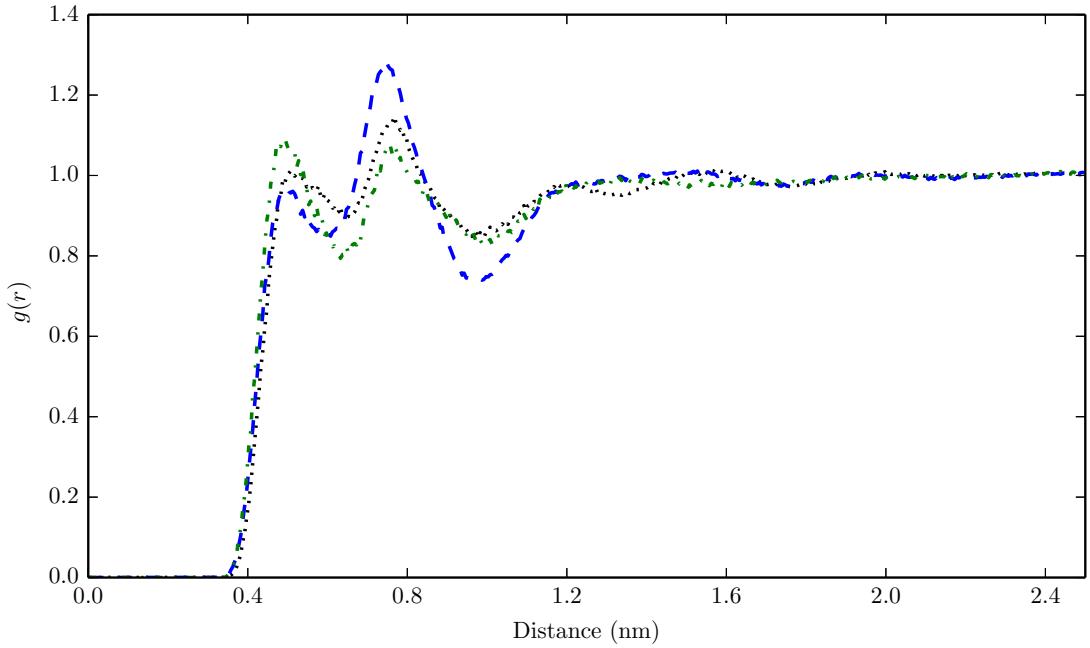


(b) A - M2

Figure B.8: Intermolecular RDFs compared across different scale models. Colours: Black-Atomistic, Blue-Hybrid, Green-Coarse

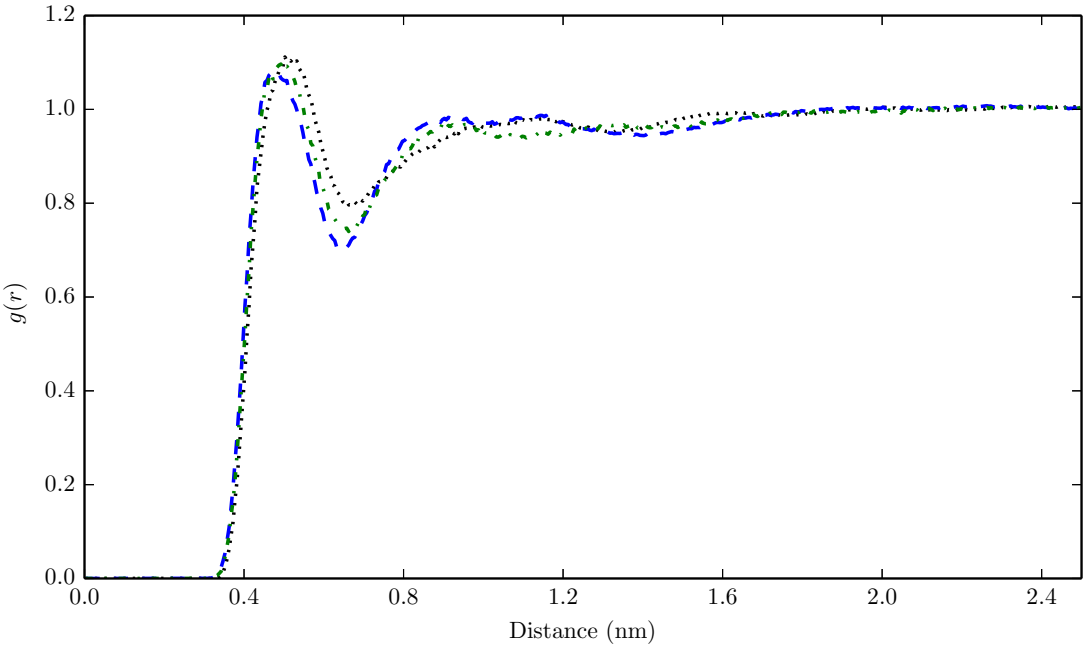


(c) A - M3

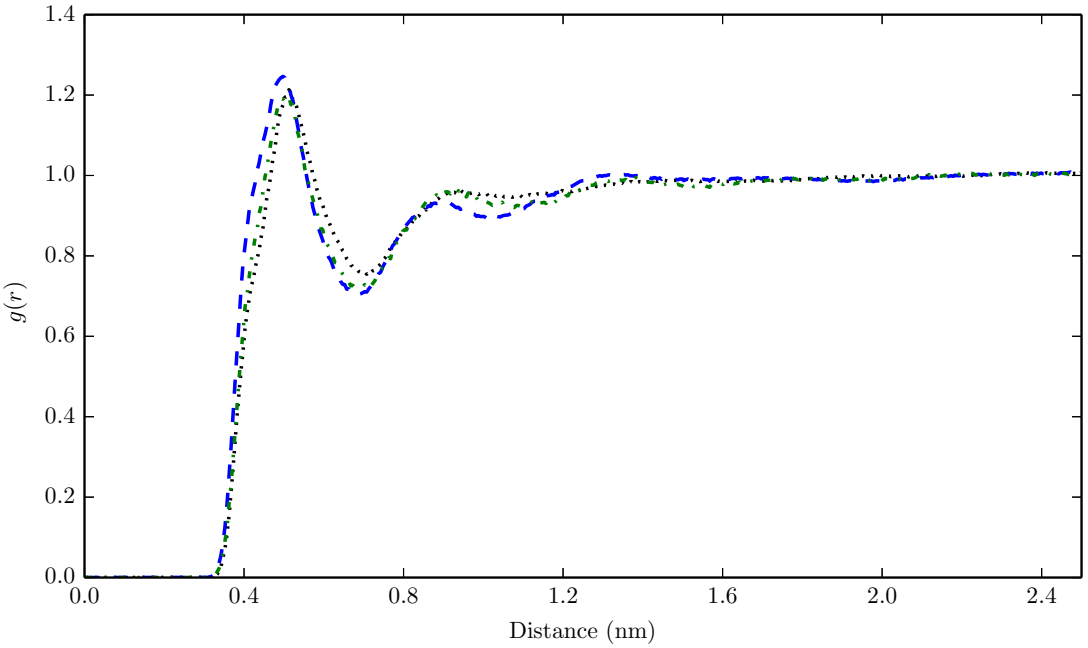


(d) M2 - M2

Figure B.8: Intermolecular RDFs compared across different scale models. Colours: Black-Atomistic, Blue-Hybrid, Green-Coarse



(e) M2 - M3



(f) M3 - M3

Figure B.8: Intermolecular RDFs compared across different scale models. Colours: Black-Atomistic, Blue-Hybrid, Green-Coarse

B.3 Transferability Results

The transferability of the forcefields are evaluated by comparing the A-A intermolecular rdf at 350, 400, 450 and 500 K.

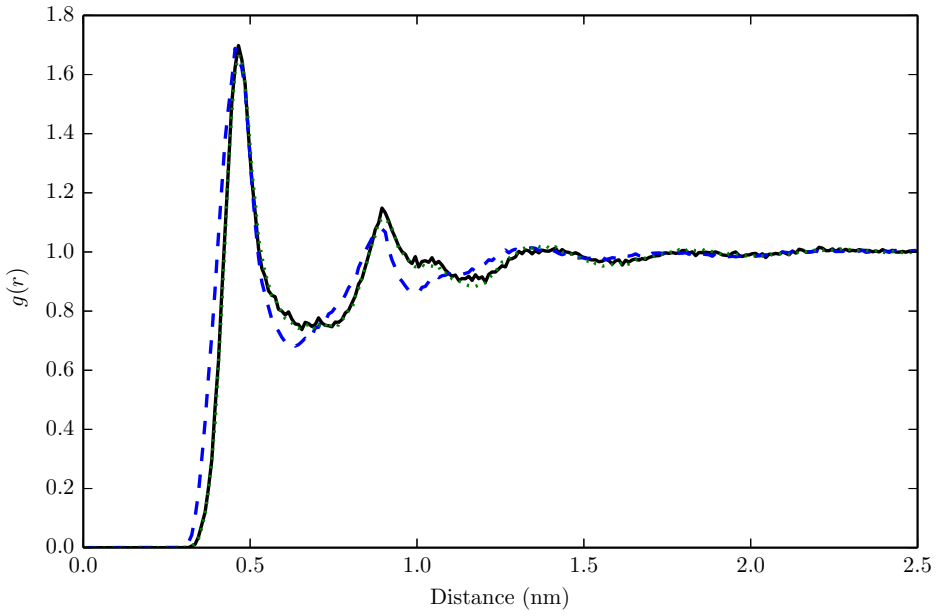


Figure B.9: T=350 K

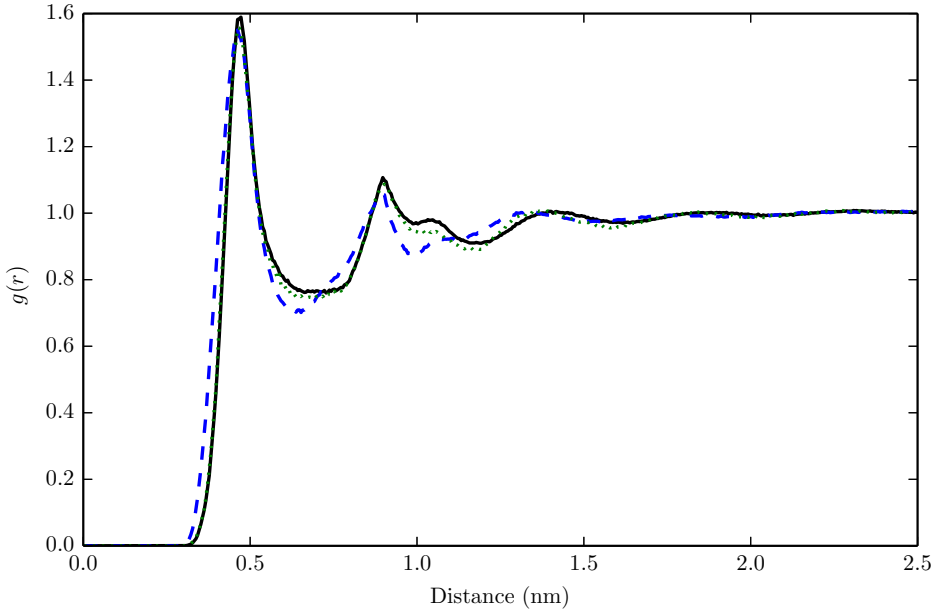


Figure B.10: T=400 K

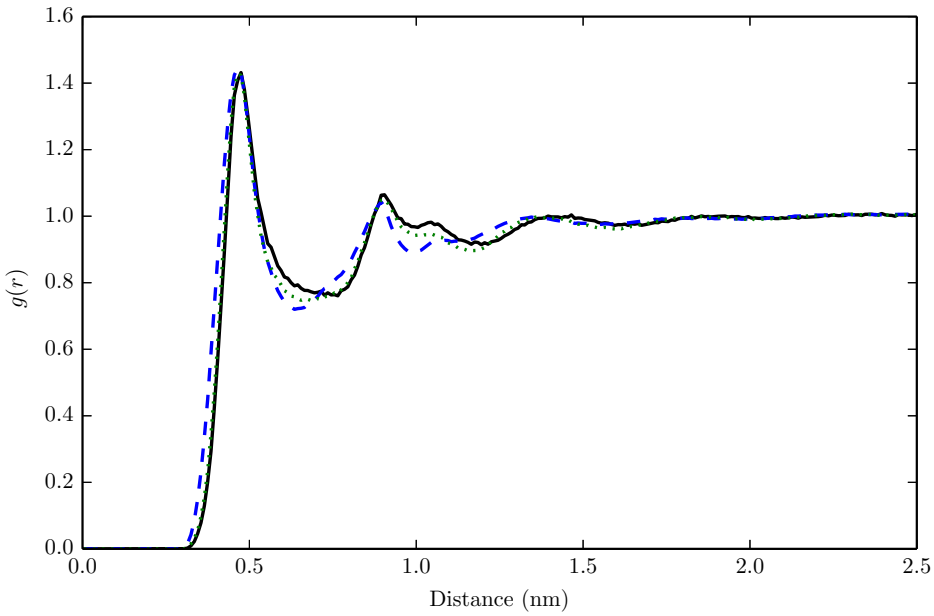


Figure B.11: T=450 K

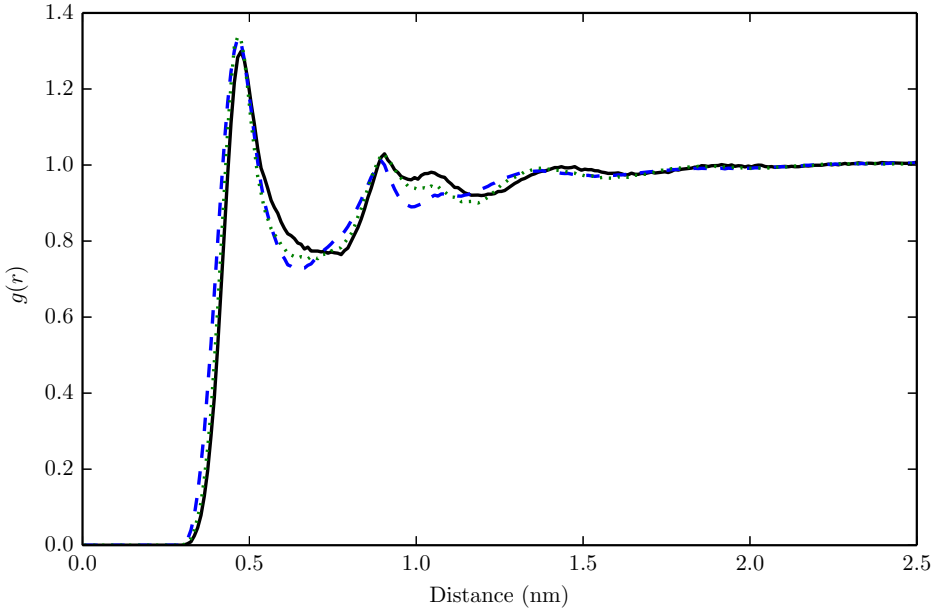
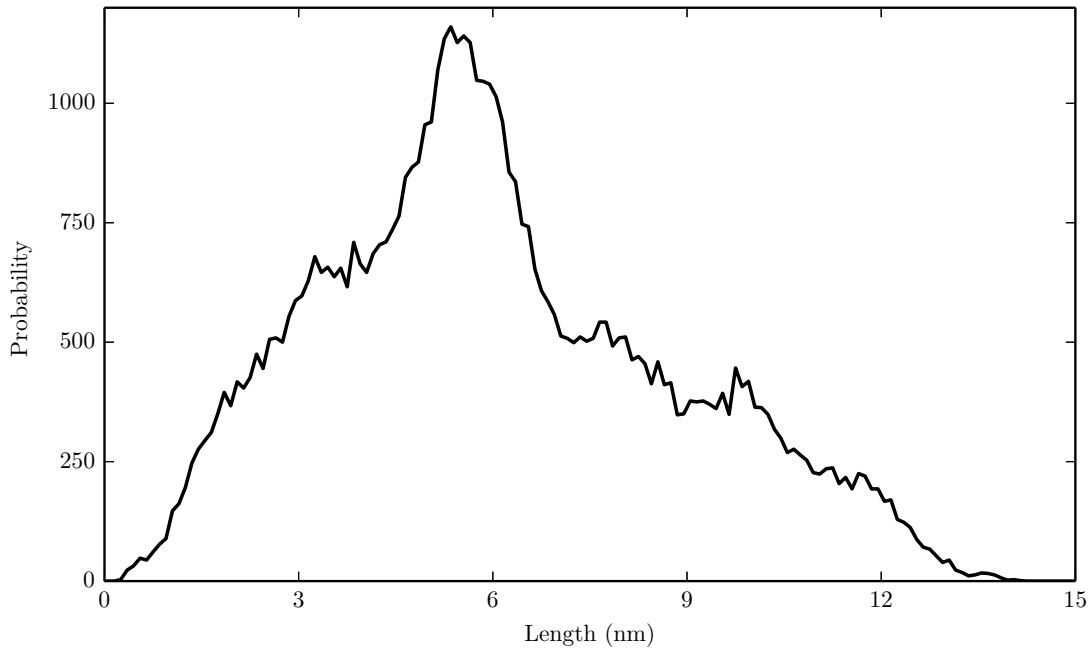


Figure B.12: T=500 K

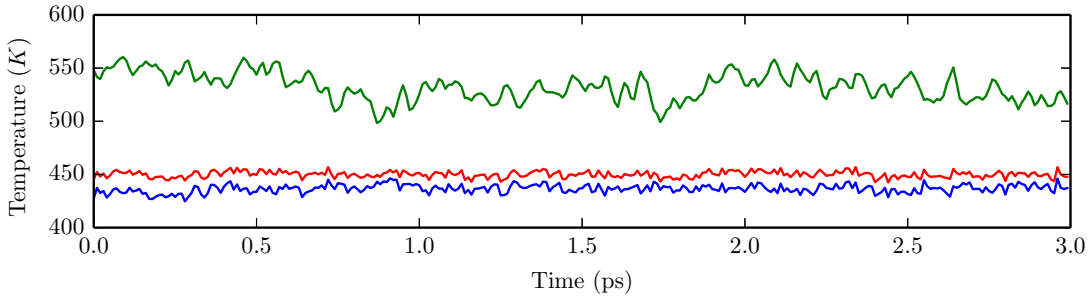
B.4 Structural Results

The end to end distance distribution for atomistic polyamide at 500 K

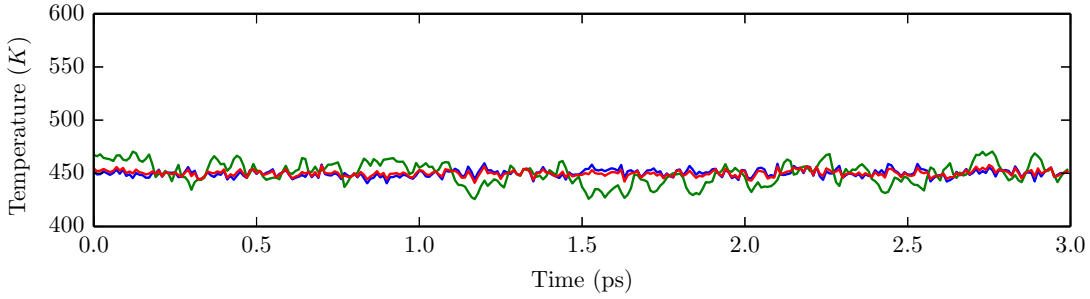


B.5 Thermostat results

The temperature of the atoms and beads, and the mean average of these in a hybrid simulation were recorded every 0.01 ps. This was repeated using a single Berendsen for the entire system, or a separate Berendsen thermostat for the atomistic and coarse-grain parts.



(a) Single thermostat



(b) Double thermostat

Figure B.13: Comparison of the temperature of the system when using either a single or separate thermostats. Colours: Green - Coarse-grained, Blue-Atomistic, Red-Average

B.6 Hydrogen Bond Structures

The contour map for hydrogen bond length and angle at 350, 400 and 500 K are presented.

B.7 Hydrogen Bond Dynamics

In approximating the time scaling factor for intermittent hydrogen bond lifetime between the two models, a factor a was found based on the total amount of trajectory

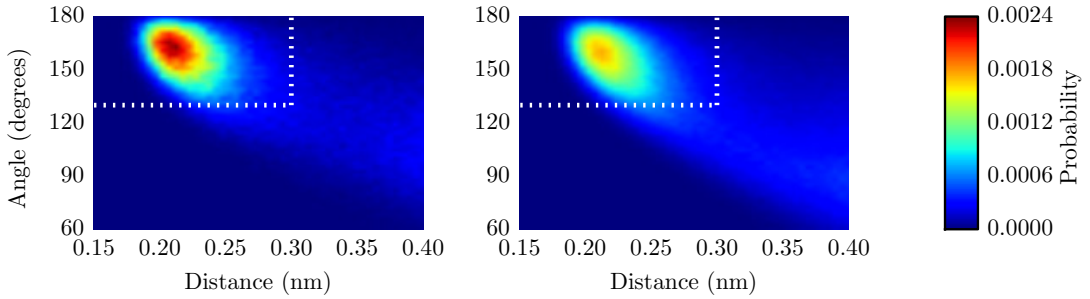


Figure B.14: T=350 K

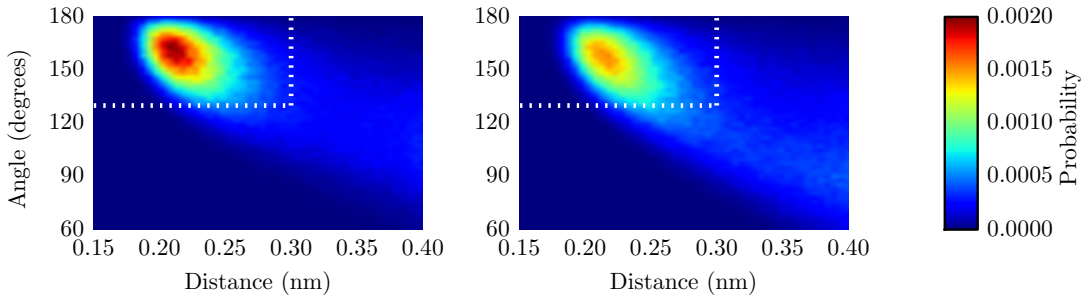


Figure B.15: T=400 K

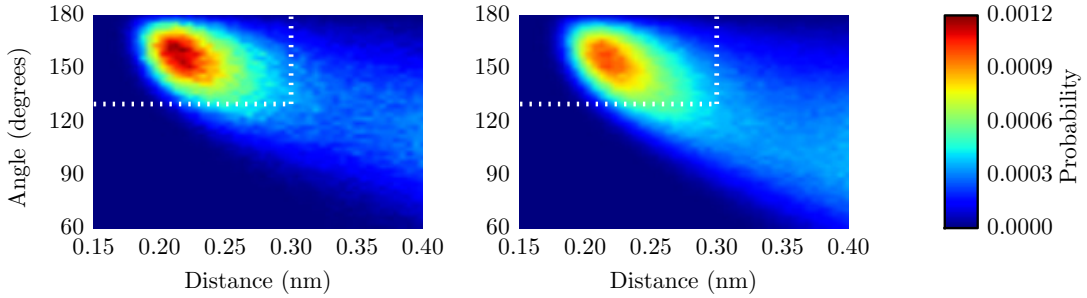


Figure B.16: T=500 K

available. To check that this was not related to the amount of trajectory used in the analysis, the value of a that would be found by truncating the trajectory as $t = \tau$, as defined in Equation B.3, was calculated.

$$a(\tau) = \min \left(\sum_{t=0}^{t=\tau} \left(C_{I,atom}(t) - C_{I,hybrid}(at) \right)^2 \right) \tag{B.3}$$

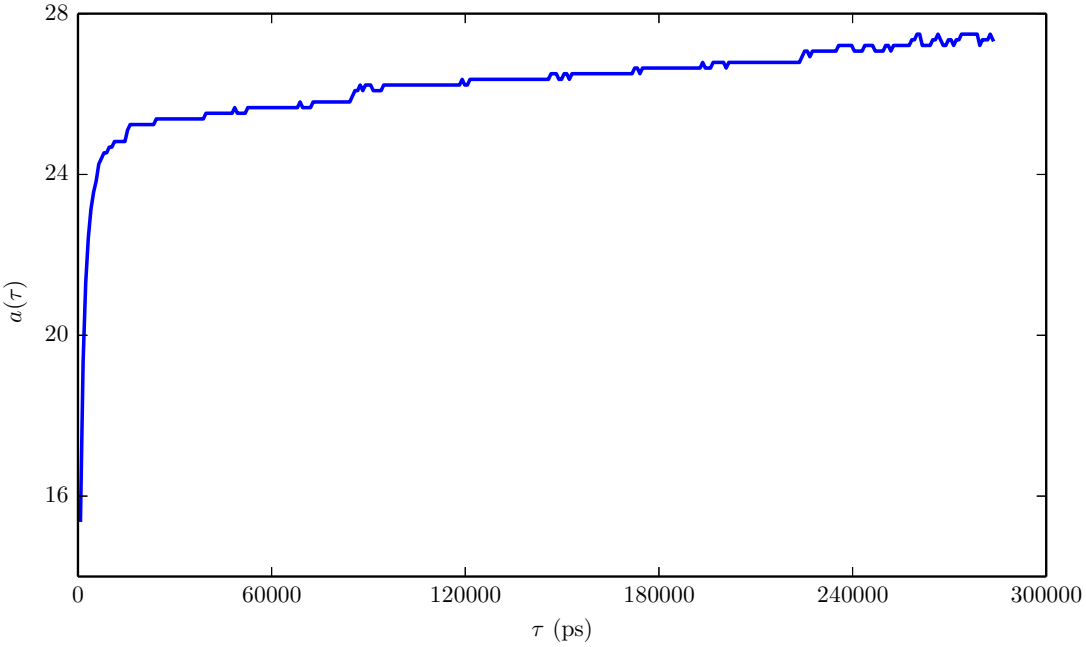


Figure B.17: Comparison of fitting factor found against time used in fitting

Appendix C

Supporting information for: A systematic approach to dual scale models

C.1 Forcefield details

C.1.1 ITP files

The itp files for the hybrid models are given below. These were used with Gromacs v5.

AA model

forcefields/aa/ffnonbonded.itp

```
[ defaults ]
 1 1 no

[ atomtypes ]
;; What things should be
;; Sigma & Epsilon versions
; opls_155 HO 1 1.00800    0.418 A 0.00000e+00 0.00000e+00
; opls_154 OH 8 15.99940 -0.683 A 3.12000e-01 7.11280e-01
; opls_140 HC 1 1.00800    0.060 A 2.50000e-01 1.25520e-01
; opls_157 CT 6 12.01100   0.145 A 3.50000e-01 2.76144e-01
; opls_135 CT 6 12.01100 -0.180 A 3.50000e-01 2.76144e-01
; These are 4*eps*sigma^12 and 4*eps*sigma^6
 opls_155 HO 1 1.00800    0.418 A 0.00000e+00 0.00000e+00
 opls_154 OH 8 15.99940 -0.683 A 2.62439e-03 2.42078e-06
 opls_140 HC 1 1.00800    0.060 A 1.22578e-04 2.99263e-08
 opls_157 CT 6 12.01100   0.145 A 2.03050e-03 3.73261e-06
 opls_135 CT 6 12.01100 -0.180 A 2.03050e-03 3.73261e-06
;; Non interacting dudes
 Cni       CT 6 12.01100   0.000 A 0.00000e+00 0.00000e+00
 Hni       HC 1 1.00800    0.000 A 0.00000e+00 0.00000e+00
;; The beads, off by default, turned on later
 A         A  2 0.00       0.0   A 0.00000e+00 0.00000e+00
 B         B  2 0.00       0.0   A 0.00000e+00 0.00000e+00
 C         C  2 0.00       0.0   A 0.00000e+00 0.00000e+00

[ nonbond_params ]
;; These are tabulated potentials
;; A-A is left out, done atomisticly
 A B 1 1.0 1.0
```

```
 A C 1 1.0 1.0
 B B 1 1.0 1.0
 B C 1 1.0 1.0
 C C 1 1.0 1.0

[ pair_types ]
;; For adding intramolecular LJ interactions back in
;; HO - anything is zero for now
 opls_155 Cni       1 0.00000e+00 0.00000e+00
 opls_155 Hni       1 0.00000e+00 0.00000e+00
 opls_155 opls_140  1 0.00000e+00 0.00000e+00
 opls_155 opls_135  1 0.00000e+00 0.00000e+00
;; CT-OH
 Cni opls_154  1 2.30852e-03 3.00596e-06
;; HC-OH
 opls_154 opls_140 1 5.67179e-04 2.69156e-07
 opls_154 Hni      1 5.67179e-04 2.69156e-07
;; CT-CT
 Cni Cni       1 2.03050e-03 3.73261e-06
 Cni opls_157  1 2.03050e-03 3.73261e-06
 Cni opls_135  1 2.03050e-03 3.73261e-06
;; HC-HC
 opls_140 opls_140 1 1.22578e-04 2.99263e-08
 Hni      Hni      1 1.22578e-04 2.99263e-08
 opls_140 Hni      1 1.22578e-04 2.99263e-08
;; CT-HC
 opls_157 opls_140 1 4.98893e-04 3.34220e-07
 opls_157 Hni      1 4.98893e-04 3.34220e-07
 opls_135 opls_140 1 4.98893e-04 3.34220e-07
 opls_135 Hni      1 4.98893e-04 3.34220e-07
 Cni      opls_140 1 4.98893e-04 3.34220e-07
 Cni      Hni      1 4.98893e-04 3.34220e-07
```

forcefields/aa/ffbonded.itp

```
[ bondtypes ]
 HO OH 1 0.09450 462750.4 ; SUG(OL) wlj mod 0.96-> 0.945
 CT OH 1 0.14100 267776.0 ;
 CT CT 1 0.15290 224262.4 ; CHARMM 22 parameter file
 CT HC 1 0.10900 284512.0 ; CHARMM 22 parameter file

[ angletypes ]
 CT OH HO 1 108.500 460.240 ;
 CT CT OH 1 109.500 418.400 ;
 CT CT CT 1 112.700 488.273 ; CHARMM 22 parameter file
 HC CT HC 1 107.800 276.144 ; CHARMM 22 parameter file
 CT CT HC 1 110.700 313.800 ; CHARMM 22 parameter file
 HC CT OH 1 109.500 292.880 ;

[ dihedraltypes ]
 CT CT OH HO 3 -0.44350 3.83255 0.72801 -4.11705 0.00000 0.00000 ; alcohols AA
 HC CT OH HO 3 0.94140 2.82420 0.00000 -3.76560 0.00000 0.00000 ; alcohols AA
 HC CT CT OH 3 0.97905 2.93716 0.00000 -3.91622 0.00000 0.00000 ; alcohols, ethers AA
 CT CT CT OH 3 2.87441 0.58158 2.09200 -5.54799 0.00000 0.00000 ; alcohols, ethers AA
 HC CT CT HC 3 0.62760 1.88280 0.00000 -2.51040 0.00000 0.00000 ; hydrocarbon *new* 11/99
 CT CT CT HC 3 0.62760 1.88280 0.00000 -2.51040 0.00000 0.00000 ; hydrocarbon all-atom
 CT CT CT CT 3 2.92880 -1.46440 0.20920 -1.67360 0.00000 0.00000 ; hydrocarbon all-atom
```

forcefields/aa/virtual_oct.itp

```
[ moleculetype ]
; Name            nrexcl
OcOH                3

[ atoms ]
;   nr       type  resnr residue  atom   cgnr     charge       mass   ; Bead
     1   opls_155      1   OcOH     HO      1      0.418      1.008   ; A qtot 0.418
     2   opls_154      1   OcOH     OH      1     -0.683     15.9994  ; A qtot -0.265
     3   opls_140      1   OcOH     HA1     1       0.06      1.008   ; A qtot -0.205
     4   opls_140      1   OcOH     HA2     1       0.06      1.008   ; A qtot -0.145
     5   opls_157      1   OcOH     CA      1      0.145     12.011   ; A qtot 0
     6   opls_140      1   OcOH     HB1     2       0.06      1.008   ; A qtot 0.06
     7   opls_140      1   OcOH     HB2     2       0.06      1.008   ; A qtot 0.12
     8   opls_135      1   OcOH     CB      2      -0.12     12.011   ; A qtot 0
     9   Hni           1   OcOH     HC1     3       0.00      1.008   ; B
    10   Hni           1   OcOH     HC2     3       0.00      1.008   ; B
    11   Cni           1   OcOH     CC      3       0.00     12.011   ; B
```

```
     12   Hni          1   OcOH    HD1      4       0.00      1.008   ; B
     13   Hni          1   OcOH    HD2      4       0.00      1.008   ; B
     14   Cni          1   OcOH     CD      4       0.00     12.011   ; B
     15   Hni          1   OcOH    HE1      5       0.00      1.008   ; B
     16   Hni          1   OcOH    HE2      5       0.00      1.008   ; B
     17   Cni          1   OcOH     CE      5       0.00     12.011   ; B
     18   Hni          1   OcOH    HF1      6       0.00      1.008   ; C
     19   Hni          1   OcOH    HF2      6       0.00      1.008   ; C
     20   Cni          1   OcOH     CF      6       0.00     12.011   ; C
     21   Hni          1   OcOH    HG1      7       0.00      1.008   ; C
     22   Hni          1   OcOH    HG2      7       0.00      1.008   ; C
     23   Cni          1   OcOH     CG      7       0.00     12.011   ; C
     24   Hni          1   OcOH    HH1      8       0.00      1.008   ; C
     25   Hni          1   OcOH    HH2      8       0.00      1.008   ; C
     26   Hni          1   OcOH    HH3      8       0.00      1.008   ; C
     27   Cni          1   OcOH     CH      8       0.00     12.011   ; C
     28   A            1   OcOH      A      9       0.00      0.000
     29   B            1   OcOH      B     10       0.00      0.000
     30   C            1   OcOH      C     11       0.00      0.000

[ bonds ]
;   ai    aj  funct            c0          c1          c2          c3
     1     2      1
     2     5      1
     3     5      1
     4     5      1
     5     8      1
     6     8      1
     7     8      1
     8    11      1
     9    11      1
    10    11      1
    11    14      1
    12    14      1
    13    14      1
    14    17      1
    15    17      1
    16    17      1
    17    20      1
    18    20      1
    19    20      1
    20    23      1
    21    23      1
    22    23      1
    23    27      1
    24    27      1
    25    27      1
    26    27      1

[ angles ]
;   ai    aj    ak  funct            c0          c1          c2          c3
     1     2     5      1
     2     5     3      1
     2     5     4      1
     2     5     8      1
     3     5     4      1
     3     5     8      1
     4     5     8      1
     5     8     6      1
     5     8     7      1
     5     8    11      1
     6     8     7      1
     6     8    11      1
     7     8    11      1
     8    11     9      1
     8    11    10      1
     8    11    14      1
     9    11    10      1
     9    11    14      1
    10    11    14      1
    11    14    12      1
    11    14    13      1
    11    14    17      1
    12    14    13      1
    12    14    17      1
    13    14    17      1
    14    17    15      1
    14    17    16      1
    14    17    20      1
    15    17    16      1
```

```
    15     17     20      1
    16     17     20      1
    17     20     18      1
    17     20     19      1
    17     20     23      1
    18     20     19      1
    18     20     23      1
    19     20     23      1
    20     23     21      1
    20     23     22      1
    20     23     27      1
    21     23     22      1
    21     23     27      1
    22     23     27      1
    23     27     24      1
    23     27     25      1
    23     27     26      1
    24     27     25      1
    24     27     26      1
    25     27     26      1

[ dihedrals ]
;   ai     aj     ak     al funct            c0            c1            c2            c3
                  c4            c5
     1      2      5      3     3
     1      2      5      4     3
     1      2      5      8     3
     2      5      8      6     3
     2      5      8      7     3
     2      5      8     11     3
     3      5      8      6     3
     3      5      8      7     3
     3      5      8     11     3
     4      5      8      6     3
     4      5      8      7     3
     4      5      8     11     3
     5      8     11      9     3
     5      8     11     10     3
     5      8     11     14     3
     6      8     11      9     3
     6      8     11     10     3
     6      8     11     14     3
     7      8     11      9     3
     7      8     11     10     3
     7      8     11     14     3
     8     11     14     12     3
     8     11     14     13     3
     8     11     14     17     3
     9     11     14     12     3
     9     11     14     13     3
     9     11     14     17     3
    10     11     14     12     3
    10     11     14     13     3
    10     11     14     17     3
    11     14     17     15     3
    11     14     17     16     3
    11     14     17     20     3
    12     14     17     15     3
    12     14     17     16     3
    12     14     17     20     3
    13     14     17     15     3
    13     14     17     16     3
    13     14     17     20     3
    14     17     20     18     3
    14     17     20     19     3
    14     17     20     23     3
    15     17     20     18     3
    15     17     20     19     3
    15     17     20     23     3
    16     17     20     18     3
    16     17     20     19     3
    16     17     20     23     3
    17     20     23     21     3
    17     20     23     22     3
    17     20     23     27     3
    18     20     23     21     3
    18     20     23     22     3
    18     20     23     27     3
    19     20     23     21     3
    19     20     23     22     3
```

```
    19     20     23     27      3
178 20     23     27     24      3
    20     23     27     25      3
    20     23     27     26      3
    21     23     27     24      3
    21     23     27     25      3
183 21     23     27     26      3
    22     23     27     24      3
    22     23     27     25      3
    22     23     27     26      3

188
[ pairs ] ;; HO has no sigma, only charge
     1      3      1 ; HO HA1
     1      4      1 ; HO HA2
     1      8      1 ; HO CB
193  2      6      1 ; OH HB1
     2      7      1 ; OH HB2
     2     11      1 ; OH CC
     3      6      1 ; HA1 HB2
     3      7      1
198  3     11      1
     4      6      1
     4      7      1
     4     11      1
     5      9      1
203  5     10      1
     5     14      1
     6      9      1
     6     10      1
     6     14      1
208  7      9      1
     7     10      1
     7     14      1
     8     12      1
     8     13      1
213  8     17      1
     9     12      1
     9     13      1
     9     17      1
    10     12      1
218 10     13      1
    10     17      1
    11     15      1
    11     16      1
    11     20      1
223 12     15      1
    12     16      1
    12     20      1
    13     15      1
    13     16      1
228 13     20      1
    14     18      1
    14     19      1
    14     23      1
    15     18      1
233 15     19      1
    15     23      1
    16     18      1
    16     19      1
    16     23      1
238 17     21      1
    17     22      1
    17     27      1
    18     21      1
    18     22      1
243 18     27      1
    19     21      1
    19     22      1
    19     27      1
    20     24      1
248 20     25      1
    20     26      1
    21     24      1
    21     25      1
    21     26      1
253 22     24      1
    22     25      1
    22     26      1
  1 6   1
```

```
    1  7   1
258 1  9   1
    1  10  1
    1  11  1
    1  12  1
    1  13  1
263 1  14  1
    1  15  1
    1  16  1
    1  17  1
    1  18  1
268 1  19  1
    1  20  1
    1  21  1
    1  22  1
    1  23  1
273 1  24  1
    1  25  1
    1  26  1
    1  27  1
    2  9   1
278 2  10  1
    2  12  1
    2  13  1
    2  14  1
    2  15  1
283 2  16  1
    2  17  1
    2  18  1
    2  19  1
    2  20  1
288 2  21  1
    2  22  1
    2  23  1
    2  24  1
    2  25  1
293 2  26  1
    2  27  1
    3  9   1
    3  10  1
    3  12  1
298 3  13  1
    3  14  1
    3  15  1
    3  16  1
    3  17  1
303 3  18  1
    3  19  1
    3  20  1
    3  21  1
    3  22  1
308 3  23  1
    3  24  1
    3  25  1
    3  26  1
    3  27  1
313 4  9   1
    4  10  1
    4  12  1
    4  13  1
    4  14  1
318 4  15  1
    4  16  1
    4  17  1
    4  18  1
    4  19  1
323 4  20  1
    4  21  1
    4  22  1
    4  23  1
    4  24  1
328 4  25  1
    4  26  1
    4  27  1
    5  12  1
    5  13  1
333 5  15  1
    5  16  1
    5  17  1
    5  18  1
```

```
    5  19   1
338 5  20   1
    5  21   1
    5  22   1
    5  23   1
    5  24   1
343 5  25   1
    5  26   1
    5  27   1
    6  12   1
    6  13   1
348 6  15   1
    6  16   1
    6  17   1
    6  18   1
    6  19   1
353 6  20   1
    6  21   1
    6  22   1
    6  23   1
    6  24   1
358 6  25   1
    6  26   1
    6  27   1
    7  12   1
    7  13   1
363 7  15   1
    7  16   1
    7  17   1
    7  18   1
    7  19   1
368 7  20   1
    7  21   1
    7  22   1
    7  23   1
    7  24   1
373 7  25   1
    7  26   1
    7  27   1
    8  15   1
    8  16   1
378 8  18   1
    8  19   1
    8  20   1
    8  21   1
    8  22   1
383 8  23   1
    8  24   1
    8  25   1
    8  26   1
    8  27   1
388 9  15   1
    9  16   1
    9  18   1
    9  19   1
    9  20   1
393 9  21   1
    9  22   1
    9  23   1
    9  24   1
    9  25   1
398 9  26   1
    9  27   1
    10 15   1
    10 16   1
    10 18   1
403 10 19   1
    10 20   1
    10 21   1
    10 22   1
    10 23   1
408 10 24   1
    10 25   1
    10 26   1
    10 27   1
    11 18   1
413 11 19   1
    11 21   1
    11 22   1
    11 23   1
```

```
  11 24  1
  11 25  1
  11 26  1
  11 27  1
  12 18  1
  12 19  1
  12 21  1
  12 22  1
  12 23  1
  12 24  1
  12 25  1
  12 26  1
  12 27  1
  13 18  1
  13 19  1
  13 21  1
  13 22  1
  13 23  1
  13 24  1
  13 25  1
  13 26  1
  13 27  1
  14 21  1
  14 22  1
  14 24  1
  14 25  1
  14 26  1
  14 27  1
  15 21  1
  15 22  1
  15 24  1
  15 25  1
  15 26  1
  15 27  1
  16 21  1
  16 22  1
  16 24  1
  16 25  1
  16 26  1
  16 27  1
  17 24  1
  17 25  1
  17 26  1
  18 24  1
  18 25  1
  18 26  1
  19 24  1
  19 25  1
  19 26  1


#define mapping virtual_sitesn
[ mapping ]
;; Atom no., Type of VS (2=COM), Atoms contained
  28  2  1 2 3 4 5 6 7 8
  29  2  9 10 11 12 13 14 15 16 17
  30  2  18 19 20 21 22 23 24 25 26 27

;; Make sure virtual sites don't interact with eachother
[ exclusions ]
;; Atom no., their exclusions
   28  29 30
   29  28 30
   30  28 29
```

UA model

forcefields/ua/H_ffgmxnb.itp

```
[ defaults ]
;; Func type, 1=LJ, 2=Buck ; Mixing rule 2=LB ; genpairs ; fudge LJ ; fudge QQ
 1                          1                  no

[ atomtypes ]
;name        mass       charge   ptype            c6           c12
;; from lipid.itp
```

```
 LP2 14.0270 0.000 A 5.87400e-03 2.26500e-05 ;RB CH2, Bergers LJ
 LP3 15.0350 0.000 A 8.77700e-03 3.38500e-05 ;RB CH3, Bergers LJ
;;
   OA     15.99940        0.000        A    0.22617E-02    0.15062E-05
   HO      1.00800        0.000        A    0.00000E+00    0.00000E+00
;; New atomtypes for hybrid simulation
;; These are generally the old atomtypes prefixed by the
;; bead that they appear in
 VS_LP2 LP2 14.027 0.0 A 0.0 0.0
 VS_OA OA 15.9994 0.0 A 0.0 0.0
 VS_HO HO 1.008 0.0 A 0.0 0.0
 VS_LP3 LP3 15.035 0.0 A 0.0 0.0
;; Begin the atoms that are actually beads
 X X 0.0 0.0 A 0.0 0.0
 Y Y 0.0 0.0 A 0.0 0.0
 Z Z 0.0 0.0 A 0.0 0.0


[ nonbond_params ]
  ; i     j func          c6           c12
;; from lipid.itp
 LP2 LP2 1 5.87400e-03 2.26500e-05
 LP2 LP3 1 7.18000e-03 2.76900e-05
 LP2 OA 1 4.536057e-03 7.295106e-06
 LP2 HO 1 0.000000e+00 0.000000e+00
 LP3 OA 1 4.480623e-03 6.275917e-06
 LP3 HO 1 0.000000e+00 0.000000e+00
   OA   OA     1 0.22617E-02    0.15062E-05

X Y 1 1.0 1.0
X Z 1 1.0 1.0
Y Y 1 1.0 1.0
Y Z 1 1.0 1.0
Z Z 1 1.0 1.0
```

forcefields/ua/H_ffgmxbon.itp

```
[ bondtypes ]
  ; i     j func         b0          kb
   HO   OA     1    0.10000      313800.


[ dihedraltypes ]
  ; j     k func       phi0          cp    mult
  LP2   LP2     3    9.2789    12.156   -13.120  -3.0597  26.240  -31.495


#define ANG_180_0 180 0
#define DIH_0_0_2 0 0 2
```

forcefields/ua/H_UA_oct.itp

```
[ moleculetype ]
; Name nrexcl
Octanol 3


[ atoms ]
      1 HO              1 OCT   HO      8 0.407
      2 OA              1 OCT   OH      8 -0.563
      3 LP2             1 OCT   CA      8 0.156
      4 LP2             1 OCT   CB      7 0.0
      5 VS_LP2          1 OCT   CC      6 0.0
      6 VS_LP2          1 OCT   CD      5 0.0
      7 VS_LP2          1 OCT   CE      4 0.0
      8 VS_LP2          1 OCT   CF      3 0.0
      9 VS_LP2          1 OCT   CG      2 0.0
     10 VS_LP3          1 OCT   CH      1 0.0
     11 X               1 OCT   X1      9   0.00 0.00 ; VS
     12 Y               1 OCT   Y2      10 0.00 0.00 ; VS
     13 Z               1 OCT   Z3      11 0.00 0.00 ; VS


[ bonds ]
; ai  aj func b0 cb
10  9  1 0.15300E+00 0.33470E+06
9   8  1 0.15300E+00 0.33470E+06
```

```
8   7  1  0.15300E+00  0.33470E+06
7   6  1  0.15300E+00  0.33470E+06
6   5  1  0.15300E+00  0.33470E+06
5   4  1  0.15300E+00  0.33470E+06
4   3  1  0.15300E+00  0.33470E+06
3   2  1  0.12300E+00  0.50210E+06
2   1  1


[ angles ]
; ai  aj  ak  func  th0  cth
1  2  3  1  0.11100E+03  0.46020E+03
2  3  4  1  0.11100E+03  0.46020E+03
3  4  5  1  0.11100E+03  0.46020E+03
4  5  6  1  0.11100E+03  0.46020E+03
5  6  7  1  0.11100E+03  0.46020E+03
6  7  8  1  0.11100E+03  0.46020E+03
7  8  9  1  0.12100E+03  0.50210E+03
8  9  10  1  0.10950E+03  0.39748E+03


[ dihedrals ]
; ai  aj  ak  al  func
10  9  8  7   3
9   8  7  6   3
8   7  6  5   3
7   6  5  4   3
6   5  4  3   3

;ai  aj  ak  al  func  phi0  cphi  mult
5  4  3  2  1  0.000  1.255  3
4  3  2  1  1  0.000  1.255  3

[ pairs ]
; Begin pairs from hybrid method
 1      5       1   0.000000E+00  0.000000E+00
 1      6       1   0.000000E+00  0.000000E+00
 1      7       1   0.000000E+00  0.000000E+00
 2      6       1   3.644890E-03  5.840842E-06
 2      7       1   3.644890E-03  5.840842E-06
 3      7       1   5.874000E-03  2.265000E-05
 1      8       1   0.000000E+00  0.000000E+00
 1      9       1   0.000000E+00  0.000000E+00
 1      10      1   0.000000E+00  0.000000E+00
 2      8       1   3.644890E-03  5.840842E-06
 2      9       1   3.644890E-03  5.840842E-06
 2      10      1   4.455439E-03  7.140369E-06
 3      8       1   5.874000E-03  2.265000E-05
 3      9       1   5.874000E-03  2.265000E-05
 3      10      1   7.180258E-03  2.768939E-05
 4      8       1   5.874000E-03  2.265000E-05
 4      9       1   5.874000E-03  2.265000E-05
 4      10      1   7.180258E-03  2.768939E-05
 5      9       1   5.874000E-03  2.265000E-05
 5      10      1   7.180258E-03  2.768939E-05
 6      10      1   7.180258E-03  2.768939E-05


[ exclusions ]
  11    12 13
  12    11 13
  13    11 12

[ virtualsitesn ]
  11   2   1 2 3 4
  12   2   5 6 7
  13   2   8 9 10
```

C.2 Distribution of forces

The distribution of the magnitude of forces on atoms, normalised by their mass, is shown in Figure C.1. The bimodal distribution is caused by some atoms having charges, and hence a higher total average force.

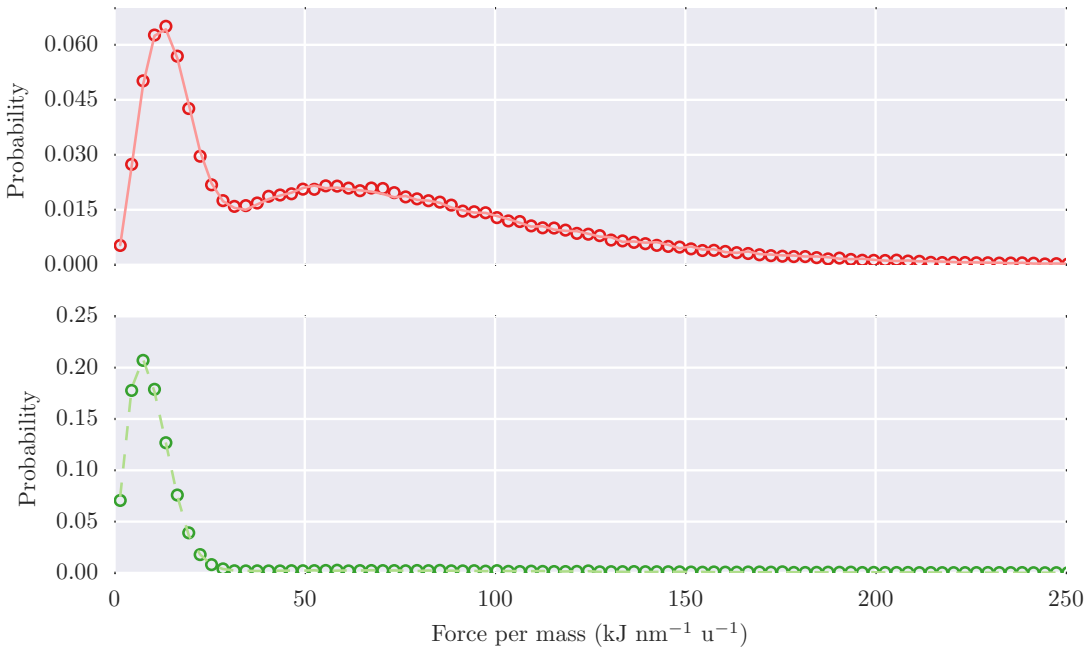


Figure C.1: Distribution of forces

Appendix D

IBIsCO Source code listing

This code was used in Chapters 6 and 5 to simulate the dual scale models. This code originally started as a copy of the IBIsCO code[1], which was designed for simulating IBI CG models, with the changes made presented in Chapter 7.

src/main.F90

```
PROGRAM IBISCO
  USE MODULEPARSING
  USE MTS
  USE VAR
  IMPLICIT NONE

  INTEGER :: I, A

  WRITE(*,*) '-----------------------------------------'
  WRITE(*,*) 'Hybrid IBIsCO! Revision 80 - Now with MTS'
  WRITE(*,*) '-----------------------------------------'

  OPEN ( 115 , FILE = 's-md.out')
  OPEN ( 116 , FILE = 's-md.tp')
  OPEN ( 113 , FILE = 's-md.trj', form='UNFORMATTED', access='SEQUENTIAL')

  WRITE( 115, *) 'IBIsCO Revision 80'
  OPEN (1, FILE='ERROR')

  ISTOP = 0

  !DEFINE REDUCED UNITS
  CALL UNIT()

  !Read control parameters
  NVIRTA = 0 !By default no virtual sites
  CALL RDCONTROL()
  IF (ISTOP == 1) STOP 'Failed in RDCONTROL'

  CALL ALLOCATEVAR()

  CALL RDINTERACT()
  IF (ISTOP == 1) STOP 'Failed in RDINTERACT'

  CALL  RDCOOR()
  IF (ISTOP == 1) STOP 'Failed in RDCOOR'

  CALL ALLOCATEVAR2()

  NCOARSE = 0
  IF(IBRDESCR .EQ. 0) THEN
     CALL RDVIRTUAL()

     IF(ISTOP .eq. 1) STOP 'Failed at RDVIRTUAL'
  END IF

  IF(IBRDESCR .EQ. 0) THEN
     CALL MAKE_LISTS()
```

```
      IF(ISTOP .eq. 1) STOP 'Failed at MAKE_LISTS'
   ELSE
      !Nonhybrid settings, single list containing all atoms
      NUMATOMS = NATOMS
      ALLOCATE(ATOM(NUMATOMS))
      DO I=1,NATOMS
         ATOM(I) = I
      END DO
   END IF

   IF(IBRDESCR .EQ. 0) THEN
      CALL VIRTUAL_DEF()
   END IF

   CALL SHIFT ()

   MAXNAB_ATOM = 1000 !Max number of neighbours for a single atom
   ALLOCATE(LIST_ATOM(MAXNAB_ATOM,NUMATOMS), &
         CELL_ATOM(NUMATOMS), &
         LCLIST_ATOM(NUMATOMS))
   IF(IBRDESCR .eq. 0) THEN
      MAXNAB_BEAD = 1000
      ALLOCATE(LIST_BEAD(MAXNAB_BEAD,NCOARSE), &
            CELL_BEAD(NCOARSE), &
            LCLIST_BEAD(NCOARSE))
   END IF
   ALLOCATE(NNEBS(NITEMS))

   !MAKE TABLE FORCES
   CALL FTABLE ()

   ! GIVE THE VELOCITIES IF THE INITIAL TIME IS ZERO
   IF (RESTART == 1) THEN
      CALL COMVEL ()
      CALL SCALEV ()
   END IF

   ! CALCULATE THE INITIAL TEMPERATURE AND KINETIC ENERGY
   EK(1) = 0.0
   EK(2) = 0.0

   MKTEMP_ATOM = 2.0 / REAL(3.0 * (NUMATOMS - 1))
   MKTEMP_BEAD = 2.0 / REAL(3.0 * (NUMBEADS - 1))
   DO A = 1, NUMATOMS
      I = ATOM(A)
      EK(1) = EK(1) + MASS(ITYPE(I)) * SUM(VXYZ**2.0)
      TOTMASS = TOTMASS + MASS(ITYPE(I))
   END DO

   DO A=1,NUMBEADS
      I = BEAD(A)
      EK(2) = EK(2) + MASS(ITYPE(I)) * SUM(VXYZ**2.0)
      TOTMASS = TOTMASS + MASS(ITYPE(I))
   END DO

   EK(1) = 0.5 * EK(1)
   EK(2) = 0.5 * EK(2)
   TEMP = SUM(EK) * MKTEMP
   TEMP_ATOM = EK(1) * MKTEMP_ATOM
   TEMP_BEAD = EK(2) * MKTEMP_BEAD

   ! CREATE LISTS OF BONDS, ANGLES AND TORSIONS
   CALL SETLIS()
   IF (ISTOP == 1) STOP 'Failed in SETLIS'

   !Read virtangles
   IF(IBRDESCR .eq. 0) THEN
      CALL RDVIRTANGLES()
      IF(ISTOP .eq. 1) STOP 'Failed in RDVIRTANGLES'
   END IF

   CONNECTIONS = 0
   CONNECTED_TO = 0

   CALL BUILD_CONNECTIVITY(NUMATOMS,ATOM,NONBEXC_ATOM)
   IF(ISTOP .eq. 1) STOP 'Failed in BUILD_CONNECTIVITY'

   IF(IBRDESCR .eq. 0) THEN
      CALL BUILD_CONNECTIVITY(NCOARSE,BEAD,NONBEXC_BEAD)
      IF(ISTOP .eq. 1) STOP 'Failed in BUILD_CONNECTIVITY beads'
```

```
      CALL RDVIRTBONDS()
      IF(ISTOP .EQ. 1) STOP 'Failed in RDVIRTBONDS'
   END IF

#ifdef REPORT_EXCLUSIONS
   CALL REPORT_EXCLUSIONS() ! Reports all nonbonded exclusions
#endif

   CALL MAPS (MAP_ATOM,MAPSIZE_ATOM, NCELLX_ATOM, NCELLY_ATOM, NCELLZ_ATOM)
   CALL LINKS (HEAD_ATOM,MAXNUMCELL_ATOM,ATOM,NUMATOMS,CELL_ATOM &
        , NCELLX_ATOM, NCELLY_ATOM, NCELLZ_ATOM, LCLIST_ATOM)

   IF(IBRDESCR .eq. 0) THEN
      CALL MAPS (MAP_BEAD,MAPSIZE_BEAD, NCELLX_BEAD, NCELLY_BEAD, NCELLZ_BEAD)
      CALL LINKS (HEAD_BEAD,MAXNUMCELL_BEAD,BEAD,NCOARSE,CELL_BEAD, &
           NCELLX_BEAD, NCELLY_BEAD, NCELLZ_BEAD, LCLIST_BEAD)
   END IF

   CALL UPDATE_NEIGHBOURLIST()

   !      WRITE THE TOPOLOGY FILE
   CALL WRITEPSF()

   WRITE(*,*)
   WRITE(*,*) 'Beginning simulation'
   WRITE(*,*)
   WRITE(*,*) 'Step:     Temperature:         Pressure:'

   IF(NMTS .eq. 0) THEN
      CALL NEW_LOOP()
   ELSE
      CALL MTS_LOOP()
   END IF

   CLOSE (201)
   CLOSE (202)

   Write(*,*)'************************************'
   Write(*,*)'********* END OF DYNAMICS ***********'
   Write(*,*)'  .   .   .   .        `   `   .   .   .        `   `   .  '
   Write(*,*)'  (|-(|(|-(|       (|-(|      '
   Write(*,*)'  (="','-',"" (="','-',"")      (="','-',"")    '
   Write(*,*)'  (,('')('')     (,('')('')    (,('')('')  '
   Write(*,*)'  .   .   .   .        `   `   .   .   .        `   `   .  '
   Write(*,*)'************************************'

END PROGRAM IBISCO
```

D.1 File IO Subroutines

These subroutines dealt with the reading of files into the program. These are mostly unchanged from the original, with the obvious exceptions of anything related to hybrid models.

src/RDCONTROL.f90

```
SUBROUTINE RDCONTROL ()
   USE MODULEPARSING
   USE VAR
   USE MTS
   IMPLICIT NONE

   INTEGER :: ALARM, IOS, IOS2
   CHARACTER(80) TEXT

   OPEN (2, IOSTAT=IOS, FILE='control', STATUS='OLD')

   IF (IOS.NE.0) THEN
      WRITE(1,*) '**** FATAL ERROR!'
      WRITE(1,*) 'File control does not exist ****'
      ISTOP=1
      RETURN
```

```
    END IF

    ! TAKE A INVALID VALUE TO ENSEMBLE AND INTERACT AND RESTART
    ENSEMBLE = 10
    RESTART  = 10
    ALARM = 10

    DO WHILE (.TRUE.)

       READ (2, '(A80)',IOSTAT=IOS2) LINE
       CALL PARSE ()
       IF (STRNGS(1) == 'ensemble') THEN
          READ (STRNGS(2),*) TEXT
          IF ((TEXT == 'NVE').OR.((TEXT == 'nve'))) ENSEMBLE = 0
          IF ((TEXT == 'NVT').OR.((TEXT == 'nvt'))) ENSEMBLE = 1
          IF ((TEXT == 'NPT').OR.((TEXT == 'npt'))) ENSEMBLE = 2

          IF ( ENSEMBLE == 10 ) THEN
             WRITE(1,*) '**** FATAL ERROR! ****'
             WRITE(1,*) 'You did not input a valid Ensemble in control'
             WRITE(1,*) '**** Possible options: NVT/NVE/NPT ****'
             ISTOP=1
             RETURN
          END IF
          EXIT
       END IF
       IF (IOS2 /= 0) EXIT
    END DO
    REWIND (2)

    DO WHILE (.TRUE.)
       READ (2, '(A80)',IOSTAT=IOS2) LINE
       CALL PARSE ()
       IF (STRNGS(1) == 'temperature') THEN
          READ (STRNGS(2),*) TEMP0
          TEMP = TEMP0 / TEMPSCALE
          TEMP_IN = TEMP
          ALARM = 0
          EXIT
       END IF

       IF (IOS2 /= 0) EXIT
    END DO
    IF (ALARM.NE.0) THEN
       CALL CONTROL_ERROR('temperature')
       ISTOP=1
    END IF
    REWIND (2)

    ALARM = 10
    DO WHILE (.TRUE.)
       READ (2, '(A80)',IOSTAT=IOS2) LINE
       CALL PARSE ()
       IF (STRNGS(1) == 'pressure') THEN
          READ (STRNGS(2),*) PRESSURE0
          PRESSURE = PRESSURE0/PSCALE
          ALARM = 0
          EXIT
       END IF
       IF (IOS2 /= 0) EXIT
    END DO
    IF (ALARM.NE.0) THEN
       CALL CONTROL_ERROR('pressure')
       ISTOP=1
    END IF
    ALARM = 10
    REWIND (2)
    DO WHILE (.TRUE.)

       READ (2, '(A80)',IOSTAT=IOS2) LINE
       CALL PARSE ()
       IF (STRNGS(1) == 'atoms') THEN
          READ (STRNGS(2),*) NATOMS
          ALARM = 0
          EXIT
       END IF

       IF (IOS2 /= 0) EXIT
    END DO
    IF (ALARM.NE.0) THEN
```

```
      CALL  CONTROL_ERROR('atoms')
      ISTOP=1
    END IF
100 ALARM = 10
    REWIND (2)

    DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
105   CALL PARSE ()
      IF (STRNGS(1) == 'num_of_time_steps') THEN
        READ (STRNGS(2),*) NSTEP
        ALARM = 0
        EXIT
110   END IF

      IF (IOS2 /= 0) EXIT
    END DO
    IF (ALARM.NE.0) THEN
115   CALL CONTROL_ERROR('num_of_time_steps')
      ISTOP=1
    END IF
    ALARM = 10
    REWIND (2)
120
    DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'time_step') THEN
125     READ (STRNGS(2),*) DT0
        DT = DT0*1.0D-12/TIMESCALE
        ALARM = 0
        EXIT
      END IF
130
      IF (IOS2 /= 0) EXIT
    END DO
    IF (ALARM.NE.0) THEN
      CALL CONTROL_ERROR('time_step')
135   ISTOP=1
    END IF

    IBRDESCR = 10
    REWIND (2)
140
    DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'virtual_sites') THEN
145     READ (STRNGS(2),*) NVIRTA
        NITEMS = NATOMS + NVIRTA
        IF(NVIRTA .gt. 0) THEN
          IBRDESCR = 0
        ELSE
150       IBRDESCR = 1
        END IF
        EXIT
      END IF
      IF (IOS2 /= 0) EXIT
155 END DO

    IF(IBRDESCR .eq. 10) THEN
      CALL CONTROL_ERROR('virtual_sites')
      ISTOP =1
160 END IF

    ALARM = 10
    REWIND (2)
    DO WHILE (.TRUE.)
165   READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'bead_cutoff') THEN
        READ (STRNGS(2),*) RCUT_BEAD
        RCUTSQ_BEAD = RCUT_BEAD * RCUT_BEAD
170     ALARM = 0
        EXIT
      END IF
      IF (IOS2 /= 0) EXIT
    END DO
175
    IF (ALARM .NE. 0 .AND. IBRDESCR .NE. 1) THEN
```

```
      CALL CONTROL_ERROR('bead_cutoff')
      ISTOP=1
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'bead_neighbour_list_cutoff') THEN
         READ (STRNGS(2),*) RLIST_BEAD
         ALARM = 0

         IF ( RLIST_BEAD < RCUT_BEAD ) THEN
            WRITE(1,*) '**** FATAL ERROR! ****'
            WRITE(1,*) 'Neighbour list cutoff must be larger than the cutoff.'
            ISTOP=1
            RETURN
         END IF
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM .NE. 0 .AND. IBRDESCR .NE. 1) THEN
      CALL CONTROL_ERROR('bead_neighbour_list_cutoff')
      ISTOP=1
   END IF

   IF(IBRDESCR .EQ. 0) THEN
      ALARM = 10
      REWIND (2)
      DO WHILE (.TRUE.)
         READ (2, '(A80)',IOSTAT=IOS2) LINE
         CALL PARSE ()
         IF (STRNGS(1) == 'non_bonded_bead') THEN
            READ (STRNGS(2),*) NONBEXC_BEAD
            ALARM = 0

            IF ((NONBEXC_BEAD .NE. 4).AND.(NONBEXC_BEAD .NE. 5)) THEN
               WRITE(1,*) '**** FATAL ERROR! FATAL ERROR! ****'
               WRITE(1,*) 'Invalid non-bonded interactions'
               WRITE(1,*) '**** Check control file ****'
               ISTOP = 1
               RETURN
            END IF
            EXIT
         END IF

         IF (IOS2 /= 0) EXIT
      END DO
      IF (ALARM.NE.0) THEN
         CALL CONTROL_ERROR('non_bonded_bead')
         ISTOP=1
      END IF
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'update_neighbour_list') THEN
         READ (STRNGS(2),*) NUPDATE
         ALARM = 0
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0) THEN
      CALL CONTROL_ERROR('update_neighbour_list')
      ISTOP=1
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
```

```
      IF (STRNGS(1) == 'temperature_coupling_time') THEN
         READ (STRNGS(2),*) TAUT0
         TAUT = TAUT0*1.0D-12/TIMESCALE
         ALARM = 0
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0.and.(ENSEMBLE.EQ.1.OR.ENSEMBLE.EQ.2)) THEN
      CALL CONTROL_ERROR('temperature_coupling_time')
      ISTOP=1
   END IF
   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'pressure_coupling_time') THEN
         READ (STRNGS(2),*) TAUP0
         TAUP = TAUP0 * 1.0D-12 / TIMESCALE
         ALARM = 0
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0.and.ENSEMBLE.EQ.2) THEN
      CALL CONTROL_ERROR('pressure_coupling_time')
      ISTOP=1
   END IF
   ALARM = 10
   REWIND (2)

   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'isothermal_compressibility') THEN
         READ (STRNGS(2),*) BETA0
         BETA = BETA0*PSCALE
         ALARM = 0
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0.AND.ENSEMBLE.EQ.2) THEN
      CALL CONTROL_ERROR('isothermal_compressibility')
      ISTOP=1
   END IF
   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'cutoff') THEN
         READ (STRNGS(2),*) RCUT_ATOM
         RCUTSQ_ATOM = RCUT_ATOM * RCUT_ATOM
         ALARM = 0
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0) THEN
      CALL CONTROL_ERROR('cutoff')
      ISTOP=1
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'neighbour_list_cutoff') THEN
         READ (STRNGS(2),*) RLIST_ATOM
         ALARM = 0
         IF ( RLIST_ATOM < RCUT_ATOM ) THEN
            WRITE(1,*) '**** FATAL ERROR! ****'
            WRITE(1,*) 'Neighbour list cutoff must be larger than the cutoff.'
```

```
            ISTOP=1
         END IF
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0) THEN
      CALL CONTROL_ERROR('neighbour_list_cutoff')
      ISTOP=1
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'trajectory') THEN
         READ (STRNGS(2),*) NTRJ
         ALARM = 0
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0) THEN
      CALL CONTROL_ERROR('trajectory')
      ISTOP=1
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'halt_drift') THEN
         READ (STRNGS(2),*) HALT_DRIFT
         ALARM = 0
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0) THEN
      CALL CONTROL_ERROR('halt_drift')
      ISTOP=1
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'non_bonded') THEN
         READ (STRNGS(2),*) NONBEXC_ATOM
         ALARM = 0

         IF ((NONBEXC_ATOM .NE. 4).AND.(NONBEXC_ATOM .NE. 5)) THEN
            WRITE(1,*) '**** FATAL ERROR! FATAL ERROR! ****'
            WRITE(1,*) 'Invalid non-bonded interactions '
            WRITE(1,*) '**** Check control file ****'
            ISTOP = 1
            RETURN
         END IF
         EXIT
      END IF

      IF (IOS2 /= 0) EXIT
   END DO
   IF (ALARM.NE.0) THEN
      CALL CONTROL_ERROR('non_bonded')
      ISTOP=1
   END IF

   ALARM = 10
   REWIND (2)
   DO WHILE (.TRUE.)
      READ (2, '(A80)',IOSTAT=IOS2) LINE
      CALL PARSE ()
      IF (STRNGS(1) == 'initialize_velocities') THEN
```

```
            READ (STRNGS(2),*) TEXT
            ALARM = 0
            IF ((TEXT(1:1) == 'Y').OR.((TEXT(1:1) == 'y'))) RESTART = 1
            IF ((TEXT(1:1) == 'N').OR.((TEXT(1:1) == 'n'))) RESTART = 0

            IF ( RESTART == 10 ) THEN
               CALL CONTROL_ERROR('initialize_velocities')
               ISTOP=1
               RETURN
            END IF
            EXIT
         END IF

         IF (IOS2 /= 0) EXIT
      END DO
      IF (ALARM.NE.0) THEN
         CALL CONTROL_ERROR('initialize_velocities')
         ISTOP=1
      END IF

      ALARM = 10
      REWIND(2)
      DO WHILE (.TRUE.)
         READ(2, '(A80)', IOSTAT=IOS2) LINE
         CALL PARSE()
         IF (STRNGS(1) == 'mts') THEN
            READ(STRNGS(2), *) NMTS
            ALARM = 0
         END IF
         IF (IOS2 .ne. 0) EXIT
      END DO
      IF (ALARM .ne. 0) THEN
         CALL CONTROL_ERROR('mts')
         ISTOP = 1
      END IF


      ALARM = 10
      REWIND (2)
      DO  WHILE (.TRUE.)
         READ (2, '(A80)',IOSTAT=IOS2) LINE
         CALL PARSE ()
         IF (STRNGS(1) == 'END'.or.STRNGS(1).EQ.'end' &
               .or.STRNGS(1).EQ.'End') THEN
            ALARM = 0
            EXIT
         END IF
         IF (IOS2 /= 0) EXIT
      END DO

      IF (ALARM.NE.0) THEN
         CALL CONTROL_ERROR('end')
         ISTOP=1
         RETURN
      END IF

      CLOSE (2)

      RETURN

   END SUBROUTINE RDCONTROL

   SUBROUTINE CONTROL_ERROR(variable)
      IMPLICIT NONE

      CHARACTER(*) :: variable

      WRITE(1,*) '** Fatal error: Keyword ', variable, ' missing from control file **'

      RETURN
   END SUBROUTINE CONTROL_ERROR
```

src/RDINTERACT.f90

```
SUBROUTINE RDINTERACT()
   USE MODULEPARSING
   USE VAR
   IMPLICIT NONE
```

```
    INTEGER :: I, J, L, LL, IB, JB, IA, JA, KA, IT, JT, KT, LT, K, ITA
    INTEGER :: INB, JNB, TYPEI
    INTEGER :: ios = 0, iosIN = 0, ios2=0
    REAL*8 :: R
    CHARACTER(len=80) :: LINE2
    logical :: alloc=.true.
    !Detailed energy breakdown variables
    INTEGER :: HASH, TI, TJ, TK, TL, TI_TYPE, TJ_TYPE, TK_TYPE, TL_TYPE, WHAT_TYPE
    INTEGER, DIMENSION(:), ALLOCATABLE :: BOND_I, BOND_J, ANGLE_I, ANGLE_J, ANGLE_K
    INTEGER, DIMENSION(:), ALLOCATABLE :: TORSION_I, TORSION_J, TORSION_K, TORSION_L
    INTEGER, DIMENSION(:), ALLOCATABLE :: OOP_I, OOP_J, OOP_K, OOP_L

    OPEN (4, IOSTAT=IOS, FILE='interaction', STATUS='OLD')

    IF (IOS.NE.0) THEN
       WRITE(1,*)' **** FATAL ERROR! File interaction does not exist ****'
       ISTOP=1
       RETURN
    END IF

    READ (4, '(A80)') LINE
    CALL PARSE ()
    IF (STRNGS(1) == 'atom_types') THEN
       READ (STRNGS(2),*) NTYPE
    END IF

    ALLOCATE(LABEL(NTYPE))
    ALLOCATE(MASS0(NTYPE))
    ALLOCATE(MASS(NTYPE))
    ALLOCATE(INVMASS(NTYPE))
    ALLOCATE(NAME_LABEL(NTYPE))

    READ(4,*)
    DO I = 1, NTYPE
       READ (4, '(A80)') LINE
       CALL PARSE ()
       IF (STRNGS(1) .EQ. 'bond_types') THEN
          ISTOP = 1
          WRITE(*,*) '****************** FATAL ERROR **********************'
          WRITE(*,*) '* Number of atom_types different from Number of Atom *'
          WRITE(*,*) '*                 Check Interaction File              *'
          WRITE(*,*) '*******************************************************'
          RETURN
       END IF
       READ (STRNGS(1),*) TYPEI
       READ (STRNGS(2),*) LABEL(TYPEI)
       READ (STRNGS(3),*) MASS0(TYPEI)
       READ (STRNGS(4),*,iostat=ios) name_label(TYPEI)
       if(ios .ne. 0)then
          call error_inter ()
          ISTOP = 1
          return
       end if
       if(name_label(TYPEI) .ne. 'A' .and. name_label(TYPEI) .ne. 'a')then
          if(name_label(TYPEI) .ne. 'B' .and. name_label(TYPEI) .ne. 'b')then
             call error_inter ()
             ISTOP = 1
             return
          end if
       end if
       MASS(I) = MASS0(I)/NA/MASSSCALE!/1000.0
       INVMASS(I) = 1.0D0 / MASS(I)
    END DO

    ALLOCATE(IBONDT(NTYPE, NTYPE ))
    ALLOCATE(INBONDT(NTYPE, NTYPE ))

    IBONDT = 0
    INBONDT = 0

    READ (4, '(A80)') LINE
    CALL PARSE ()
    IF (STRNGS(1) == 'bond_types') THEN
       READ (STRNGS(2),*) NBTYPE
    ELSE
       ISTOP = 1
       WRITE(*,*) '****************** FATAL ERROR **********************'
       WRITE(*,*) '* Number of atom_types different from Number of Atom *'
       WRITE(*,*) '*                 Check Interaction File              *'
       WRITE(*,*) '*******************************************************'
```

```
      RETURN
    END IF
88  READ(4,*)
    IF (NBTYPE.GT.0) THEN
      ALLOCATE(RBOND(NBTYPE,0:MAXINPUT))
      ALLOCATE(BOND_FORCE(NBTYPE,0:MAXINPUT))
      ALLOCATE(BOND_POT(NBTYPE,0:MAXINPUT))
93    ALLOCATE(BINB(NBTYPE))
      ALLOCATE(NDATB(NBTYPE))
      ALLOCATE(BOND_I(NBTYPE), BOND_J(NBTYPE))

      BOND_FORCE = 0.0D0
98    BOND_POT = 0.0D0

      DO I = 1, NBTYPE
        READ (4, '(A80)') LINE
        CALL PARSE ()
103     IF (STRNGS(1) .EQ. 'angle_types') THEN
          ISTOP = 1
          WRITE(1,*) '**** FATAL ERROR ****'
          WRITE(1,*) 'Number of bond_types different from'
          WRITE(1,*) 'Number bond interaction '
108       WRITE(1,*) 'Check Interaction File'
          RETURN
        END IF
        READ (STRNGS(1),*) IB
        READ (STRNGS(2),*) JB
113
        BOND_I(I) = IB
        BOND_J(I) = JB

        IF(IBONDT(IB,JB).EQ.0) THEN
118       IBONDT(IB,JB) = I
          IBONDT(JB,IB) = I
        ELSE
          WRITE(1,*) '**** FATAL ERROR: ****'
          WRITE(1,*) 'Duplicate entry in interaction file for bonds'
123       ISTOP=1
          RETURN
        ENDIF

        OPEN (11, IOSTAT=IOS, FILE=STRNGS(3), STATUS='OLD')
128     IF (IOS.NE.0) THEN
          WRITE(1,*)' **** FATAL ERROR! File ', STRNGS(3),' does not exist ****'
          ISTOP=1
          RETURN
        END IF
133
        K = 0
        DO WHILE (.TRUE.)
          READ(11,*,IOSTAT=IOS2) RBOND(I,K), BOND_POT(I,K)
          IF(IOS2 .ne. 0) EXIT
138       K = K+1
        END DO
        NDATB(I) = K - 1

        CLOSE(11)
143
        !Rescale energies
        DO J =0,NDATB(I)
          BOND_POT(I,J) = BOND_POT(I,J) *1000.0 /NA /ESCALE
        END DO
148
        BINB(I) = RBOND(I,1)-RBOND(I,0)

      END DO
    END IF !If NBTYPE gt 0
153
    READ (4, '(A80)') LINE
    CALL PARSE ()
    IF (STRNGS(1) == 'angle_types') THEN
      READ (STRNGS(2),*) NATYPE
158 ELSE
      ISTOP = 1
      WRITE(1,*) '**** FATAL ERROR ****'
      WRITE(1,*) 'Number of bond_types different from'
      WRITE(1,*) 'Number bond interaction '
163   WRITE(1,*) 'Check Interaction File'
      RETURN
    END IF
```

```
      READ(4,*)
      IF (NATYPE.GT.0) THEN
168       ALLOCATE(ANGLE(NATYPE, 0:MAXINPUT))
          ALLOCATE(BEND_FORCE(NATYPE, 0:MAXINPUT))
          ALLOCATE(BEND_POT(NATYPE, 0:MAXINPUT))
          ALLOCATE(BINA(NATYPE))
          ALLOCATE(NDATAN(NATYPE))
173       ALLOCATE(ANGLE_I(NATYPE), ANGLE_J(NATYPE), ANGLE_K(NATYPE))

          ALLOCATE(JANGLEIJK(NITEMS,10))
          ALLOCATE(KANGLEIJK(NITEMS,10))
          ALLOCATE(NOJANGLEIJK(NITEMS,10))
178       ALLOCATE(NOKANGLEIJK(NITEMS,10))
          ALLOCATE(IANGT(NTYPE,NTYPE,NTYPE))
          ALLOCATE(ATIANG(NATYPE*3))

          IANGT = 0
183
          BEND_FORCE = 0.0D0
          BEND_POT = 0.0D0

          DO I = 1, NATYPE
188           ITA = 3*(I-1)
              READ (4, '(A80)') LINE
              CALL PARSE ()
              IF (STRNGS(1) .EQ. 'torsion_types') THEN
                  ISTOP = 1
193               WRITE(1,*) '**** FATAL ERROR ****'
                  WRITE(1,*) 'Number of angle_types different from'
                  WRITE(1,*) 'Number angle interaction '
                  WRITE(1,*) 'Check Interaction File'
                  RETURN
198           END IF
              READ (STRNGS(1),*) IA
              READ (STRNGS(2),*) JA
              READ (STRNGS(3),*) KA

203           ANGLE_I(I) = IA
              ANGLE_J(I) = JA
              ANGLE_K(I) = KA

              IF(IANGT(IA,JA,KA).EQ.0) THEN
208               IANGT(IA,JA,KA) = I
                  IANGT(KA,JA,IA) = I
                  ATIANG(ITA+1) = IA
                  ATIANG(ITA+2) = JA
                  ATIANG(ITA+3) = KA
213           ELSE
                  WRITE(1,*) '**** FATAL ERROR: ****'
                  WRITE(1,*) 'Duplicate entry in interaction file for angles'
                  ISTOP=1
                  RETURN
218           ENDIF

              OPEN (11, IOSTAT=IOS, FILE=STRNGS(4), STATUS='OLD')
              IF (IOS.NE.0) THEN
                  WRITE(1,*)' **** FATAL ERROR! File ', STRNGS(4),' does not exist ****'
223               WRITE(*,*)' **** FATAL ERROR! File ', STRNGS(4),' does not exist ****'
                  ISTOP=1
                  RETURN
              END IF

228           READ(11,*,IOSTAT=IOS2)ANGLE(I,0), BEND_POT(I,0)
              READ(11,*,IOSTAT=IOS2)ANGLE(I,1), BEND_POT(I,1)
              BEND_POT(I,0) = BEND_POT(I,0)*1000.0/ NA / ESCALE
              BEND_POT(I,1) = BEND_POT(I,1)*1000.0/ NA / ESCALE

233           BINA(I) = ANGLE(I,1)-ANGLE(I,0)

              IF(ANGLE(I, 0) /= 0.0) THEN
                  ANGLE(I,0) = 0.0
                  ANGLE(I,2) = ANGLE(I,1)
238               BEND_POT(I,2) = BEND_POT(I,1)
                  ANGLE(I,1) = BINA(I)
                  BEND_POT(I,1) = BEND_POT(I,0)
                  K = 3
              ELSE
243               K = 2
              END IF
```

```
        DO WHILE (.TRUE.)
          READ(11,*,IOSTAT=IOS2)ANGLE(I, K), BEND_POT(I,K)
248       BEND_POT(I,K) = BEND_POT(I,K)*1000.0/ NA / ESCALE
          IF (IOS2 /= 0) EXIT
          K = K + 1
        END DO
        CLOSE (11)
253     NDATAN(I) = K - 1
      END DO
    END IF

    READ (4, '(A80)') LINE
258 CALL PARSE ()
    IF (STRNGS(1) == 'torsion_types') THEN
      READ (STRNGS(2),*) NTTYPE
    ELSE
      ISTOP = 1
263   WRITE(1,*) '**** FATAL ERROR ****'
      WRITE(1,*) 'Number of torsion_types from'
      WRITE(1,*) 'Number torsion interaction '
      WRITE(1,*) 'Check Interaction File'
      RETURN
268 END IF
    READ(4,*)

    IF (NTTYPE.GT.0) THEN
      ALLOCATE(ANGLE_TOR(NTTYPE, 0:MAXINPUT))
273   ALLOCATE(TOR_FORCE(NTTYPE, 0:MAXINPUT))
      ALLOCATE(TOR_POT(NTTYPE, 0:MAXINPUT))
      ALLOCATE(BINT(NTTYPE))
      ALLOCATE(NDATT(NTTYPE))

278   ALLOCATE(JTORIJKL(NATOMS,10))
      ALLOCATE(KTORIJKL(NATOMS,10))
      ALLOCATE(LTORIJKL(NATOMS,10))

      ALLOCATE(FJTORIJKL(NATOMS,10))
283   ALLOCATE(FKTORIJKL(NATOMS,10))
      ALLOCATE(FLTORIJKL(NATOMS,10))

      ALLOCATE(ITORT(NTYPE,NTYPE,NTYPE,NTYPE))
      ALLOCATE(TORSION_I(NTTYPE), TORSION_J(NTTYPE), &
288         TORSION_K(NTTYPE), TORSION_L(NTTYPE))

      ITORT = 0
      TOR_FORCE = 0.0D0
      TOR_POT = 0.0D0
293
      DO I = 1, NTTYPE
        READ (4, '(A80)') LINE
        CALL PARSE ()
        IF (STRNGS(1) .EQ. 'non_bonded_interaction_types') THEN
298       ISTOP = 1
          WRITE(1,*) '**** FATAL ERROR ****'
          WRITE(1,*) 'Number of torsion_types from'
          WRITE(1,*) 'Number torsion interaction '
          WRITE(1,*) 'Check Interaction File'
303       RETURN
        END IF
        READ (STRNGS(1),*) IT
        READ (STRNGS(2),*) JT
        READ (STRNGS(3),*) KT
308     READ (STRNGS(4),*) LT

        TORSION_I(I) = IT
        TORSION_J(I) = JT
        TORSION_K(I) = KT
313     TORSION_L(I) = LT

        IF(ITORT(IT,JT,KT,LT).EQ.0) THEN
          ITORT(IT,JT,KT,LT) = I
          ITORT(LT,KT,JT,IT) = I
318     ELSE
          WRITE(1,*) '**** FATAL ERROR: ****'
          WRITE(1,*) 'Duplicate entry in interaction file for torsions  ****'
          WRITE(1,*)IT,JT,KT,LT
          ISTOP=1
323       RETURN
        ENDIF
```

```
          OPEN (11, IOSTAT=IOS, FILE=STRNGS(5), STATUS='OLD')
          IF (IOS.NE.0) THEN
328           WRITE(1,*) ' **** FATAL ERROR! File ', STRNGS(5),' does not exist ****'
              ISTOP=1
              RETURN
          END IF

333       READ(11,*,IOSTAT=IOS2)ANGLE_TOR(I,0), TOR_POT(I,0)
          READ(11,*,IOSTAT=IOS2)ANGLE_TOR(I,1), TOR_POT(I,1)
          TOR_POT(I,0) = TOR_POT(I,0)*1000.0/ NA / ESCALE
          TOR_POT(I,1) = TOR_POT(I,1)*1000.0/ NA / ESCALE

338       BINT(I) = ANGLE_TOR(I,1)-ANGLE_TOR(I,0)

          IF(ANGLE_TOR(I, 0) /= 0.0) THEN
              ANGLE_TOR(I,0) = 0.0
              ANGLE_TOR(I,2) = ANGLE_TOR(I,1)
343           TOR_POT(I,2) = TOR_POT(I,1)
              ANGLE_TOR(I,1) = BINT(I)
              TOR_POT(I,1) = TOR_POT(I,0)
              K = 3
          ELSE
348           K = 2
          END IF

          DO WHILE (.TRUE.)
              READ(11,*,IOSTAT=IOS2)ANGLE_TOR(I, K), TOR_POT(I,K)
353           TOR_POT(I,K) = TOR_POT(I,K)*1000.0/ NA / ESCALE
              IF (IOS2 /= 0) EXIT
              K = K + 1
          END DO
          NDATT(I) = K - 1
358       CLOSE (11)
      END DO
  END IF

  READ (4, '(A80)') LINE
363 CALL PARSE ()
  IF (STRNGS(1) == 'non_bonded_interaction_types') THEN
      READ (STRNGS(2),*) NNBTYPE
  ELSE
      ISTOP = 1
368   WRITE(1,*) '**** FATAL ERROR ****'
      WRITE(1,*) 'Number of torsion different from'
      WRITE(1,*) 'Number torsion interaction '
      WRITE(1,*) 'Check Interaction File'
      RETURN
373 END IF
  READ(4,*)
  IF (NNBTYPE.GT.0) THEN
      ALLOCATE(RNBOND(0:MAXINPUT, NNBTYPE))
      ALLOCATE(NBOND_FORCE(0:MAXINPUT,NNBTYPE))
378   ALLOCATE(NBOND_POT(0:MAXINPUT,NNBTYPE))
      ALLOCATE(BINNB(NNBTYPE))
      ALLOCATE(NDATNB(NNBTYPE))
      NBOND_FORCE = 0.0
      NBOND_POT = 0.0
383
      DO I = 1, NNBTYPE

          READ (4, '(A80)') LINE
          CALL PARSE ()
388       IF (STRNGS(1) .EQ. 'Out_of_Planes') THEN
              ISTOP = 1
              WRITE(1,*) '**** FATAL ERROR ****'
              WRITE(1,*) 'Number of non_bonded_interaction_types different from'
              WRITE(1,*) 'Number nonbonded interaction '
393           WRITE(1,*) 'Check Interaction File'
              RETURN
          END IF
          READ (STRNGS(1),*) INB
          READ (STRNGS(2),*) JNB
398
          IF(INBONDT(INB,JNB).EQ.0) THEN
              INBONDT(INB,JNB) = I
              INBONDT(JNB,INB) = I
          ELSE
403           WRITE(1,*) '**** FATAL ERROR: ****'
              WRITE(1,*) 'Duplicate entry in interaction file for non-bonded interactions'
              WRITE(1,*)INB,JNB
```

```
            ISTOP=1
            RETURN
408         ENDIF

            OPEN (11, IOSTAT=IOS, FILE=STRNGS(3), STATUS='OLD')
            IF (IOS.NE.0) THEN
               WRITE(1,*)' **** FATAL ERROR! File ', STRNGS(3),' does not exist ****'
413            ISTOP=1
               RETURN
            END IF

            READ(11,*,IOSTAT=IOS2)RNBOND(0, I), NBOND_POT(0,I)
418         READ(11,*,IOSTAT=IOS2)RNBOND(1, I), NBOND_POT(1,I)
            NBOND_POT(0,I) = NBOND_POT(0,I) * 1000.0 / NA / ESCALE
            NBOND_POT(1,I) = NBOND_POT(1,I) * 1000.0 / NA / ESCALE

            BINNB(I) = RNBOND(1,I) - RNBOND(0,I)
423
            IF(RNBOND(0,I) /= 0.0) THEN
               RNBOND(0,I) = 0.0
               WRITE(*,*) ' Nonbonded file ', I, ' doesnt begin with 0'
               RNBOND(2,I) = RNBOND(1,I)
428            NBOND_POT(2,I) = NBOND_POT(1,I)
               RNBOND(1,I) = BINNB(I)
               NBOND_POT(1,I) = NBOND_POT(0,I)
               K = 3
            ELSE
433            K = 2
            END IF

            DO WHILE (.TRUE.)
               READ(11,*,IOSTAT=IOS2)RNBOND(K,I), NBOND_POT(K,I)
438            IF (IOS2 .EQ. 0) THEN
                  NBOND_POT(K,I) = NBOND_POT(K,I) * 1000.0 / NA / ESCALE
                  K = K + 1
               ELSE
                  EXIT
443            END IF
            END DO

            NDATNB(I) = K - 1
            CLOSE (11)
448
         END DO
      END IF

      READ (4, '(A80)') LINE
453   CALL PARSE ()
      IF (STRNGS(1) == 'Out_of_Planes') THEN
         READ (STRNGS(2),*) NOTYPE
      ELSE
         ISTOP = 1
458      WRITE(1,*) '**** FATAL ERROR ****'
         WRITE(1,*) 'Number of nonbonded_interactions different from'
         WRITE(1,*) 'Number of nonbonded interactions'
         WRITE(1,*) 'Check Interaction File'
         RETURN
463   END IF
      READ(4,*)

      IF (NOTYPE.GT.0) THEN
         ALLOCATE(ANGLE_OOP(NOTYPE, 0:MAXINPUT))
468      ALLOCATE(OOP_FORCE(NOTYPE, 0:MAXINPUT))
         ALLOCATE(OOP_POT(NOTYPE, 0:MAXINPUT))
         ALLOCATE(BINO(NOTYPE))
         ALLOCATE(NDATO(NOTYPE))

473      ALLOCATE(JOOPIJKL(NATOMS,20))
         ALLOCATE(KOOPIJKL(NATOMS,20))
         ALLOCATE(LOOPIJKL(NATOMS,20))

         ALLOCATE(IOOPT(NTYPE,NTYPE,NTYPE,NTYPE))
478      ALLOCATE(OOP_I(NOTYPE), OOP_J(NOTYPE), OOP_K(NOTYPE), OOP_L(NOTYPE))

         IOOPT = 0
         OOP_FORCE = 0.0D0
         OOP_POT = 0.0D0
483
         DO I = 1, NOTYPE
            READ (4, '(A80)',IOSTAT = iosIN) LINE
```

```
      IF (iosIN .NE. 0) THEN
         ISTOP = 1
488      WRITE(1,*) '**** FATAL ERROR ****'
         WRITE(1,*) 'Number of Out_of_Planes different from'
         WRITE(1,*) 'Number Out of Planes interaction '
         WRITE(1,*) 'Check Interaction File'
         RETURN
493   END IF
      CALL PARSE ()
      READ (STRNGS(1),*) IT
      READ (STRNGS(2),*) JT
      READ (STRNGS(3),*) KT
498   READ (STRNGS(4),*) LT

      OOP_I(I) = IT
      OOP_J(I) = JT
      OOP_K(I) = KT
503   OOP_L(I) = LT

      IF (I .EQ. NOTYPE) THEN
         READ (4, '(A80)',IOSTAT = iosIN) LINE2
         IF(iosIN .EQ. 0) THEN
508         ISTOP = 1
            WRITE(1,*) '**** FATAL ERROR ****'
            WRITE(1,*) 'Number of Out_of_Planes different from'
            WRITE(1,*) 'Number Out of Planes interaction '
            WRITE(1,*) 'Check Interaction File'
513         RETURN
         END IF
      END IF

      IF(IOOPT(IT,JT,KT,LT).EQ.0) THEN
518      IOOPT(IT,JT,KT,LT) = I
      ELSE
         WRITE(1,*) '**** FATAL ERROR: ****'
         WRITE(1,*) 'Duplicate entry in interaction file for out of planes'
         WRITE(1,*)IT,JT,KT,LT
523      ISTOP=1
         RETURN
      ENDIF

      OPEN (11, IOSTAT=IOS, FILE=STRNGS(5), STATUS='OLD')
528   IF (IOS.NE.0) THEN
         WRITE(1,*)' **** FATAL ERROR! File', STRNGS(5),' does not exist ****'
         ISTOP=1
         RETURN
      END IF
533
      READ(11,*,IOSTAT=IOS2)ANGLE_OOP(I,0), OOP_POT(I,0)
      READ(11,*,IOSTAT=IOS2)ANGLE_OOP(I,1), OOP_POT(I,1)
      OOP_POT(I,0) = OOP_POT(I,0)*1000.0/ NA / ESCALE
      OOP_POT(I,1) = OOP_POT(I,1)*1000.0/ NA / ESCALE
538
      if(ANGLE_OOP(I,1) .lt. ANGLE_OOP(I,0))then
         BINO(I) = ANGLE_OOP(I,0)-ANGLE_OOP(I,1)
      else
         BINO(I) = ANGLE_OOP(I,1)-ANGLE_OOP(I,0)
543   end if

      IF(ANGLE_OOP(I, 0) /= 0.0) THEN
         ANGLE_OOP(I,0) = 0.0
         ANGLE_OOP(I,2) = ANGLE_OOP(I,1)
548      OOP_POT(I,2) = OOP_POT(I,1)
         ANGLE_OOP(I,1) = BINO(I)
         OOP_POT(I,1) = OOP_POT(I,0)
         K = 3
      ELSE
553      K = 2
      END IF

      DO WHILE (.TRUE.)
         READ(11,*,IOSTAT=IOS2)ANGLE_OOP(I, K), OOP_POT(I,K)
558      OOP_POT(I,K) = OOP_POT(I,K) *1000.0/ NA / ESCALE
         IF (IOS2 /= 0) EXIT
         K = K + 1
      END DO
      NDATO(I) = K - 1
563   CLOSE (11)

   END DO
```

```
   END IF

   CLOSE (4)

!Determine the type (all atomistic, all CG, hybrid) of
!each bonded type, so it can be easily sorted later
!BOND_TYPE_LABEL(TIJ) returns:
! 1 - all atomistic
! 2 - all CG
! 3 - mixed
   ALLOCATE(BOND_TYPE_LABEL(NBTYPE), ANGLE_TYPE_LABEL(NATYPE), &
         TORSION_TYPE_LABEL(NTTYPE), OOP_TYPE_LABEL(NOTYPE))

   DO I=1,NBTYPE
      TI = BOND_I(I)
      TJ = BOND_J(I)
      TI_TYPE = WHAT_TYPE(NAME_LABEL(TI))
      TJ_TYPE = WHAT_TYPE(NAME_LABEL(TJ))
      HASH = TI_TYPE + TJ_TYPE
      IF(HASH .EQ. 2) THEN
         BOND_TYPE_LABEL(I) = 1
      ELSE IF (HASH .EQ. 4) THEN
         BOND_TYPE_LABEL(I) = 2
      ELSE
         BOND_TYPE_LABEL(I) = 3
      END IF
   END DO

   DO I=1,NATYPE
      TI = ANGLE_I(I)
      TJ = ANGLE_J(I)
      TK = ANGLE_K(I)
      TI_TYPE = WHAT_TYPE(NAME_LABEL(TI))
      TJ_TYPE = WHAT_TYPE(NAME_LABEL(TJ))
      TK_TYPE = WHAT_TYPE(NAME_LABEL(TK))
      HASH = TI_TYPE + TJ_TYPE + TK_TYPE !sum the types as a makeshift hash
      IF (HASH .EQ. 3) THEN !minimum hash = all atoms
         ANGLE_TYPE_LABEL(I) = 1
      ELSE IF (HASH .EQ. 6) THEN !max mash = all beads
         ANGLE_TYPE_LABEL(I) = 2
      ELSE !else mixture of the two
         ANGLE_TYPE_LABEL(I) = 3
      END IF
   END DO

   DO I=1,NTTYPE
      TI = TORSION_I(I)
      TJ = TORSION_J(I)
      TK = TORSION_K(I)
      TL = TORSION_L(I)
      TI_TYPE = WHAT_TYPE(NAME_LABEL(TI))
      TJ_TYPE = WHAT_TYPE(NAME_LABEL(TJ))
      TK_TYPE = WHAT_TYPE(NAME_LABEL(TK))
      TL_TYPE = WHAT_TYPE(NAME_LABEL(TL))
      HASH = TI_TYPE + TJ_TYPE + TK_TYPE + TL_TYPE
      IF (HASH .EQ. 4) THEN
         TORSION_TYPE_LABEL(I) = 1
      ELSE IF(HASH .EQ. 8) THEN
         TORSION_TYPE_LABEL(I) = 2
      ELSE
         TORSION_TYPE_LABEL(I) = 3
      END IF
   END DO

   IF( NOTYPE .GT. 0) THEN
      DO I=1,NOTYPE
         TI = OOP_I(I)
         TJ = OOP_J(I)
         TK = OOP_K(I)
         TL = OOP_L(I)
         TI_TYPE = WHAT_TYPE(NAME_LABEL(TI))
         TJ_TYPE = WHAT_TYPE(NAME_LABEL(TJ))
         TK_TYPE = WHAT_TYPE(NAME_LABEL(TK))
         TL_TYPE = WHAT_TYPE(NAME_LABEL(TL))
         HASH = TI_TYPE + TJ_TYPE + TK_TYPE + TL_TYPE
         IF (HASH .EQ. 4) THEN
            OOP_TYPE_LABEL(I) = 1
         ELSE IF(HASH .EQ. 8) THEN
            OOP_TYPE_LABEL(I) = 2
```

```
          ELSE
             OOP_TYPE_LABEL(I) = 3
          END IF
       END DO
    END IF

    RETURN
END SUBROUTINE RDINTERACT

INTEGER FUNCTION WHAT_TYPE(LABEL)
    !Parses the label into 1 for A/a or 2 for B/b
    CHARACTER :: LABEL

    WHAT_TYPE = 100

    IF(LABEL .EQ. 'A') THEN
       WHAT_TYPE = 1
    ELSE IF(LABEL .EQ. 'a') THEN
       WHAT_TYPE = 1
    ELSE IF(LABEL .EQ. 'B') THEN
       WHAT_TYPE = 2
    ELSE IF(LABEL .EQ. 'b') THEN
       WHAT_TYPE = 2
    END IF

    RETURN
END FUNCTION WHAT_TYPE

subroutine error_inter

    WRITE(*,*) '*************** FATAL ERROR ********************'
    WRITE(*,*) '*  the label for hybrid model must be equal to: *'
    WRITE(*,*) '*    A or a:  atoms                             *'
    WRITE(*,*) '*    B or b:  beads                             *'
    WRITE(*,*) '************************************************'

end subroutine error_inter
```

src/RDCOOR.f90

```
SUBROUTINE RDCOOR()
    USE MODULEPARSING
    USE VAR
    IMPLICIT NONE

    INTEGER :: IOS
    INTEGER :: I, J, L=0, LL, K, TT
    character(len=20) :: text

    OPEN (3, IOSTAT=IOS, FILE='coordinate', STATUS='OLD')

    IF (IOS.NE.0) THEN
       WRITE (1,*) ' **** FATAL ERROR! File coordinate does not exist ****'
       ISTOP=1
       RETURN
    END IF

    READ (3, '(A80)', iostat=ios) LINE
    IF (ios .NE. 0) CALL RDCOOR_ERROR()
    CALL PARSE ()
    READ (STRNGS(1),*) TITLE

    READ (3, '(A80)', iostat=ios) LINE
    IF (ios .NE. 0) CALL RDCOOR_ERROR()
    CALL PARSE ()
    IF (STRNGS(1) == 'Time') THEN
       READ (STRNGS(2),*) INITIME
    END IF
    READ(3,*)
    READ(3,*, iostat=ios)BOX(1), BOX(2), BOX(3)
    IF (ios .NE. 0) CALL RDCOOR_ERROR()

    BOXINV(:) = 1.0 / BOX(:)
    BOX2(:) = BOX(:) / 2.0

    READ (3,*)
    READ (3,*)
    READ (3,*)
    READ (3,*)
```

```
   READ (3,*)
   READ (3,*, iostat=ios)text, NMOL
   IF (ios .NE. 0) CALL RDCOOR_ERROR()

   ALLOCATE(NATM(NMOL))
   ALLOCATE(name_mol(NMOL))
   ALLOCATE(TYPE_LABEL(NITEMS))
   TYPE_LABEL = 1 !All things are labelled as atoms by default

   DO I = 1, NMOL !101

      READ(3,*,iostat=ios)NATM(I),text,tt,name_mol(i)
      if(ios .ne. 0)then
         call error_noname()
         istop=1
         return
      end if

      DO J = 1, NATM(I)  !102
         L = L + 1
         IF ( NATOMS .lt. L ) THEN
            WRITE(1,*) ' **** FATAL ERROR! ****'
            WRITE(1,*) 'No. Atoms in coordinate does not equal to NATOMS ****'
            ISTOP=1
            RETURN
         END IF
         READ (3,*,iostat=ios) LL,ITYPE(L),NBONDS(L),SXYZ(1,L),SXYZ(2,L),SXYZ(3,L)
         READ (3,*,iostat=ios) VXYZ(1,L),VXYZ(2,L),VXYZ(3,L),(JBOND(L,K),K=1,NBONDS(L))

         IF(ios .NE. 0) CALL RDCOOR_ERROR()

         MOL(L) = I !Which molecule atom L belongs to

         IF(IBRDESCR .eq. 0) THEN
            IF(name_label(itype(l)) .EQ. 'A' .OR. name_label(itype(l)) .EQ. 'a') THEN
               TYPE_LABEL(L) = 1
            ELSE IF (name_label(itype(l)) .EQ. 'B' .OR. name_label(itype(l)) .EQ. 'b') THEN
               TYPE_LABEL(L) = 2
            ELSE
               WRITE(1,*) '**** FATAL ERROR, the label must be equal to A or B ****'
               WRITE(1,*) '**** in atom number: ',L,' *****'
               ISTOP = 1
               RETURN
            END IF
         END IF

         VXYZ(:,L) = VXYZ(:,L) * 1.e3 / VSCALE

      END DO !102                  CONTINUE
   END DO !101      CONTINUE

   IF ( NATOMS /= L ) THEN
      WRITE(1,*) ' **** FATAL ERROR! ****'
      WRITE(1,*) 'No. Atoms in coordinate does not equal to NATOMS ****'
      ISTOP=1
      RETURN
   END IF
   CLOSE (3)

   RETURN
END SUBROUTINE RDCOOR

SUBROUTINE RDCOOR_ERROR()

   IMPLICIT NONE

   WRITE(*,*) '*********************************************'
   WRITE(*,*) '*                                           *'
   WRITE(*,*) '* Fatal error in RDCOOR, check coordinate file *'
   WRITE(*,*) '*                                           *'
   WRITE(*,*) '*********************************************'

   STOP

END SUBROUTINE RDCOOR_ERROR

subroutine error_noname
   WRITE (*,*) '***************** FATAL ERROR ******************'
   WRITE (*,*) '*   Some molecules name missing. Check your    *'
   WRITE (*,*) '*               coordinate file                *'
```

```
  WRITE (*,*) '***********************************************'
end subroutine
```

src/RDVIRTUAL.f90

```
SUBROUTINE RDVIRTUAL()
  USE VAR
  IMPLICIT NONE

  INTEGER :: iost, I, J, K, NUMATOM, TI, TJ, A
  INTEGER, PARAMETER :: MAX_ATOMS = 15 !Temp max number of atoms within a VS
  INTEGER :: ACTUAL_MAX
  REAL*4, PARAMETER :: MASSTOL = 0.0001
  REAL*8 :: SUMTOTBMASS
  !Temp array for filling in index of atoms before max VIRT_NUMATOMS is known
  INTEGER, POINTER :: INDX_ATM(:,:)

  OPEN(UNIT=10,FILE='virtual',status='old',IOSTAT=iost)

  IF(iost .ne. 0) THEN
     WRITE(*,*) '*** FATAL ERROR! File virtual site does not exist ****'
     WRITE(1,*) '*** FATAL ERROR! File virtual site does not exist ****'
     ISTOP=1
     RETURN
  END IF

  READ(10,*)
  READ(10,*)
  READ(10,*) NVIRTA
  READ(10,*)
  READ(10,*)

  !NVIRTA Number of virtual sites
  !Number of atoms in different virtual site types
  ALLOCATE(VIRT_NUMATOMS(NTYPE))
  !Mass of VS
  ALLOCATE(VIRT_MASS(NTYPE), VIRT_INVMASS(NTYPE))
  !Masscoefficient for atom in VS. Usage VIRT_MASSCOEFF(virt type, atom type)
  ALLOCATE(VIRT_MASSCOEFF(NTYPE,NTYPE))
  !Type of each virtual site.  Types are the same as in interaction file
  ALLOCATE(VITYPE(NVIRTA))
  !Center is 0 for COM VS or index of atom
  ALLOCATE(VIRT_CENTER(NVIRTA))
  !Returns the VS that an atom belongs to
  ALLOCATE(VIRT_VS_IND(NATOMS))
  !Temp array for reading info
  ALLOCATE(INDX_ATM(NVIRTA,MAX_ATOMS))
  !The number of atoms and mass of a virtual site is defined by the type of virtual site
  !ie all virtual sites have the same properties

  !The mass coefficient of atoms within a site can change as atoms can appear in a different
      order within VS
  !Eg -(C-C-N-C)- or -(C-N-C-C)-
  !The type of an atom within a virtual site will always have the same mass coefficient

  indx_atm = 0
  VIRT_VS_IND = 0

  !Read virtual site information
  VIRT_NUMATOMS = 0
  DO I=1,NVIRTA
     READ(10,*,iostat=iost) K, VITYPE(I), NUMATOM, VIRT_CENTER(I)
     ITYPE(NATOMS + I) = VITYPE(I)
     IF(iost .ne. 0) THEN
        WRITE(*,*) 'ERROR READING VIRTUAL, TOO SHORT'
        WRITE(1,*) 'ERROR READING VIRTUAL, TOO SHORT'
        ISTOP = 1
        RETURN
     END IF

     IF(VIRT_CENTER(I) .ne. 0) VIRT_VS_IND(VIRT_CENTER(I)) = I

     IF(NUMATOM .gt. MAX_ATOMS) THEN
        WRITE(*,*) 'Number of atoms in VS exceeds maximum, check virtual file'
        ISTOP = 1
        RETURN
     END IF

     READ(10,*) (INDX_ATM(I,J), J=1,NUMATOM)
```

```
74
      IF(VIRT_NUMATOMS(VITYPE(I)) .eq. 0) THEN !If number of atoms in VITYPE not assigned
         VIRT_NUMATOMS(VITYPE(I)) = NUMATOM
      ELSE IF(VIRT_NUMATOMS(VITYPE(I)) .ne. NUMATOM) THEN !If numatoms doesn't agree with old
          value
         WRITE(*,*) 'Disagreement in number of atoms in a VS type, check virtual file'
79       ISTOP = 1
         RETURN
      END IF
   END DO

84 !Transfer atom index info into smallest array possible
   ACTUAL_MAX = 0
   ACTUAL_MAX = MAXVAL(VIRT_NUMATOMS) !Find the maximum number of atoms in a VS
   ALLOCATE(VIRT_ATM_IND(NVIRTA,ACTUAL_MAX))
   VIRT_ATM_IND = 0
89
   DO I=1,NVIRTA
      DO J=1,VIRT_NUMATOMS(VITYPE(I))
         K = INDX_ATM(I,J)

94       VIRT_ATM_IND(I,J) = K
         !       VIRT_VS_IND(K) = I
      END DO
   END DO

99 !Calculate VS mass information
   !VS mass is taken as sum of atoms within and NOT the mass of the bead it represents

   VIRT_MASS = 0.0D0 !Mass of different types of virtual sites
   VIRT_INVMASS = 0.0D0 !Reciprocal of mass to reduce divide operations
104 VIRT_MASSCOEFF = 0.0D0 !VIRT_MASSCOEFF(virtual type, atom type) returns the mass coefficient
       for the atom

   DO I=1,NVIRTA
      TI = VITYPE(I)
      sumtotBmass = 0.0D0
109   DO J=1,VIRT_NUMATOMS(TI)
         TJ = ITYPE(VIRT_ATM_IND(I,J))
         sumtotBmass = sumtotBmass + MASS(TJ)
      END DO

114   IF(VIRT_MASS(TI) .eq. 0) THEN
         VIRT_MASS(TI) = sumtotBmass
         VIRT_INVMASS(TI) = 1.0D0 / sumtotBmass
      ELSE IF(ABS(VIRT_MASS(TI) - sumtotBmass) .gt. MASSTOL) THEN
         WRITE(*,*) 'Disagreement in VS mass, check virtual file'
119      WRITE(*,*) sumtotBmass, VIRT_MASS(TI)
         ISTOP = 1
         RETURN
      END IF

124   DO J=1,VIRT_NUMATOMS(TI)
         TJ = ITYPE(VIRT_ATM_IND(I,J))
         VIRT_MASSCOEFF(TI,TJ) = MASS(TJ)*VIRT_INVMASS(TI)
      END DO
   END DO
129
   DO I=1,NVIRTA
      TI = VITYPE(I)
      DO A=1,VIRT_NUMATOMS(TI)
         J = VIRT_ATM_IND(I,A)
134      TJ = ITYPE(J)
      END DO
   END DO

   RETURN
139 END SUBROUTINE RDVIRTUAL
```

src/RDVIRTBONDS.f90

```
1  SUBROUTINE RDVIRTBONDS()
   USE VAR
   IMPLICIT NONE

   INTEGER :: IOS = 0
6  INTEGER :: I, J

   !Reads the file 'virtbonds' if it exists and adds to the nonbonded exclusions
```

```
    !
    !'virtbonds' is a list of pairs of atoms that need to be exluded from nonbonded
    ! interactions the index given must be the global index of the atoms
    OPEN(4, IOSTAT=IOS, FILE='virtbonds', STATUS='OLD')

    IF(IOS .NE. 0) THEN !File is not compulsory
       CLOSE(4)
       RETURN
    END IF

    DO !Read infinitely until EOF
       READ(4,*,IOSTAT=IOS) I, J
       IF(IOS .NE. 0) EXIT

       CONNECTIONS(I) = CONNECTIONS(I) + 1
       IF(CONNECTIONS(I) .GT. MAXCONNECTIONS) THEN
          WRITE(*,*) 'Max number of connections exceeded with atom ',I
          WRITE(1,*) 'Max number of connections exceeded with atom ',I
          ISTOP = 1
          RETURN
       END IF
       CONNECTED_TO(I,CONNECTIONS(I)) = J

       CONNECTIONS(J) = CONNECTIONS(J) + 1
       IF(CONNECTIONS(J) .GT. MAXCONNECTIONS) THEN
          WRITE(*,*) 'Max number of connections exceeded with atom ',J
          WRITE(1,*) 'Max number of connections exceeded with atom ',J
          ISTOP = 1
          RETURN
       END IF
       CONNECTED_TO(J,CONNECTIONS(J)) = I

    END DO

    CLOSE(4)

    RETURN

END SUBROUTINE RDVIRTBONDS
```

src/RDVIRTANGLES.f90

```
SUBROUTINE RDVIRTANGLES()
  USE VAR
  IMPLICIT NONE

  INTEGER :: IOS = 0
  INTEGER :: I, J, K, t_ijk

  OPEN(4,IOSTAT=IOS, FILE='virtangles', STATUS='old')

  IF(IOS .ne. 0) THEN !File not compulsory
     CLOSE(4)
     RETURN
  END IF

  DO
     READ(4,*,IOSTAT=IOS) I, J, K, t_ijk
     IF(IOS .ne. 0) EXIT
     NIJK(I) = NIJK(I) + 1
     JANGLEIJK(I,NIJK(I)) = J
     KANGLEIJK(I,NIJK(I)) = K
  END DO
  CLOSE(4)
  RETURN
END SUBROUTINE RDVIRTANGLES
```

D.2 The MD Loop

src/NEW_LOOP.F90

```
SUBROUTINE NEW_LOOP()
```

```
   USE VAR
   USE OMP_LIB

6  IMPLICIT NONE

   DO STEP=1,NSTEP
      CALL SHIFT() !Applies PBC

11    IF(IBRDESCR .eq. 0) THEN
         CALL VIRTUAL_DEF() !Defines the position of virtual sites
      END IF

      !If needed, update neighbour list
16    IF(MOD(STEP,NUPDATE) .eq. 0) THEN
         CALL UPDATE_NEIGHBOURLIST()
      END IF

      !Calculate all forces on atoms
21    CALL NEW_FORCE()

      !Move atoms within box
      CALL MOVE()

26    !If needed, halt the net drift of box
      IF(MOD(STEP,HALT_DRIFT) .eq. 0) THEN
         CALL MOMENTUM()
      END IF

31    IF(ENSEMBLE .eq. 2) THEN !If NPT
         !Change box size and scale positions
         CALL SCALEBP(STEP)
      END IF

36    !STORING AVERAGE DATA AND RESTART FILE
      CALL AVERAGE()

      !STORING THE TRAJECTORY FILE
      IF (MOD(STEP, NTRJ) == 0) THEN
41       CALL WRITETRJ (STEP)
         CALL OUTPUT(STEP)
      END IF
   END DO

46 RETURN
END SUBROUTINE NEW_LOOP
```

D.3 Neighbourlists

src/NEW_NEIGHBOUR_WITHLIST.f90

```
   SUBROUTINE NEW_NEIGHBOUR_WITHLIST(INDEX_LIST,CELL,LCLIST,N,RLIST,LIST,MAXNAB &
        , MAP, SIZEMAP, HEAD, MAXNUMCELL)
3  USE VAR
   USE OMP_LIB

   IMPLICIT NONE

8  INTEGER, INTENT(IN) :: N !! @param n Number of particles to process
   INTEGER, INTENT(IN) :: MAXNUMCELL, MAXNAB
   INTEGER, INTENT(IN) :: INDEX_LIST(N) !! Array which translates onto the master coordinate list
   INTEGER, INTENT(IN) :: CELL(N), LCLIST(N), SIZEMAP, MAP(SIZEMAP), HEAD(MAXNUMCELL)
   INTEGER, INTENT(INOUT) :: LIST(MAXNAB,N)
13 INTEGER :: A, B, C, D, I, J, NLIST, JCELL, JCELL0, NABOR
   INTEGER :: NONBOND
   REAL*4, INTENT(IN) :: RLIST
   REAL*4, DIMENSION(3) :: RXYZ_I, RXYZ_IJ
   REAL*4 :: RIJSQ
18 REAL*4 :: RLISTSQ

   RLISTSQ = RLIST * RLIST
   LIST = 0 !Should set all unoccupied list slots to 0, can then use this to detect end of list

23 !$OMP PARALLEL DO DEFAULT(NONE), SCHEDULE(STATIC,1)&
```

```
!$OMP& SHARED(N,INDEX_LIST,CELL,RXYZ,LCLIST,CONNECTIONS,CONNECTED_TO,IBRDESCR,NATOMS)&
!$OMP& SHARED(BOX,BOXINV,RLISTSQ,LIST,MAXNAB,MAP,HEAD, NNEBS)&
!$OMP& PRIVATE(A,I,NLIST,JCELL,RXYZ_I,B,J,NONBOND,C,D,RXYZ_IJ,RIJSQ,JCELL0,NABOR)
DO A = 1,N
   I = INDEX_LIST(A)
   RXYZ_I(:) = RXYZ(:,I)

   NLIST = 0 !Counter for number of neighbours this atom has
   JCELL = CELL(A)

   B = LCLIST(A)
   DO
      IF(B .eq. 0) EXIT !If reached end of list
      J = INDEX_LIST(B) !Fetch real index of B

      NONBOND=1
      IF(CONNECTIONS(I) .gt. 0) THEN !If I has connections:
         DO C=1,CONNECTIONS(I) !Connections is the number of connected atoms to I
            D = CONNECTED_TO(I,C) !Connected_to has the index of all connected atoms
            IF(J .eq. D) THEN !If J is D then they are connected
               NONBOND=0
               EXIT
            END IF
         END DO
      END IF
      IF(IBRDESCR .eq. 0) THEN
         !Exclude VS-VS interactions
         IF(I .gt. NATOMS .and. J .gt. NATOMS) THEN
            NONBOND = 0
         END IF
      END IF

      IF (NONBOND.EQ.1) THEN
         RXYZ_IJ(:) = RXYZ_I(:) - RXYZ(:,J)

         RXYZ_IJ(:) = RXYZ_IJ(:) - ANINT(RXYZ_IJ(:) * BOXINV(:)) * BOX(:)
         RIJSQ = SUM(RXYZ_IJ(:) * RXYZ_IJ(:))

         IF (RIJSQ.LT.RLISTSQ) THEN
            NLIST = NLIST + 1
            LIST(NLIST,A) = J
            IF (NLIST.EQ.MAXNAB) STOP 'LIST TOO SMALL'
         ENDIF
      ENDIF
      B = LCLIST(B)
   END DO

   ! LOOP IN THE NEIGHBOURING CELLS
   JCELL0 = 13*(JCELL - 1)
   DO NABOR = 1,13
      JCELL = MAP(JCELL0+NABOR)
      B = HEAD(JCELL)

      DO
         IF(B .eq. 0) EXIT
         J = INDEX_LIST(B)
         NONBOND=1
         IF(CONNECTIONS(I) .gt. 0) THEN !If I has connections:
            DO C=1,CONNECTIONS(I)
               D = CONNECTED_TO(I,C)
               IF(J .eq. D) THEN !If J is C then they are connected
                  NONBOND=0
                  EXIT
               END IF
            END DO
         END IF
         IF(IBRDESCR .eq. 0) THEN
            IF(I .gt. NATOMS .and. J .gt. NATOMS) THEN
               NONBOND = 0
            END IF
         END IF

         IF (NONBOND.EQ.1) THEN
            RXYZ_IJ(:) = RXYZ_I(:) - RXYZ(:,J)
            RXYZ_IJ(:) = RXYZ_IJ(:) - ANINT(RXYZ_IJ(:) * BOXINV(:)) * BOX(:)
            RIJSQ = SUM(RXYZ_IJ(:) * RXYZ_IJ(:))

            IF (RIJSQ.LT.RLISTSQ) THEN
               NLIST = NLIST + 1
               LIST(NLIST,A) = J
```

```
                IF (NLIST.EQ.MAXNAB) STOP 'LIST TOO SMALL'
              ENDIF
            ENDIF
            B = LCLIST(B)
          END DO
        END DO
        ! Save number of neighbours this particle has
        NNEBS(I) = NLIST
      END DO
      !$OMP END PARALLEL DO

      RETURN
    END SUBROUTINE NEW_NEIGHBOUR_WITHLIST
```

D.4 Calculating Force

src/NONBONDED_FORCE.F90

```
    SUBROUTINE NONBONDED_FORCE(N,INDEX_LIST,MAXNAB,LIST,RCUT,RCUTSQ)
      USE VAR
      USE OMP_LIB

      IMPLICIT NONE

      INTEGER, INTENT(IN) :: N !< Size of the particle list you have passed to it
      INTEGER, INTENT(IN) :: INDEX_LIST(N) !< Index containing the address in master array of
        !!each particle in this group
      INTEGER, INTENT(IN) :: MAXNAB !< The maximum number of neighbours that a particle could ever
          have
      INTEGER, INTENT(IN) :: LIST(MAXNAB,N) !< 2d array, first dimension goes over all particles,
        !! second dimension contains neighbours for
        !!              this particle
      REAL*4, INTENT(IN) :: RCUT !< Cutoff radius for nonbonded interactions
      REAL*4, INTENT(IN) :: RCUTSQ !< Cutoff radius squared
      INTEGER :: A, B, I, J, TI, TJ, TIJ
      INTEGER :: JNAB
      INTEGER :: NI
      REAL*4, DIMENSION(3) :: FXYZ_I
      REAL*4, DIMENSION(3) :: RXYZ_I, RXYZ_IJ
      REAL*4 :: RIJSQ, RIJ
      REAL*4 :: ALPHA, FIJ, VIJ

      !$OMP PARALLEL DO DEFAULT(NONE) SCHEDULE(STATIC,1)&
      !$OMP& SHARED(N,B,NITEMS,INDEX_LIST,LIST,ITYPE,INBONDT,NNEBS)&
      !$OMP& SHARED(RXYZ,BOXINV,BOX,RCUTSQ)&
      !$OMP& SHARED(BINNB,RNBOND,NBOND_FORCE,NBOND_POT,TYPE_LABEL)&
      !$OMP& PRIVATE(A,I,TI,J,TJ,TIJ,JNAB)&
      !$OMP& PRIVATE(FXYZ_I,RXYZ_I,RXYZ_IJ,RIJSQ,RIJ)&
      !$OMP& PRIVATE(NI,ALPHA,FIJ,VIJ)&
      !$OMP& REDUCTION(+:V_NB)&
      !$OMP& REDUCTION(+:PT11,PT22,PT33,PT12,PT13,PT23) &
      !$OMP& REDUCTION(+:FXYZ)
      DO A=1,N
        I = INDEX_LIST(A) !I is the index of atom being considered
        RXYZ_I(:) = RXYZ(:,I)
        FXYZ_I(:) = 0.0

        TI = ITYPE(I)

        DO JNAB = 1, NNEBS(I)
          J = LIST(JNAB,A)
          TJ = ITYPE(J)
          TIJ = INBONDT(TI,TJ)

          RXYZ_IJ(:) = RXYZ_I(:) - RXYZ(:,J)
          RXYZ_IJ(:) = RXYZ_IJ(:) - ANINT(RXYZ_IJ(:) * BOXINV(:)) * BOX(:)
          RIJSQ = SUM(RXYZ_IJ(:) * RXYZ_IJ(:))

          IF(RIJSQ .LT. RCUTSQ) THEN
            RIJ = SQRT(RIJSQ)

            NI = INT(RIJ / BINNB(TIJ))

            ALPHA = (RIJ - RNBOND(NI,TIJ)) / BINNB(TIJ)
```

```
            FIJ = NBOND_FORCE(NI,TIJ)*(1.0 - ALPHA) &
                + NBOND_FORCE(NI+1,TIJ)*ALPHA

            VIJ = NBOND_POT(NI,TIJ)*(1.0 - ALPHA) &
                + NBOND_POT(NI+1,TIJ)*ALPHA

            IF(TYPE_LABEL(I) .eq. 1) THEN !ATOM
               V_NB(1) = V_NB(1) + VIJ
            ELSE IF(TYPE_LABEL(I) .eq. 2) THEN !BEAD
               IF(TYPE_LABEL(J) .eq.2) THEN !BOTH BEADS
                  V_NB(2) = V_NB(2) + VIJ
               ELSE IF(TYPE_LABEL(J) .eq. 3) THEN!MIXED
                  V_NB(3) = V_NB(3) + VIJ
               END IF
            ELSE IF(TYPE_LABEL(I) .eq. 3) THEN !VS
               V_NB(3) = V_NB(3) + VIJ
            END IF

            FXYZ_I(:) = FXYZ_I(:) + FIJ * RXYZ_IJ(:)
            FXYZ(:,J) = FXYZ(:,J) - FIJ * RXYZ_IJ(:)

            PT11 = PT11 + FIJ * RXYZ_IJ(1) * RXYZ_IJ(1)
            PT22 = PT22 + FIJ * RXYZ_IJ(2) * RXYZ_IJ(2)
            PT33 = PT33 + FIJ * RXYZ_IJ(3) * RXYZ_IJ(3)

            PT12 = PT12 + FIJ * RXYZ_IJ(2) * RXYZ_IJ(1)
            PT13 = PT13 + FIJ * RXYZ_IJ(3) * RXYZ_IJ(1)
            PT23 = PT23 + FIJ * RXYZ_IJ(3) * RXYZ_IJ(2)

         END IF !End if RIJSQ lt RCUTSQ
      END DO !End loop over neighbours

      FXYZ(:,I) = FXYZ(:,I) + FXYZ_I(:)
   END DO !End loop over all items
   !$OMP END PARALLEL DO

   RETURN
END SUBROUTINE NONBONDED_FORCE
```

src/BONDED_FORCE.F90

```
SUBROUTINE BONDED_FORCE()
   USE VAR
   USE OMP_LIB

   IMPLICIT NONE

   INTEGER :: A
   INTEGER :: I, J, K, L, TI, TJ, TK, TL, TIJ, TIJK, TIJKL, TOIJKL
   INTEGER :: NI
   REAL*4, DIMENSION(3) :: RXYZ_I, RXYZ_IJ
   REAL*4, DIMENSION(3) :: FXYZ_I
   REAL*4 :: RIJSQ, RIJ
   REAL*4, DIMENSION(3) :: RXYZ_KJ, RXYZ_JK, RXYZ_KL
   REAL*4 :: RKJ, RJK, RKL
   REAL*4 :: THETA, COSM, SINM, COTANM, COSN, SINN, COTANN, COST, SIGNT, PHI
   REAL*4 :: CT, eps = 1.0D-7, PHI_t
   REAL*4, DIMENSION(3) :: RXYZ_M, RXYZ_N, RXYZ_S
   REAL*4 :: RM, RN
   REAL*4 :: ALPHA, FIJ, VIJ, FIJK, VIJK, FIJKL, VIJKL
   REAL*4, DIMENSION(3) :: FXYZ_IJ, FXYZ_IJK
   REAL*4, DIMENSION(3) :: FXYZ_1, FXYZ_4, FXYZ_12
   REAL*4, DIMENSION(3) :: RXYZ_mN

   !$OMP PARALLEL DO DEFAULT(NONE) &
   !$OMP& SHARED(NITEMS,NATOMS,SXYZ,ITYPE,STEP,R2D)&
   !$OMP& SHARED(NBONDS,JBOND,IBONDT,BINB,NDATB,RBOND,BOND_FORCE,BOND_POT)&
   !$OMP& SHARED(NIJK,JANGLEIJK,KANGLEIJK,IANGT,BINA,ANGLE,BEND_FORCE,BEND_POT)&
   !$OMP& SHARED(NIJKL,JTORIJKL,KTORIJKL,LTORIJKL,ITORT,BINT,NDATT,ANGLE_TOR)&
   !$OMP& SHARED(TOR_FORCE,TOR_POT)&
   !$OMP& SHARED(NOOPIJKL,JOOPIJKL,KOOPIJKL,LOOPIJKL,IOOPT,BINO,NDATO,ANGLE_OOP)&
   !$OMP& SHARED(OOP_FORCE,OOP_POT, EPS)&
   !$OMP& SHARED(BOND_TYPE_LABEL, ANGLE_TYPE_LABEL)&
   !$OMP& SHARED(TORSION_TYPE_LABEL, OOP_TYPE_LABEL)&
   !$OMP& PRIVATE(A, I, J, K, L, TI, TJ, TK, TL, TIJ, TIJK, TIJKL, NI, TOIJKL)&
   !$OMP& PRIVATE(RXYZ_I, FXYZ_I)&
   !$OMP& PRIVATE(RXYZ_IJ, RIJSQ, RIJ)&
   !$OMP& PRIVATE(RXYZ_KJ, RKJ)&
```

```
!$OMP& PRIVATE(RXYZ_JK, RJK, RXYZ_KL, RKL)&
!$OMP& PRIVATE(THETA, COSM, SINM, COTANM, COSN, SINN, COTANN, COST, SIGNT, PHI)&
!$OMP& PRIVATE(CT, PHI_t)&
!$OMP& PRIVATE(RXYZ_M, RM, RXYZ_N, RN, RXYZ_S)&
!$OMP& PRIVATE(ALPHA, FIJ, VIJ, FIJK, VIJK, FIJKL, VIJKL)&
!$OMP& PRIVATE(FXYZ_IJ, FXYZ_IJK)&
!$OMP& PRIVATE(FXYZ_1, FXYZ_4, FXYZ_12)&
!$OMP& PRIVATE(RXYZ_mN)&
!$OMP& REDUCTION(+:V_BOND, V_ANGLE, V_TORSION, V_OOP)&
!$OMP& REDUCTION(+:FXYZ)
DO I=1,NITEMS
    RXYZ_I(:) = SXYZ(:, I)
    FXYZ_I = 0.0
    TI = ITYPE(I)

    DO A=1,NBONDS(I)
        J = JBOND(I,A)
        IF (J > I) THEN
            TJ = ITYPE(J)
            TIJ = IBONDT(TI, TJ)
            RXYZ_IJ(:) = RXYZ_I(:) - SXYZ(:,J)
            RIJSQ = SUM(RXYZ_IJ * RXYZ_IJ)
            RIJ = SQRT(RIJSQ)
            NI = INT((RIJ - RBOND(TIJ,0)) / BINB(TIJ))

            !If doesn't fall within table, use last/first entry
            IF(NI .GT. NDATB(TIJ)) THEN
               WRITE(*,*) 'Overextended bond between atoms ', I, ' and ', J
               WRITE(*,*) 'Distance ', RIJ, ' Timestep ', STEP
               WRITE(*,*) 'Continuing on, but this might be a problem!'
               WRITE(1,*) 'Overextended bond between atoms ', I, ' and ', J
               WRITE(1,*) 'Distance ', RIJ, ' Timestep ', STEP
               WRITE(1,*) 'Continuing on, but this might be a problem!'
               NI = NDATB(TIJ)
            ELSE IF(NI .LT. 0) THEN
               NI = 0
            END IF

            ALPHA=(RIJ-RBOND(TIJ,NI))/BINB(TIJ)
            FIJ = BOND_FORCE(TIJ,NI)*(1.0D0-ALPHA) &
                  + ALPHA*BOND_FORCE(TIJ,NI+1)
            VIJ = BOND_POT(TIJ,NI)*(1.0D0-ALPHA) &
                  + ALPHA*BOND_POT(TIJ,NI+1)

            V_BOND(BOND_TYPE_LABEL(TIJ)) = V_BOND(BOND_TYPE_LABEL(TIJ)) + VIJ

            FXYZ_IJ(:)  = FIJ * RXYZ_IJ(:)
            FXYZ_I(:) = FXYZ_I(:) + FXYZ_IJ(:)
            FXYZ(:,J) = FXYZ(:,J) - FXYZ_IJ(:)
        END IF
    END DO

    DO A = 1, NIJK(I)
        J = JANGLEIJK(I,A)
        K = KANGLEIJK(I,A)

        TJ = ITYPE(J)
        TK = ITYPE(K)
        TIJK = IANGT(TI, TJ, TK)

        RXYZ_IJ(:) = RXYZ_I(:) - SXYZ(:, J)
        RXYZ_KJ(:) = SXYZ(:, K) - SXYZ(:, J)

        RIJ = SQRT(SUM(RXYZ_IJ** 2.0))
        RKJ = SQRT(SUM(RXYZ_KJ** 2.0))
        CT = (RXYZ_IJ * RXYZ_KJ)/RIJ/RKJ
        THETA = ACOS(CT) * R2D

        RXYZ_M = CROSS(RXYZ_IJ, RXYZ_KJ)
        RM = SQRT(SUM(RXYZ_M ** 2.0))

        RXYZ_M = RXYZ_M / RM

        RXYZ_N = CROSS(RXYZ_M, RXYZ_IJ)
        RXYZ_S = CROSS(RXYZ_M, RXYZ_KJ)

        NI = INT (THETA / BINA(TIJK))
        IF (THETA == 180.0) THEN
            FIJK = BEND_FORCE(TIJK,NI)
            VIJK = BEND_POT(TIJK,NI)
```

```
            ELSE
               ALPHA=(THETA-ANGLE(TIJK,NI))/BINA(TIJK)

               FIJK = BEND_FORCE(TIJK,NI)*(1.0D0-ALPHA) &
                    + ALPHA*BEND_FORCE(TIJK,NI+1)

               VIJK = BEND_POT(TIJK,NI)*(1.0D0-ALPHA) &
                    + ALPHA*BEND_POT(TIJK,NI+1)
            END IF

            V_ANGLE(ANGLE_TYPE_LABEL(TIJK)) = V_ANGLE(ANGLE_TYPE_LABEL(TIJK)) + VIJK

            FXYZ_IJK(:)  = FIJK * RXYZ_N(:) / RIJ
            FXYZ_I(:)    = FXYZ_I(:) + FXYZ_IJK(:)
            FXYZ(:, K) = FXYZ(:, K) + FIJK * RXYZ_S(:) / RKJ
            FXYZ(:, J) = FXYZ(:, J) - FIJK * RXYZ_S(:) / RKJ - FXYZ_IJK(:)
         END DO

         DO  A = 1, NIJKL(I)
            J = JTORIJKL(I, A)
            K = KTORIJKL(I, A)
            L = LTORIJKL(I, A)

            TJ = ITYPE(J)
            TK = ITYPE(K)
            TL = ITYPE(L)
            TIJKL = ITORT(TI, TJ, TK, TL)

            RXYZ_IJ(:) = RXYZ_I(:) - SXYZ(:, J)
            RXYZ_JK(:) = SXYZ(:, J) - SXYZ(:, K)
            RXYZ_KL(:) = SXYZ(:, K) - SXYZ(:, L)

            RIJ = SQRT(SUM(RXYZ_IJ ** 2.0))
            RJK = SQRT(SUM(RXYZ_JK ** 2.0))
            RKL = SQRT(SUM(RXYZ_KL ** 2.0))

            RXYZ_M = CROSS(RXYZ_IJ, RXYZ_JK)
            RM = SQRT(SUM(RXYZ_M ** 2.0))

            !if RM = 0 IJ & JK are parallel
            if (RM.GT.0) THEN
               RXYZ_M(:) = RXYZ_M(:) / RM
            else if (RM .EQ. 0) then
               RXYZ_IJ(1) = RXYZ_I(1)*0.99 - SXYZ(1,J)
               RIJ = SQRT(SUM(RXYZ_IJ ** 2.0)
               RXYZ_M = CROSS(RXYZ_IJ, RXYZ_JK)
               RM = SQRT(SUM(RXYZ_M ** 2.0))
               RXYZ_M(:) = RXYZ_M(:) / RM
            end if

            COSM = RXYZ_IJ * RXYZ_JK
            SINM = SQRT(1-COSM*COSM)
            COTANM = COSM/SINM

            RXYZ_N = CROSS(RXYZ_JK, RXYZ_KL)
            RN = SQRT(SUM(RXYZ_N ** 2.0))

            if (RN .GT. 0) then
               RXYZ_N(:) = RXYZ_N(:) / RN
               COSN = RXYZ_JK * RXYZ_KL
            elseif (RN .EQ. 0)then
               RXYZ_KL(1) = SXYZ(1,K) - SXYZ(1,L)*0.99
               RKL = SQRT(SUM(RXYZ_KL ** 2.0))
               RXYZ_N = CROSS(RXYZ_JK, RXYZ_KL)
               RN = SQRT(SUM(RXYZ_N ** 2.0))
               RXYZ_N(:) = RXYZ_N(:) / RN
               COSN = RXYZ_JK * RXYZ_KL
            endif
            SINN = SQRT(1-COSN*COSN)
            COTANN = COSN/SINN

            COST = RXYZ_M * RXYZ_N
            SIGNT = - RXYZ_M * RXYZ_KL

            if(COST .GT. 1.0) then
               COST = 1.0
            else if (COST .LT. -1.0) THEN
               COST = -1.0
            end if
```

```
        PHI = ACOS(COST)*R2D

        IF (SIGNT.LT.0) PHI=360. - PHI

        NI = INT (PHI / BINT(TIJKL))

        IF(NI .GT. NDATT(TIJKL)) THEN
           WRITE(*,*)'FATAL ERROR: Entry in torsion table', TIJKL,' does not exist'
           WRITE(1,*)'FATAL ERROR: Entry in torsion table', TIJKL,' does not exist'
           STOP
        END IF

        IF (PHI == 360.0) THEN
           FIJKL = TOR_FORCE(TIJKL,NI)
           VIJKL = TOR_POT(TIJKL,NI)
        ELSE
           ALPHA=(PHI-ANGLE_TOR(TIJKL,NI))/BINT(TIJKL)
           FIJKL = TOR_FORCE(TIJKL,NI)*(1.0-ALPHA) &
                + ALPHA*TOR_FORCE(TIJKL,NI+1)
           VIJKL = TOR_POT(TIJKL,NI)*(1.0-ALPHA) &
                + ALPHA*TOR_POT(TIJKL,NI+1)
        END  IF

        V_TORSION(TORSION_TYPE_LABEL(TIJKL)) = V_TORSION(TORSION_TYPE_LABEL(TIJKL)) + VIJKL

        FXYZ_1(:) = -FIJKL * RXYZ_M(:)/SINM/RIJ
        FXYZ_I(:) = FXYZ_I(:) + FXYZ_1(:)
        FXYZ_4(:) = FIJKL * RXYZ_N(:)/SINN/RKL
        FXYZ(:,L) = FXYZ(:,L) + FXYZ_4(:)
        FXYZ_12(:) = FIJKL*(COTANM*RXYZ_M(:)+COTANN*RXYZ_N(:))/RJK
        FXYZ(:,J) = FXYZ(:,J) - FXYZ_1(:) + FXYZ_12(:)
        FXYZ(:,K) = FXYZ(:,K) - FXYZ_4(:) - FXYZ_12(:)
     END DO

     DO A = 1, NOOPIJKL(I)
        J = JOOPIJKL(I, A)
        K = KOOPIJKL(I, A)
        L = LOOPIJKL(I, A)

        TJ = ITYPE(J)
        TK = ITYPE(K)
        TL = ITYPE(L)
        TOIJKL = IOOPT(TI,TJ,TK,TL)

        RXYZ_IJ(:) = RXYZ_I(:) - SXYZ(:, J)
        RXYZ_JK(:) = SXYZ(:, K) - SXYZ(:, J)
        RXYZ_KL(:) = SXYZ(:, K) - SXYZ(:, L)

        RIJ = SQRT(SUM(RXYZ_IJ ** 2.0))
        RJK = SQRT(SUM(RXYZ_JK ** 2.0))
        RKL = SQRT(SUM(RXYZ_KL ** 2.0))

        RXYZ_M = CROSS(RXYZ_IJ, RXYZ_JK)
        RM = SQRT(SUM(RXYZ_M ** 2.0))

        RXYZ_M(:) = RXYZ_M(:) / RM
        COSM = RXYZ_IJ * RXYZ_JK
        SINM = SQRT(1-COSM*COSM)
        COTANM = COSM/SINM

        RXYZ_N = CROSS(RXYZ_JK, RXYZ_KL)
        RN = SQRT(SUM(RXYZ_N ** 2.0))

        RXYZ_N(:) = RXYZ_N(:) / RN

        COSN = RXYZ_JK * RXYZ_KL
        SINN = SQRT(1-COSN*COSN)
        COTANN = COSN/SINN

        COST = RXYZ_M * RXYZ_N

        RXYZ_mN = CROSS(RXYZ_M, RXYZ_N)

        SIGNT = sign(1.0, RXYZ_mN ** 2.0)

        IF (COST .GT. 1.0 + eps) THEN
           write(*,*) i,j,k,l
           write(*,*)COST
           stop 'COST .GT. 1.00'
        else if(COST .lT. -(1.00+eps))THEN
```

```
            write(*,*) i,j,k,l
            write(*,*)COST
            stop 'COST .lT. 1.00'
          END IF

          IF (COST .GT. 1.0 ) THEN
             COST=1.0
          else if(COST .lT. -1.00)THEN
             COST=-1.0
          END IF

          PHI = ACOS(COST)
          PHI = signt * PHI * R2D

          PHI_t= 180.0 + PHI
          NI = anint(PHI_t / BINO(TOIJKL))

          if(NI .lt. 0)then
             write(*,*) i,j,k,l,signt
             write(*,*) NI,phi_t
             write(*,*)PHI_t / BINO(TOIJKL)
             stop
          end if

          IF(NI .GT. NDATO(TOIJKL)) THEN
             WRITE(*,*)'FATAL ERROR: Entry in out of plane table', TOIJKL,' does not exist'
             WRITE(1,*)'FATAL ERROR: Entry in out of plane table', TOIJKL,' does not exist'
             write(*,*) i,j,k,l
             write(*,*) NI,phi
             WRITE(*,*)'Simulation stopped at Time Step: ',STEP
             STOP
          END IF

          ALPHA=abs(abs(PHI)-abs(ANGLE_OOP(TOIJKL,NI)))/BINO(TOIJKL)
          if(alpha .gt. 1.0)then
             ni = ni+1
             ALPHA=abs(abs(PHI)-abs(ANGLE_OOP(TOIJKL,NI)))/BINO(TOIJKL)
          end if

          FIJKL = OOP_FORCE(TOIJKL,NI)*(1.0-ALPHA)+ ALPHA*OOP_FORCE(TOIJKL,NI+1)
          VIJKL = OOP_POT(TOIJKL,NI)*(1.0-ALPHA)+ ALPHA*OOP_POT(TOIJKL,NI+1)

          V_OOP(OOP_TYPE_LABEL(TOIJKL)) = V_OOP(OOP_TYPE_LABEL(TOIJKL)) + VIJKL

          FXYZ_1(:) = FIJKL * RXYZ_M(:)/SINM/RIJ
          FXYZ_I(:) = FXYZ_I(:) + FXYZ_1(:)
          FXYZ_4(:) = -FIJKL * RXYZ_N(:)/SINN/RKL
          FXYZ(:, L) = FXYZ(:, L) + FXYZ_4(:)
          FXYZ_12(:) = FIJKL*(COTANM*RXYZ_M(:)+COTANN*RXYZ_N(:))/RJK
          FXYZ(:, J) = FXYZ(:, J) - FXYZ_1(:) + FXYZ_12(:)
          FXYZ(:, K) = FXYZ(:, K) - FXYZ_4(:) - FXYZ_12(:)
        END DO

        FXYZ(:,I) = FXYZ(:,I) + FXYZ_I(:)
      END DO
      !$OMP END PARALLEL DO

      RETURN

    END SUBROUTINE BONDED_FORCE

    FUNCTION CROSS(A, B)
      REAL(KIND=RKIND), DIMENSION(3) :: CROSS
      REAL(KIND=RKIND), DIMENSION(3), INTENT(IN) :: A, B

      CROSS(1) = A(2) * B(3) - A(3) * B(2)
      CROSS(2) = A(3) * B(1) - A(1) * B(3)
      CROSS(3) = A(1) * B(2) - A(2) * B(1)
    END FUNCTION CROSS
```

D.5 Reporting results

src/WRITETRJ.f90

```
SUBROUTINE WRITETRJ (TM)
  USE VAR
  IMPLICIT NONE

  INTEGER, INTENT(IN) :: TM !< Current time step
  INTEGER :: I, J
  REAL*8 :: PRESS, PT_11, PT_12, PT_13, PT_22, PT_23, PT_33
  REAL*8 :: PTOT, ETOT, T, TREAL
  REAL*8 :: BOX_X, BOX_Y, BOX_Z
  REAL*8 :: smallnumber
  REAL*4, POINTER :: NSX(:), NSY(:), NSZ(:), NVX(:), NVY(:), NVZ(:)
  INTEGER*4 :: nrec
  CHARACTER(80) :: YASPTITLE

  data nrec / 10 /

  NFRAME = NFRAME + 1

  ALLOCATE(NSX(NATOMS))
  ALLOCATE(NSY(NATOMS))
  ALLOCATE(NSZ(NATOMS))
  ALLOCATE(NVX(NATOMS))
  ALLOCATE(NVY(NATOMS))
  ALLOCATE(NVZ(NATOMS))

  !      CONVERT TO 4-BYTE PRECISION (COORDINATES AND VELOCITES ONLY)
  smallnumber = 1.0D0 / 1.0D+12 / TIMESCALE

  DO J = 1, NATOMS
     NSX(J) = SX(J)
     NSY(J) = SY(J)
     NSZ(J) = SZ(J)

     NVX(J) = VX(J) * smallnumber
     NVY(J) = VY(J) * smallnumber
     NVZ(J) = VZ(J) * smallnumber
  END DO

  ! write header record (together with first frame)
  IF (NFRAME == 1) THEN
     YASPTITLE = TITLE
     WRITE(113)YASPTITLE
     WRITE(113)nrec
  END IF
  TREAL = TM * DT * TIMESCALE * 1.0D+12 + INITIME

  WRITE(113) NFRAME, NTRJ*NFRAME, NATOMS, TREAL

  ! simulation cell unit vectors
  BOX_X = BOXX
  BOX_Y = BOXY
  BOX_Z = BOXZ
  WRITE(113) BOX_X, 0.0D00, 0.0D00, &
       0.0D00, BOX_Y, 0.0D00, &
       0.0D00, 0.0D00, BOX_Z

  ! isotropic pressure and pressure tensor
  PRESS = (PT11+PT22+PT33)*PSCALE/3.0d0
  PT_11 = PT11*PSCALE
  PT_12 = PT12*PSCALE
  PT_13 = PT13*PSCALE
  PT_22 = PT22*PSCALE
  PT_23 = PT23*PSCALE
  PT_33 = PT33*PSCALE
  WRITE(113) PRESS,       &
       PT_11,             &
       PT_12, PT_22,      &
       PT_13, PT_23, PT_33

  PTOT = SUM(V_BOND) + SUM(V_ANGLE) + SUM(V_TORSION) + SUM(V_OOP) + SUM(V_NB)
  ETOT = PTOT + SUM(EK)
  T = SUM(EK) * MKTEMP

  WRITE(113) 6, ETOT*CONV, PTOT*CONV, SUM(EK)*CONV, T, 0.0D00, 0.0D00
  WRITE(113)NSX
  WRITE(113)NSY
  WRITE(113)NSZ
  WRITE(113)NVX
  WRITE(113)NVY
  WRITE(113)NVZ
```

```
  OPEN(UNIT=114, FILE = 'config.xyz', STATUS='replace')
  WRITE(114,*)NATOMS
  WRITE(114,*) 't'
  DO I = 1, NATOMS
     WRITE(114,9050) LABEL(ITYPE(I)), NSX(I)*10, NSY(I)*10, NSZ(I)*10
  END DO
9050 FORMAT (1X,A8,1X,3 (G21.14,1X))
  CLOSE (114)

  DEALLOCATE(NSX)
  DEALLOCATE(NSY)
  DEALLOCATE(NSZ)
  DEALLOCATE(NVX)
  DEALLOCATE(NVY)
  DEALLOCATE(NVZ)

  RETURN

END SUBROUTINE WRITETRJ
```

src/OUTPUT.F90

```
SUBROUTINE OUTPUT (I)
  USE VAR
  IMPLICIT NONE

  INTEGER :: I
  REAL*8 :: TREAL
  REAL*8 :: AV_CONV, AV_PSCALE, INV_AV

  INV_AV = 1.0 / NTRJ
  AV_CONV = INV_AV * CONV
  AV_PSCALE = INV_AV * PSCALE

  TREAL = I * DT * TIMESCALE * 1.0D+12 + INITIME !ps

  WRITE(*,*) I, TEMP*TEMPSCALE, PRES(1)*PSCALE

  WRITE (115, *)'Step:                          ', I
  WRITE (115, 100)'Simulated_time:              ', TREAL
  WRITE (115, 100)'Total_energy:                ', TOT_E*CONV, AV_TOT_E*AV_CONV
  WRITE (115, 100)'Potential_energy:            ', POT_E*CONV, AV_POT_E*AV_CONV
  WRITE (115, 100)'Kinetic_energy:              ', SUM(KIN_E)*CONV, SUM(AV_KIN_E)*AV_CONV
  WRITE (115, 100)'       Kinetic_energy_atom:  ', KIN_E(1)*CONV, AV_KIN_E(1)*CONV
  WRITE (115, 100)'       Kinetic_energy_bead:  ', KIN_E(2)*CONV, AV_KIN_E(2)*CONV
  WRITE (115, 100)'Tot._Nonbonded_energy:           ', SUM(V_NB)*CONV, SUM(AV_V_NB)*AV_CONV
  WRITE (115, 100)'       Nonbonded_Atom_energy:    ', V_NB(1)*CONV, AV_V_NB(1)*AV_CONV
  WRITE (115, 100)'       Nonbonded_Beads_energy:   ', V_NB(2)*CONV, AV_V_NB(2)*AV_CONV
  WRITE (115, 100)'       Nonbonded_mix_energy:     ', V_NB(3)*CONV, AV_V_NB(3)*AV_CONV
  WRITE (115, 100)'Tot._Bond_energy:            ', SUM(V_BOND)*CONV, SUM(AV_V_BOND)*AV_CONV
  WRITE (115, 100)'       Bond_Atom_energy:     ', V_BOND(1)*CONV, AV_V_BOND(1)*AV_CONV
  WRITE (115, 100)'       Bond_Beads_energy:    ', V_BOND(2)*CONV, AV_V_BOND(2)*AV_CONV
  WRITE (115, 100)'       Bond_mix_energy:      ', V_BOND(3)*CONV, AV_V_BOND(3)*AV_CONV
  WRITE (115, 100)'Tot._Angle_energy:           ', SUM(V_ANGLE)*CONV, SUM(AV_V_ANGLE)*AV_CONV
  WRITE (115, 100)'       Angle_Atom_energy:    ', V_ANGLE(1)*CONV, AV_V_ANGLE(1)*AV_CONV
  WRITE (115, 100)'       Angle_Beads_energy:   ', V_ANGLE(2)*CONV, AV_V_ANGLE(2)*AV_CONV
  WRITE (115, 100)'       Angle_mix_energy:     ', V_ANGLE(3)*CONV, AV_V_ANGLE(3)*AV_CONV
  WRITE (115, 100)'Tot._Torsion energy:         ', SUM(V_TORSION)*CONV, SUM(AV_V_TORSION)*AV_CONV
  WRITE (115, 100)'      Torsion_Atom_energy:   ', V_TORSION(1)*CONV, AV_V_TORSION(1)*AV_CONV
  WRITE (115, 100)'      Torsion_Beads_energy:  ', V_TORSION(2)*CONV, AV_V_TORSION(2)*AV_CONV
  WRITE (115, 100)'      Torsion_mix_energy:    ', V_TORSION(3)*CONV, AV_V_TORSION(3)*AV_CONV
  WRITE (115, 100)'Improper_torsion_energy:     ', SUM(V_OOP)*CONV, SUM(AV_V_OOP)*AV_CONV
  WRITE (115, 100)'Temperature:                 ', TEMP*TEMPSCALE, AV_TEMP*INV_AV*TEMPSCALE
  WRITE (115, 100)'Pressure:                    ', PRES(1)*PSCALE, AV_PRES(1)*AV_PSCALE
  WRITE (115, 100)'Pressure(x):                 ', PRES(2)*PSCALE, AV_PRES(2)*AV_PSCALE
  WRITE (115, 100)'Pressure(y):                 ', PRES(3)*PSCALE, AV_PRES(3)*AV_PSCALE
  WRITE (115, 100)'Pressure(z):                 ', PRES(4)*PSCALE, AV_PRES(4)*AV_PSCALE
  WRITE (115, 100)'Pressure(xy):                ', PRES(5)*PSCALE, AV_PRES(5)*AV_PSCALE
  WRITE (115, 100)'Pressure(xz):                ', PRES(6)*PSCALE, AV_PRES(6)*AV_PSCALE
  WRITE (115, 100)'Pressure(yz):                ', PRES(7)*PSCALE, AV_PRES(7)*AV_PSCALE
  WRITE (115, 100)'Box_volume:                  ', BOXSIZE(1), AV_BOXSIZE(1)*INV_AV
  WRITE (115, 100)'Box_length(x):               ', BOXSIZE(2), AV_BOXSIZE(2)*INV_AV
  WRITE (115, 100)'Box_length(y):               ', BOXSIZE(3), AV_BOXSIZE(3)*INV_AV
  WRITE (115, 100)'Box_length(z):               ', BOXSIZE(4), AV_BOXSIZE(4)*INV_AV
  WRITE (115, 100)'Mass_density:                ', DENS * DSCALE, AV_DENS*INV_AV*DSCALE
  WRITE (115, 100)
  WRITE (115, 100)
```

```
   FLUSH(115)

100 format (A,2(F16.5,2X))

   !Reset average totals to 0 to begin new averaging window
   AV_TOT_E = 0.0
   AV_POT_E = 0.0
   AV_KIN_E = 0.0
   AV_V_NB = 0.0
   AV_V_BOND = 0.0
   AV_V_ANGLE = 0.0
   AV_V_TORSION = 0.0
   AV_V_OOP = 0.0
   AV_TEMP = 0.0
   AV_PRES = 0.0
   AV_BOXSIZE = 0.0
   AV_DENS = 0.0

   RETURN
END SUBROUTINE OUTPUT
```

src/WRITEPSF.f90

```
SUBROUTINE WRITEPSF()
   USE VAR
   IMPLICIT NONE

   INTEGER :: I, J, A
   INTEGER :: NBONDS_PSF, NTHETA, NPHI, NOOP !Number of each topology type
   INTEGER :: TOP_LIMIT, NOTOPOLOGY
   INTEGER :: NLINES, P
   INTEGER, ALLOCATABLE :: BONDLIST(:,:), ANGLELIST(:,:), TORSIONLIST(:,:), OOPLIST(:,:)

   TOP_LIMIT = 2 * NATOMS !Maximum number of expected bonds
   ALLOCATE(BONDLIST(TOP_LIMIT,2), ANGLELIST(TOP_LIMIT,3), TORSIONLIST(TOP_LIMIT,4), OOPLIST(
        TOP_LIMIT,4))

   !Count bonds and build lists
   NBONDS_PSF = 0
   NTHETA = 0
   NPHI  = 0
   NOOP = 0
   NOTOPOLOGY = 0
   DO I=1,NATOMS
      DO J=1,NBONDS(I)
         IF (JBOND(I,J) > I) THEN
            NBONDS_PSF = NBONDS_PSF + 1
            IF(NBONDS_PSF > TOP_LIMIT) THEN
               NOTOPOLOGY = 1
            ELSE
               BONDLIST(NBONDS_PSF, 1) = I
               BONDLIST(NBONDS_PSF, 2) = JBOND(I,J)
            END IF
         END IF
      END DO

      DO J=1,NIJK(I)
         NTHETA = NTHETA + 1
         IF(I .LE. NATOMS .AND. JANGLEIJK(I,J) .LE. NATOMS .AND. &
              KANGLEIJK(I,J) .LE. NATOMS) THEN !Check the angle isn't a virtual site angle
            IF(NTHETA > TOP_LIMIT) THEN
               NOTOPOLOGY = 1
            ELSE
               ANGLELIST(NTHETA,1) = I
               ANGLELIST(NTHETA,2) = JANGLEIJK(I,J)
               ANGLELIST(NTHETA,3) = KANGLEIJK(I,J)
            END IF
         END IF
      END DO

      DO J=1,NIJKL(I)
         NPHI = NPHI + 1
         IF(NPHI > TOP_LIMIT) THEN
            NOTOPOLOGY = 1
         ELSE
            TORSIONLIST(NPHI,1) = I
            TORSIONLIST(NPHI,2) = JTORIJKL(I,J)
            TORSIONLIST(NPHI,3) = KTORIJKL(I,J)
            TORSIONLIST(NPHI,4) = LTORIJKL(I,J)
```

```
            END IF
        END DO

        DO J=1,NOOPIJKL(I)
            NOOP = NOOP + 1
            IF(NOOP > TOP_LIMIT) THEN
                NOTOPOLOGY = 1
            ELSE
                OOPLIST(NOOP,1) = I
                OOPLIST(NOOP,2) = JOOPIJKL(I,J)
                OOPLIST(NOOP,3) = KOOPIJKL(I,J)
                OOPLIST(NOOP,4) = LOOPIJKL(I,J)
            END IF
        END DO
    END DO

    OPEN(UNIT=117, FILE='s-md.psf')
    WRITE(117,"(A3)") 'PSF'
    WRITE(117,*) ''
    WRITE(117,*) '2 !NTITLE'
    WRITE(117,*) '  REMARKS: PSF file for system :', TITLE
    WRITE(117,*) '  REMARKS: Automatically generated by IBIsCO'
    WRITE(117,*) ''
    WRITE(117,*) ' ',NATOMS,' !NATOM'
    DO I=1,NATOMS
        WRITE(117,1001) I, NAME_MOL(MOL(I)), MOL(I), NAME_MOL(MOL(I)), LABEL(ITYPE(I)), ITYPE(I),
            0.00, MASS0(ITYPE(I)), 0
    END DO
1001 FORMAT (I6, 1X, A7, 1X, I6, 1X, A7, 1X, A7, 1X, I6, 1X, F5.2, 1X, E12.5, 1X, I1)
    WRITE(117,*) ''

    IF( NOTOPOLOGY .EQ. 0) THEN
        !Bonds
        IF( NBONDS_PSF > 0) THEN
            WRITE(117,*) '  ',NBONDS_PSF,' !NBOND'
            NLINES = NBONDS_PSF / 4
            P = 1 !Position within the bond array
            DO I=1,NLINES
                WRITE(117,*) BONDLIST(P,1), BONDLIST(P,2), BONDLIST(P+1,1), BONDLIST(P+1,2), &
                    BONDLIST(P+2,1), BONDLIST(P+2,2), BONDLIST(P+3,1), BONDLIST(P+3,2)
                P = P + 4
            END DO

            SELECT CASE(MOD(NBONDS_PSF,4))
            CASE(1)
                P = NBONDS_PSF
                WRITE(117,*) BONDLIST(P,1), BONDLIST(P,2)
            CASE(2)
                P = NBONDS_PSF - 1
                WRITE(117,*) BONDLIST(P,1), BONDLIST(P,2), BONDLIST(P+1,1), BONDLIST(P+1,2)
            CASE(3)
                P = NBONDS_PSF - 2
                WRITE(117,*) BONDLIST(P,1), BONDLIST(P,2), BONDLIST(P+1,2), BONDLIST(P+1,2), &
                    BONDLIST(P+2,1), BONDLIST(P+2,2)
            END SELECT
            WRITE(117,*) ''
        END IF
        !Angles
        IF( NTHETA > 0) THEN
            WRITE(117,*) '  ',NTHETA,' !NTHETA'
            NLINES = NTHETA / 3
            P = 1
            DO I=1,NLINES
                WRITE(117,*) ANGLELIST(P,1), ANGLELIST(P,2), ANGLELIST(P,3), &
                    ANGLELIST(P+1,1), ANGLELIST(P+1,2), ANGLELIST(P+1,3), &
                    ANGLELIST(P+2,1), ANGLELIST(P+2,2), ANGLELIST(P+2,3)
                P = P + 3
            END DO

            SELECT CASE(MOD(NTHETA,3))
            CASE(1)
                P = NTHETA
                WRITE(117,*) ANGLELIST(P,1), ANGLELIST(P,2), ANGLELIST(P,3)
            CASE(2)
                P = NTHETA - 1
                WRITE(117,*) ANGLELIST(P,1), ANGLELIST(P,2), ANGLELIST(P,3), &
                    ANGLELIST(P+1,1), ANGLELIST(P+1,2), ANGLELIST(P+1,3)
            END SELECT
            WRITE(117,*) ''
        END IF
```

```
      !Torsions
      IF( NPHI > 0) THEN
         WRITE(117,*) '   ',NPHI,' !NPHI'
         NLINES = NPHI / 2
         P = 1
         DO I=1,NLINES
            WRITE(117,*) TORSIONLIST(P,1), TORSIONLIST(P,2), TORSIONLIST(P,3), TORSIONLIST(P,4),
                 &
                  TORSIONLIST(P+1,1), TORSIONLIST(P+1,2), TORSIONLIST(P+1,3), TORSIONLIST(P+1,4)
            P = P + 2
         END DO

         IF(MOD(NPHI,2) .EQ. 1) THEN
            WRITE(117,*) TORSIONLIST(NPHI,1), TORSIONLIST(NPHI,2), TORSIONLIST(NPHI,3),
                 TORSIONLIST(NPHI,4)
         END IF
         WRITE(117,*) ''
      END IF
      !OOPs
      IF( NOOP > 0) THEN
         WRITE(117,*) '   ',NOOP,' !NIMPHI'
         NLINES = NOOP / 2
         P = 1
         DO I=1,NLINES
            WRITE(117,*) OOPLIST(P,1), OOPLIST(P,2), OOPLIST(P,3), OOPLIST(P,4), &
                 OOPLIST(P+1,1), OOPLIST(P+1,2), OOPLIST(P+1,3), OOPLIST(P+1,4)
            P = P + 2
         END DO

         IF(MOD(NOOP,2) .EQ. 1) THEN
            WRITE(117,*) OOPLIST(NOOP,1), OOPLIST(NOOP,2), OOPLIST(NOOP,3), OOPLIST(NOOP,4)
         END IF
         WRITE(117,*) ''
      END IF
   ELSE
      WRITE(*,*) 'Writing topology section of PSF failed'
      WRITE(1,*) 'Writing topology section of PSF failed'
   END IF

   CLOSE(117)

   DEALLOCATE(BONDLIST, ANGLELIST, TORSIONLIST, OOPLIST)

   RETURN
END SUBROUTINE WRITEPSF
```

D.6 Virtual Sites

Listing D.6: 'Makes the pointer lists to beads and atoms'

```
SUBROUTINE MAKE_LISTS()
   USE VAR
   IMPLICIT NONE

   INTEGER :: I, NUMATOM, NUMBEADVS

   NUMATOMS =0
   NUMBEADS =0

   DO I=1,NATOMS
      IF(TYPE_LABEL(I) .eq. 1) THEN !1 is atom
         NUMATOMS = NUMATOMS +1
      ELSE IF(TYPE_LABEL(I) .eq. 2) THEN!type_label = 2 is bead
         NUMBEADS = NUMBEADS +1
      ELSE
         WRITE(1,*) 'ATOM ',I,' HAS INVALID TYPE LABEL, CHECK interaction FILE'
         WRITE(*,*) 'ATOM ',I,' HAS INVALID TYPE LABEL, CHECK interaction FILE'
         ISTOP = 1
         RETURN
      END IF
   END DO

   NCOARSE = NUMBEADS + NVIRTA
   ALLOCATE(ATOM(NUMATOMS), BEAD(NCOARSE))
```

```
    ATOM = 0
    BEAD = 0
28  NUMATOM =0
    NUMBEADVS =0

    !Make atom list
    DO I=1,NATOMS
33      IF(type_label(I) .eq. 1) then !If atom
            NUMATOM = NUMATOM +1
            ATOM(NUMATOM) = I
        ELSE
            NUMBEADVS = NUMBEADVS +1
38          BEAD(NUMBEADVS) = I
        END IF
    END DO

    DO I=1,NVIRTA
43      NUMBEADVS = NUMBEADVS +1
        BEAD(NUMBEADVS) = NATOMS+I
    END DO

    RETURN
48  END SUBROUTINE MAKE_LISTS
```

Listing D.7: 'Move force from VS to atoms'

```
    SUBROUTINE DISTRIBUTE_VSFORCE()
2     USE VAR
      USE OMP_LIB

      IMPLICIT NONE

7     INTEGER :: I, J, TI, TJ, A, VS_POS

      !$OMP PARALLEL DO DEFAULT(NONE) SCHEDULE(STATIC,1) &
      !$OMP& SHARED(NVIRTA,VITYPE,NATOMS,ITYPE)&
      !$OMP& SHARED(VIRT_NUMATOMS,VIRT_ATM_IND,VIRT_MASSCOEFF)&
12    !$OMP& SHARED(FXYZ)&
      !$OMP& PRIVATE(I,TI,VS_POS,A,J,TJ)
      DO I=1,NVIRTA
          TI = VITYPE(I)
          VS_POS = NATOMS + I
17        DO A=1,VIRT_NUMATOMS(TI)
              J = VIRT_ATM_IND(I,A)
              TJ = ITYPE(J)
              FXYZ(:,J) = FXYZ(:,J) + FXYZ(:,VS_POS) * VIRT_MASSCOEFF(TI,TJ)
          END DO
22        !Reset force on virtual site to 0 once distributed
          FXYZ(:,VS_POS) = 0.0
      END DO
      !$OMP END PARALLEL DO

27    RETURN
    END SUBROUTINE DISTRIBUTE_VSFORCE
```

Listing D.8: 'Define the position of VS'

```
    SUBROUTINE VIRTUAL_DEF()
2     USE VAR
      IMPLICIT NONE

      INTEGER :: I, J, K, TI, POS
      REAL*4, DIMENSION(3) :: SUMTOTXYZ
7     REAL*4, DIMENSION(3) :: SPXYZ

      !Calculate centres of virtual sites
      DO I=1,NVIRTA
          POS = I + NATOMS
12        !Assign type_labels to virtual sites
          TYPE_LABEL(POS) = 3

          IF (VIRT_CENTER(I) .NE. 0)THEN !If using a functional site
              J = VIRT_CENTER(I)
17
              RXYZ(:,POS) = SXYZ(:,J)
              SXYZ(:,POS) = SXYZ(:,J)
```

```
      ELSE !Else using a COM
         TI = VITYPE(I)
         SUMTOTXYZ(1) = 0.0
         !Finds COM
         DO J=1,VIRT_NUMATOMS(TI)
            K = VIRT_ATM_IND(I,J)
            SUMTOTXYZ(:) = SUMTOTXYZ(:) + MASS(ITYPE(K)) * SXYZ(:,K)
         END DO

         RXYZ(:,POS) = SUMTOTXYZ(:) * VIRT_INVMASS(TI)
         SXYZ(:,POS) = RXYZ(:,POS)
      END IF

      IF (ANY((SXYZ(:,POS) > BOX2(:)) .OR. (SXYZ(:,POS) < -BOX2(:)))) THEN
         RXYZ(:,POS) = RXYZ(:,POS) / BOX2(:)
         SPXYZ(:) = RXYZ(:,POS) - 2.0 * INT(RXYZ(:,POS) / 2.0)
         RXYZ(:,POS) = (SPXYZ(:) - 2.0 * INT(SPXYZ(:))) * BOX2(:)
      END IF
   END DO

   RETURN
END SUBROUTINE VIRTUAL_DEF
```

D.7 MTS Subroutines

The MTS operation of the program had a separate MD loop. These algorithms are explained in Chapter 7

src/MTS_LOOP.f90

```
!> @file
!> @brief The multiple time step (MTS) subroutines
!> @details MTS allows hybrid scale simulations to be ran faster.  Normally, hybrid scale simulations
!!          are ran using an atomistic time step, which is around an order of magnitude smaller than
!!          the native coarse grained counterpart.  Using MTS allows the nonbonded coarse grained
!!          force to be approximated in some time steps, reducing the computational load.
!!
!!          For further details, see \cite MTS
!!
!> @author Richard J Gowers

!> @brief The alternate molecular dynamics loop for MTS simulations
SUBROUTINE MTS_LOOP
   ! Could merge this with main loop, and have if statement around MTS_FORCE call to differentiate?
   USE VAR
   USE MTS

   IMPLICIT NONE

   ALLOCATE(MTS_FXYZ(3, 3, NCOARSE))
   ALLOCATE(MTS_PT11(3), MTS_PT22(3), MTS_PT33(3), MTS_PT12(3), MTS_PT13(3), MTS_PT23(3))
   ALLOCATE(MTS_V_NB(3, 3))

   DO STEP = 1, NSTEP
      CALL SHIFT()
      CALL VIRTUAL_DEF()

      IF(MOD(STEP, NUPDATE) .eq. 0) THEN
         CALL UPDATE_NEIGHBOURLIST()
      END IF

      CALL MTS_FORCE(STEP)
      CALL MOVE()

      IF(MOD(STEP, HALT_DRIFT) .eq. 0) THEN
         CALL MOMENTUM()
      END IF

      IF (ENSEMBLE .eq. 2) THEN
```

```
            CALL SCALEBP(STEP)
        END IF

        CALL AVERAGE(STEP)

        IF (MOD(STEP, NTRJ) .eq. 0) THEN
            CALL WRITETRJ(STEP)
            CALL OUTPUT(STEP)
        END IF
    END DO

    DEALLOCATE(MTS_FXYZ)
    DEALLOCATE(MTS_PT11, MTS_PT22, MTS_PT33, MTS_PT12, MTS_PT13, MTS_PT23)
    DEALLOCATE(MTS_V_NB)

    RETURN
END SUBROUTINE MTS_LOOP

!> @brief The force loop for MTS simulation
!> @details In this force step, the current step number is used to determine how the coarse grained
!!          nonbonded forces are calculated. The first three are done explicitly, then NMTS (an input
!!          setting) steps are done using an approximation.
!!
!!          This approximation is a Taylor Expansion
!!
!!          The coarse grained contributions to nonbonded energy (V_NB) and pressure tensors (PTxx)
!!          are also estimated in the same fashion.
SUBROUTINE MTS_FORCE(STEPNO)
    USE VAR
    USE MTS

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: STEPNO !< The current step number
    INTEGER :: I, MTS_MOD

    ! Reset force
    FXYZ = 0.0
    FXYZNB = 0.0

    PT11 = 0.0
    PT22 = 0.0
    PT33 = 0.0
    PT12 = 0.0
    PT13 = 0.0
    PT23 = 0.0

    !Reset potential energy measures
    V_NB = 0.0
    V_BOND = 0.0
    V_ANGLE = 0.0
    V_TORSION = 0.0
    V_OOP = 0.0

    MTS_MOD = MOD(STEPNO, (3 + NMTS))
    SELECT CASE(MTS_MOD)
        CASE(1, 2, 3) ! Explicit steps
            CALL NONBONDED_FORCE(NCOARSE, BEAD, MAXNAB_BEAD, LIST_BEAD, RCUT_BEAD, RCUTSQ_BEAD)
            CALL MTS_SAVEFORCE(MTS_MOD, BEAD, FXYZ, NITEMS, MTS_FXYZ, NCOARSE)
            CALL MTS_SAVEAUXS(MTS_MOD)

        CASE DEFAULT ! Use approximation
            CALL MTS_APPROX(MTS_MOD, BEAD, FXYZ, NITEMS, MTS_FXYZ, NCOARSE)
            CALL MTS_LOADAUXS(MTS_MOD)
    END SELECT
    CALL DISTRIBUTE_VSFORCE()

    ! Call atoms second, so that V_NB and PTxx only have CG contributions
    ! Order shouldnt matter anyway...
    CALL NONBONDED_FORCE(NUMATOMS, ATOM, MAXNAB_ATOM, LIST_ATOM, RCUT_ATOM, RCUTSQ_ATOM)

    DO I = 1, NATOMS
        FXYZNB(:,I) = FXYZ(:,I)
    END DO

    CALL BONDED_FORCE()
```

```
   CALL DISTRIBUTE_VSFORCE()

   DO I=1,NATOMS
      PT11 = PT11 + (FXYZ(1,I) - FXYZNB(1,I))*SXYZ(1,I)
      PT22 = PT22 + (FXYZ(2,I) - FXYZNB(2,I))*SXYZ(2,I)
      PT33 = PT33 + (FXYZ(3,I) - FXYZNB(3,I))*SXYZ(3,I)
      PT12 = PT12 + (FXYZ(2,I) - FXYZNB(2,I))*SXYZ(1,I)
      PT13 = PT13 + (FXYZ(3,I) - FXYZNB(3,I))*SXYZ(1,I)
      PT23 = PT23 + (FXYZ(3,I) - FXYZNB(3,I))*SXYZ(2,I)
   END DO

   RETURN
END SUBROUTINE MTS_FORCE

!> @brief Records the coarse grained force from an explicit MTS step.
SUBROUTINE MTS_SAVEFORCE(I, BEAD, FXYZ, NITEMS, MTS_FXYZ, NCOARSE)
   IMPLICIT NONE

   INTEGER, INTENT(IN) :: I !< Mod of current MTS step
   INTEGER, INTENT(IN) :: NITEMS, NCOARSE !< Total number of particles (real and virtual)
   REAL*4, DIMENSION(3, NITEMS), INTENT(IN) :: FXYZ
   REAL*4, DIMENSION(3, 3, NCOARSE), INTENT(INOUT) :: MTS_FXYZ
   INTEGER, DIMENSION(NCOARSE), INTENT(IN) :: BEAD
   INTEGER :: A, ATOM_ID

   DO A = 1, NCOARSE
      ATOM_ID = BEAD(A)
      MTS_FXYZ(:,I,A) = FXYZ(:,ATOM_ID)
   END DO

   RETURN
END SUBROUTINE MTS_SAVEFORCE

!> @brief Generates approximated coarse grained forces for MTS steps.
SUBROUTINE MTS_APPROX(I, BEAD, FXYZ, NITEMS, MTS_FXYZ, NCOARSE)
   IMPLICIT NONE

   INTEGER, INTENT(IN) :: I ! Mod of current MTS step
   INTEGER, INTENT(IN) :: NITEMS, NCOARSE !< Total number of particles (real and virtual)
   REAL*4, DIMENSION(3, NITEMS), INTENT(INOUT) :: FXYZ
   REAL*4, DIMENSION(3, 3, NCOARSE), INTENT(IN) :: MTS_FXYZ
   INTEGER, DIMENSION(NCOARSE), INTENT(IN) :: BEAD
   INTEGER :: A, ATOM_ID

   DO A = 1, NCOARSE
      ATOM_ID = BEAD(A)
      ! Forwards approx
      FXYZ(:,ATOM_ID) = MTS_FXYZ(:,3,A) + (I-3) * 0.5 * (MTS_FXYZ(:,3,A) - MTS_FXYZ(:,1,A))
   END DO

   RETURN
END SUBROUTINE MTS_APPROX

!> @brief Records the nonbonded potential and pressure tensors from explicit MTS steps.
SUBROUTINE MTS_SAVEAUXS(I)
   USE MTS
   USE VAR
   IMPLICIT NONE

   INTEGER, INTENT(IN) :: I !< The current mod of the MTS step (1, 2, or 3 here)

   MTS_V_NB(I,:) = V_NB(:)

   MTS_PT11(I) = PT11
   MTS_PT22(I) = PT22
   MTS_PT33(I) = PT33
   MTS_PT12(I) = PT12
   MTS_PT13(I) = PT13

   RETURN
END SUBROUTINE MTS_SAVEAUXS

!> @brief Approximates the nonbonded potential and pressure tensors in approximate MTS steps.
SUBROUTINE MTS_LOADAUXS(I)
   USE MTS
   USE VAR
   IMPLICIT NONE

   INTEGER, INTENT(IN) :: I
```

```
    V_NB(:) = MTS_V_NB(3,:) + (I-3) * 0.5 * (MTS_V_NB(3,:) - MTS_V_NB(1,:))

199 PT11 = MTS_PT11(3) + (I-3) * 0.5 * (MTS_PT11(3) - MTS_PT11(1))
    PT22 = MTS_PT22(3) + (I-3) * 0.5 * (MTS_PT22(3) - MTS_PT22(1))
    PT33 = MTS_PT33(3) + (I-3) * 0.5 * (MTS_PT33(3) - MTS_PT33(1))
    PT12 = MTS_PT12(3) + (I-3) * 0.5 * (MTS_PT12(3) - MTS_PT12(1))
    PT13 = MTS_PT13(3) + (I-3) * 0.5 * (MTS_PT13(3) - MTS_PT13(1))
204 PT23 = MTS_PT23(3) + (I-3) * 0.5 * (MTS_PT23(3) - MTS_PT23(1))

    RETURN
END SUBROUTINE MTS_LOADAUXS
```

Bibliography

[1] H. A. Karimi-Varzaneh, H.-J. Qian, X. Chen, P. Carbone, and F. Müller-Plathe, "IBIsCO: A molecular dynamics simulation package for coarse-grained simulation," *Journal of Computational Chemistry*, vol. 32, no. 7, pp. 1475–1487, 2011.

Appendix E

Analysis Tools

Over the course of this work, it was necessary to create various bespoke analysis tools. These were included as part of the MDAnalysis library[1] and made freely available under a GPL v3 license[2].

These were written in the Python as the performance of these wasn't critical, and so they could be written in a higher level programming language for ease of prototypinc and debugging. For good introductions to the Python programming languages, refer to References [3] and [4] For completeness, they are included here.

E.1 Persistence Length

src/polymer.py

```
# -*- Mode: python; tab-width: 4; indent-tabs-mode:nil; coding:utf-8 -*-
# vim: tabstop=4 expandtab shiftwidth=4 softtabstop=4
#
# MDAnalysis --- http://www.MDAnalysis.org
# Copyright (c) 2006-2015 Naveen Michaud-Agrawal, Elizabeth J. Denning, Oliver Beckstein
# and contributors (see AUTHORS for the full list)
#
# Released under the GNU Public Licence, v2 or any higher version
#
# Please cite your use of MDAnalysis in published work:
#
# N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein.
# MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations.
# J. Comput. Chem. 32 (2011), 2319--2327, doi:10.1002/jcc.21787
#

"""
Polymer analysis --- :mod:`MDAnalysis.analysis.polymer`
=======================================================


:Author: Richard J. Gowers
:Year: 2015
:Copyright: GNU Public License v3

This module contains various commonly used tools in analysing polymers.

"""
from six.moves import range

import numpy as np
import logging

```

```
from .. import NoDataError
from ..lib.util import blocks_of
from ..lib.distances import calc_bonds
from .base import AnalysisBase

logger = logging.getLogger(__name__)


class PersistenceLength(AnalysisBase):
    r"""Calculate the persistence length for polymer chains

    The persistence length is the length at which two points on the polymer
    chain become decorrelated.

    Notes
    -----
    This analysis requires that the trajectory supports indexing

    .. versionadded:: 0.13.0
    """
    def __init__(self, atomgroups,
                 start=None, stop=None, step=None):
        """Calculate the persistence length for polymer chains

        Parameters
        ----------
        atomgroups : list
            List of atomgroups.  Each atomgroup should represent a single
            polymer chain, ordered in the correct order.
        start : int, optional
            First frame of trajectory to analyse, Default: 0
        stop : int, optional
            Last frame of trajectory to analyse, Default: -1
        step : int, optional
            Step between frames to analyse, Default: 1
        """
        self._atomgroups = atomgroups

        # Check that all chains are the same length
        lens = [len(ag) for ag in atomgroups]
        chainlength = len(atomgroups[0])
        if not all( l == chainlength for l in lens):
            raise ValueError("Not all AtomGroups were the same size")

        self._setup_frames(atomgroups[0].universe.trajectory,
                           start, stop, step)

        self._results = np.zeros(chainlength - 1, dtype=np.float32)

    def _single_frame(self):
        # could optimise this by writing a "self dot array"
        # we're only using the upper triangle of np.inner
        # function would accept a bunch of coordinates and spit out the
        # decorrel for that
        n = len(self._atomgroups[0])

        for chain in self._atomgroups:
            # Vector from each atom to next
            vecs = chain.positions[1:] - chain.positions[:-1]
            # Normalised to unit vectors
            vecs /= np.sqrt((vecs * vecs).sum(axis=1))[:, None]

            inner_pr = np.inner(vecs, vecs)
            for i in range(n-1):
                self._results[:(n-1)-i] += inner_pr[i, i:]

    def _conclude(self):
        n = len(self._atomgroups[0])

        norm = np.linspace(n - 1, 1, n - 1)
        norm *= len(self._atomgroups) * self.nframes

        self.results = self._results / norm
        self._calc_bond_length()

    def _calc_bond_length(self):
        """calculate average bond length"""
        bs = []
        for ag in self._atomgroups:
            pos = ag.positions
```

```
            b = calc_bonds(pos[:-1], pos[1:]).mean()
            bs.append(b)
        self.lb = np.mean(bs)

    def perform_fit(self):
        """Fit the results to an exponential decay"""
        from scipy.optimize import curve_fit

        try:
            results = self.results
        except AttributeError:
            raise NoDataError("Use the run method first")
        self.x = np.arange(len(self.results)) * self.lb

        self.lp = fit_exponential_decay(self.x, self.results)

        self.fit = np.exp(-self.x/self.lp)

    def plot(self):
        """Oooh fancy"""
        import matplotlib.pyplot as plt
        plt.ylabel('C(x)')
        plt.xlabel('x')
        plt.xlim([0.0, 40 * self.lb])
        plt.plot(self.x, self.results, 'ro')
        plt.plot(self.x, self.fit)
        plt.show()


def fit_exponential_decay(x, y):
    r"""Fit a function to an exponential decay

    .. math::  y = \exp(-x/a)

    Parameters
    ----------
    x, y : array_like
      The two arrays of data

    Returns
    -------
    a : float
      The coefficient *a* for this decay

    Notes
    -----
    This function assumes that data starts at 1.0 and decays to 0.0

    Requires scipy
    """
    from scipy.optimize import curve_fit

    def expfunc(x, a):
        return np.exp(-x/a)

    a = curve_fit(expfunc, x, y)[0][0]

    return a
```

E.2 Radial distribution function

src/rdf.py

```
# -*- Mode: python; tab-width: 4; indent-tabs-mode:nil; coding:utf-8 -*-
# vim: tabstop=4 expandtab shiftwidth=4 softtabstop=4
#
# MDAnalysis --- http://www.MDAnalysis.org
# Copyright (c) 2006-2015 Naveen Michaud-Agrawal, Elizabeth J. Denning, Oliver Beckstein
# and contributors (see AUTHORS for the full list)
#
# Released under the GNU Public Licence, v2 or any higher version
#
# Please cite your use of MDAnalysis in published work:
#
```

```
# N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein.
# MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations.
# J. Comput. Chem. 32 (2011), 2319--2327, doi:10.1002/jcc.21787
#

"""
Radial Distribution Functions --- :mod:`MDAnalysis.analysis.rdf`
================================================================

Tools for calculating pair distribution functions

# TODO
 - Structure factor?
 - Coordination number
"""
import numpy as np

from ..lib.util import blocks_of
from ..lib import distances
from .base import AnalysisBase


class InterRDF(AnalysisBase):
    """Intermolecular pair distribution function

    InterRDF(g1, g2, nbins=75, range=(0.0, 15.0))

    Arguments
    ---------
    g1
      First AtomGroup
    g2
      Second AtomGroup

    Keywords
    --------
    nbins
          Number of bins in the histogram [75]
    range
          The size of the RDF [0.0, 15.0]
    exclusion_block
          A tuple representing the tile to exclude from the distance
          array. [None]
    start
          The frame to start at [0]
    stop
          The frame to end at [-1]
    step
          The step size through the trajectory in frames [0]

    Example
    -------
    First create the InterRDF object, by supplying two AtomGroups
    then use the `run` method

      rdf = InterRDF(ag1, ag2)
      rdf.run()

    Results are available through the .bins and .rdf attributes

      plt.plot(rdf.bins, rdf.rdf)

    The `exclusion_block` keyword allows the masking of pairs from
    within the same molecule.  For example, if there are 7 of each
    atom in each molecule, the exclusion mask (7, 7) can be used.

    .. versionadded:: 0.13.0
    """
    def __init__(self, g1, g2,
                 nbins=75, range=(0.0, 15.0), exclusion_block=None,
                 start=None, stop=None, step=None):
        self.g1 = g1
        self.g2 = g2
        self.u = g1.universe

        self._setup_frames(self.u.trajectory,
                           start=start,
                           stop=stop,
                           step=step)
```

```
        self.rdf_settings = {'bins':nbins,
                             'range':range}

        # Empty histogram to store the RDF
        count, edges = np.histogram([-1], **self.rdf_settings)
        count = count.astype(np.float64)
        count *= 0.0
        self.count = count
        self.edges = edges
        self.bins = 0.5 * (edges[:-1] + edges[1:])

        # Need to know average volume
        self.volume = 0.0

        # Allocate a results array which we will reuse
        self._result = np.zeros((len(self.g1), len(self.g2)), dtype=np.float64)
        # If provided exclusions, create a mask of _result which
        # lets us take these out
        if exclusion_block is not None:
            self._exclusion_block = exclusion_block
            self._exclusion_mask = blocks_of(self._result, *exclusion_block)
            self._maxrange = range[1] + 1.0
        else:
            self._exclusion_block = None
            self._exclusion_mask = None

    def _single_frame(self):
        distances.distance_array(self.g1.positions, self.g2.positions,
                                 box=self.u.dimensions, result=self._result)
        # Maybe exclude same molecule distances
        if self._exclusion_mask is not None:
            self._exclusion_mask[:] = self._maxrange

        count = np.histogram(self._result, **self.rdf_settings)[0]
        self.count += count

        self.volume += self._ts.volume

    def _conclude(self):
        # Number of each selection
        nA = len(self.g1)
        nB = len(self.g2)
        N = nA * nB

        # If we had exclusions, take these into account
        if self._exclusion_block:
            xA, xB = self._exclusion_block
            nblocks = nA / xA
            N -= xA * xB * nblocks

        # Volume in each radial shell
        vol = np.power(self.edges[1:], 3) - np.power(self.edges[:-1], 3)
        vol *= 4/3.0 * np.pi

        # Average number density
        box_vol = self.volume / self.nframes
        density = N / box_vol

        rdf = self.count / (density * vol * self.nframes)

        self.rdf = rdf
```

E.3 Hydrogen bond lifetime

This was used in Chapters 5 and 9 to quantify the dynamics at short scales through the lifetime of hydrogen bonds.

src/hbond_autocorrel.py

```
# -*- Mode: python; tab-width: 4; indent-tabs-mode:nil; coding:utf-8 -*-
# vim: tabstop=4 expandtab shiftwidth=4 softtabstop=4
#
# MDAnalysis --- http://www.MDAnalysis.org
```

```
# Copyright (c) 2006-2015 Naveen Michaud-Agrawal, Elizabeth J. Denning, Oliver Beckstein
# and contributors (see AUTHORS for the full list)
#
# Released under the GNU Public Licence, v2 or any higher version
#
# Please cite your use of MDAnalysis in published work:
#
# N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein.
# MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations.
# J. Comput. Chem. 32 (2011), 2319--2327, doi:10.1002/jcc.21787
#
"""
Hydrogen bond autocorrelation --- :mod:`MDAnalysis.analysis.hbonds.hbond_autocorrel`
==========================================================================

:Author: Richard J. Gowers
:Year: 2014
:Copyright: GNU Public License v3

.. versionadded:: 0.9.0

Description
-----------

Calculates the time autocorrelation function, :math:`C_x(t)`, for the hydrogen
bonds in the selections passed to it.  The population of hydrogen bonds at a
given startpoint, :math:`t_0`, is evaluated based on geometric criteria and
then the lifetime of these bonds is monitored over time.  Multiple passes
through the trajectory are used to build an average of the behaviour.

    :math:`C_x(t) = \\left \\langle \\frac{h_{ij}(t_0) h_{ij}(t_0 + t)}{h_{ij}(t_0)^2} \\right\\
        rangle`

The subscript :math:`x` refers to the definition of lifetime being used, either
continuous or intermittent.  The continuous definition measures the time that
a particular hydrogen bond remains continuously attached, whilst the
intermittent definition allows a bond to break and then subsequently reform and
be counted again.  The relevent lifetime, :math:`\\tau_x`, can then be found
via integration of this function

    :math:`\\tau_x = \\int_0^\\infty C_x(t) dt`

For this, the observed behaviour is fitted to a multi exponential function,
using 2 exponents for the continuous lifetime and 3 for the intermittent
lifetime.

    :math:`C_x(t) = A_1 \\exp( - t / \\tau_1)
    + A_2 \\exp( - t / \\tau_2)
    [+ A_3 \\exp( - t / \\tau_3)]`

Where the final pre expoential factor :math:`A_n` is subject to the condition:

    :math:`A_n = 1 - \\sum\\limits_{i=1}^{n-1} A_i`

For further details see [Gowers2015]_.

.. rubric:: References

.. [Gowers2015]  Richard J. Gowers and Paola Carbone,
                 A multiscale approach to model hydrogen bonding: The case of polyamide
                 The Journal of Chemical Physics, 142, 224907 (2015),
                 DOI:http://dx.doi.org/10.1063/1.4922445

Input
-----

Three AtomGroup selections representing the **hydrogens**, **donors** and
**acceptors** that you wish to analyse.  Note that the **hydrogens** and
**donors** selections must be aligned, that is **hydrogens[0]** and
**donors[0]** must represent a bonded pair.  If a single donor therefore has
two hydrogens, it must feature twice in the **donors** AtomGroup.

The keyword **exclusions** allows a tuple of array addresses to be provided,
(Hidx, Aidx),these pairs of hydrogen-acceptor are then not permitted to be
counted as part of the analysis.  This could be used to exclude the
consideration of hydrogen bonds within the same functional group, or to perform
analysis on strictly intermolecular hydrogen bonding.

Hydrogen bonds are defined on the basis of geometric criteria; a
Hydrogen-Acceptor distance of less then **dist_crit** and a
```

```
Donor-Hydrogen-Acceptor angle of greater than **angle_crit**.

The length of trajectory to analyse in ps, **sample_time**, is used to choose
what length to analyse.

Multiple passes, controlled by the keyword **nruns**, through the trajectory
are performed and an average calculated.  For each pass, **nsamples** number
of points along the run are calculated.


Output
------

All results of the analysis are available through the *solution* attribute.
This is a dictionary with the following keys

- *results*  The raw results of the time autocorrelation function.
- *time*     Time axis, in ps, for the results.
- *fit*      Results of the exponential curve fitting procedure. For the
             *continuous* lifetime these are (A1, tau1, tau2), for the
             *intermittent* lifetime these are (A1, A2, tau1, tau2, tau3).
- *tau*      Calculated time constant from the fit.
- *estimate* Estimated values generated by the calculated fit.

The *results* and *time* values are only filled after the :meth:`run` method,
*fit*, *tau* and *estimate* are filled after the :meth:`solve` method has been
used.


Examples
--------

::

  from MDAnalysis.analysis import hbonds
  import matplotlib.pyplot as plt
  H = u.select_atoms('name Hn')
  O = u.select_atoms('name O')
  N = u.select_atoms('name N')
  hb_ac = hbonds.HydrogenBondAutoCorrel(u, acceptors = u.atoms.O,
              hydrogens = u.atoms.Hn, donors = u.atoms.N,bond_type='continuous',
              sample_time = 2, nruns = 20, nsamples = 1000)
  hb_ac.run()
  hb_ac.solve()
  tau = hb_ac.solution['tau']
  time = hb_ac.solution['time']
  results = hb_ac.solution['results']
  estimate = hb_ac.solution['estimate']
  plt.plot(time, results, 'ro')
  plt.plot(time, estimate)
  plt.show()


.. autoclass:: HydrogenBondAutoCorrel

   .. automethod:: run

   .. automethod:: solve

   .. automethod:: save_results


"""
from six.moves import zip
import numpy as np
import warnings

from MDAnalysis.lib.log import ProgressMeter
from MDAnalysis.lib.distances import distance_array, calc_angles, calc_bonds


class HydrogenBondAutoCorrel(object):
    """Perform a time autocorrelation of the hydrogen bonds in the system.

    Parameters
    ----------
    universe : Universe
        MDAnalysis Universe that all selections belong to
    hydrogens : AtomGroup
        AtomGroup of Hydrogens which can form hydrogen bonds
```

```
    acceptors : AtomGroup
        AtomGroup of all Acceptor atoms
    donors : AtomGroup
        The atoms which are connected to the hydrogens.  This group
168     must be identical in length to the hydrogen group and matched,
        ie hydrogens[0] is bonded to donors[0].
        For many cases, this will mean a donor appears twice in this
        group.
    bond_type : str
173     Which definition of hydrogen bond lifetime to consider, either
        'continuous' or 'intermittent'.
    exclusions : ndarray, optional
        Indices of Hydrogen-Acceptor pairs to be excluded.
        With nH and nA Hydrogens and Acceptors, a (nH x nA) array of distances
178     is calculated, *exclusions* is used as a mask on this array to exclude
        some pairs.
    angle_crit : float, optional
        The angle (in degrees) which all bonds must be greater than [130.0]
    dist_crit : float, optional
183     The maximum distance (in Angstroms) for a hydrogen bond [3.0]
    sample_time : float, optional
        The amount of time, in ps, that you wish to observe hydrogen
        bonds for [100]
    nruns : int, optional
188     The number of different start points within the trajectory
        to use [1]
    nsamples : int, optional
        Within each run, the number of frames to analyse [50]
    pbc : bool, optional
193     Whether to consider periodic boundaries in calculations [``True``]
    """

    def __init__(self, universe,
                 hydrogens=None, acceptors=None, donors=None,
198              bond_type=None,
                 exclusions=None,
                 angle_crit=130.0, dist_crit=3.0,  # geometric criteria
                 sample_time=100,  # expected length of the decay in ps
                 time_cut=None,  # cutoff time for intermittent hbonds
203              nruns=1,  # number of times to iterate through the trajectory
                 nsamples=50,  # number of different points to sample in a run
                 pbc=True):
        self.u = universe
        # check that slicing is possible
208     try:
            self.u.trajectory[0]
        except:
            raise ValueError("Trajectory must support slicing")

213     self.h = hydrogens
        self.a = acceptors
        self.d = donors
        if not len(self.h) == len(self.d):
            raise ValueError("Donors and Hydrogen groups must be matched")
218
        self.exclusions = exclusions
        if self.exclusions:
            if not len(self.exclusions[0]) == len(self.exclusions[1]):
                raise ValueError(
223                 "'exclusion' must be two arrays of identical length")

        self.bond_type = bond_type
        if self.bond_type not in ['continuous', 'intermittent']:
            raise ValueError(
228             "bond_type must be either 'continuous' or 'intermittent'")

        self.a_crit = np.deg2rad(angle_crit)
        self.d_crit = dist_crit
        self.pbc = pbc
233     self.sample_time = sample_time
        self.nruns = nruns
        self.nsamples = nsamples
        self._slice_traj(sample_time)
        self.time_cut = time_cut
238
        self.solution = {
            'results': None,  # Raw results
            'time': None,  # Time axis of raw results
            'fit': None,  # coefficients for fit
243         'tau': None,  # integral of exponential fit
```

```
            'estimate': None  # y values of fit against time
        }

    def _slice_traj(self, sample_time):
        """Set up start and end points in the trajectory for the
        different passes
        """
        dt = self.u.trajectory.dt  # frame step size in time
        req_frames = int(sample_time / dt)  # the number of frames required

        n_frames = len(self.u.trajectory)
        if req_frames > n_frames:
            warnings.warn("Number of required frames ({}) greater than the"
                          " number of frames in trajectory ({})"
                          .format(req_frames, n_frames), RuntimeWarning)

        numruns = self.nruns
        if numruns > n_frames:
            numruns = n_frames
            warnings.warn("Number of runs ({}) greater than the number of"
                          " frames in trajectory ({})"
                          .format(self.nruns, n_frames), RuntimeWarning)

        self._starts = np.arange(0, n_frames, n_frames / numruns, dtype=int)
        # limit stop points using clip
        self._stops = np.clip(self._starts + req_frames, 0, n_frames)

        self._skip = req_frames / self.nsamples
        if self._skip == 0:  # If nsamples > req_frames
            warnings.warn("Desired number of sample points too high, using {0}"
                          .format(req_frames), RuntimeWarning)
            self._skip = 1

    def run(self, force=False):
        """Run all the required passes

        Parameters:
        ----------
        force : bool, optional
            Will overwrite previous results if they exist
        """
        # if results exist, don't waste any time
        if self.solution['results'] is not None and not force:
            return

        master_results = np.zeros_like(np.arange(self._starts[0],
                                                 self._stops[0],
                                                 self._skip),
                                       dtype=np.float32)
        # for normalising later
        counter = np.zeros_like(master_results, dtype=np.float32)

        pm = ProgressMeter(self.nruns, interval=1,
                           format="Performing run %(step)5d/%(numsteps)d"
                                  "[%(percentage)5.1f%%]\r")

        for i, (start, stop) in enumerate(zip(self._starts, self._stops)):
            pm.echo(i + 1)

            # needed else trj seek thinks a np.int64 isn't an int?
            results = self._single_run(int(start), int(stop))

            nresults = len(results)
            if nresults == len(master_results):
                master_results += results
                counter += 1.0
            else:
                master_results[:nresults] += results
                counter[:nresults] += 1.0

        master_results /= counter

        self.solution['time'] = np.arange(
            len(master_results),
            dtype=np.float32) * self.u.trajectory.dt * self._skip
        self.solution['results'] = master_results

    def _single_run(self, start, stop):
        """Perform a single pass of the trajectory"""
        self.u.trajectory[start]
```

```
        # Calculate partners at t=0
        box = self.u.dimensions if self.pbc else None

        # 2d array of all distances
        d = distance_array(self.h.positions, self.a.positions, box=box)
        if self.exclusions:
            # set to above dist crit to exclude
            d[self.exclusions] = self.d_crit + 1.0

        # find which partners satisfy distance criteria
        hidx, aidx = np.where(d < self.d_crit)

        a = calc_angles(self.d.positions[hidx], self.h.positions[hidx],
                        self.a.positions[aidx], box=box)
        # from amongst those, who also satisfiess angle crit
        idx2 = np.where(a > self.a_crit)
        hidx = hidx[idx2]
        aidx = aidx[idx2]

        nbonds = len(hidx)  # number of hbonds at t=0
        results = np.zeros_like(np.arange(start, stop, self._skip),
                                dtype=np.float32)

        if self.time_cut:
            # counter for time criteria
            count = np.zeros(nbonds, dtype=np.float64)

        for i, ts in enumerate(self.u.trajectory[start:stop:self._skip]):
            box = self.u.dimensions if self.pbc else None

            d = calc_bonds(self.h.positions[hidx], self.a.positions[aidx],
                           box=box)
            a = calc_angles(self.d.positions[hidx], self.h.positions[hidx],
                            self.a.positions[aidx], box=box)

            winners = (d < self.d_crit) & (a > self.a_crit)
            results[i] = winners.sum()

            if self.bond_type is 'continuous':
                # Remove losers for continuous definition
                hidx = hidx[np.where(winners)]
                aidx = aidx[np.where(winners)]
            elif self.bond_type is 'intermittent':
                if self.time_cut:
                    # Add to counter of where losers are
                    count[~ winners] += self._skip * self.u.trajectory.dt
                    count[winners] = 0  # Reset timer for winners

                    # Remove if you've lost too many times
                    # New arrays contain everything but removals
                    hidx = hidx[count < self.time_cut]
                    aidx = aidx[count < self.time_cut]
                    count = count[count < self.time_cut]
                else:
                    pass

            if len(hidx) == 0:  # Once everyone has lost, the fun stops
                break

        results /= nbonds

        return results

    def save_results(self, filename='hbond_autocorrel'):
        """Saves the results to a numpy zipped array (.npz, see np.savez)

        This can be loaded using np.load(filename)

        Parameters
        ----------
        filename : str, optional
            The desired filename [hbond_autocorrel]
        """
        if self.solution['results'] is not None:
            np.savez(filename, time=self.solution['time'],
                     results=self.solution['results'])
        else:
            raise ValueError(
                "Results have not been generated, use the run method first")
```

```
    def solve(self, p_guess=None):
        """Fit results to an multi exponential decay and integrate to find
        characteristic time

        Parameters
        ----------
        p_guess : tuple of floats, optional
            Initial guess for the leastsq fit, must match the shape of the
            expected coefficients

        Continuous defition results are fitted to a double exponential,
        intermittent definition are fit to a triple exponential.

        The results of this fitting procedure are saved into the *fit*,
        *tau* and *estimate* keywords in the solution dict.

         - *fit* contains the coefficients, (A1, tau1, tau2) or
           (A1, A2, tau1, tau2, tau3)
         - *tau* contains the calculated lifetime in ps for the hydrogen
           bonding
         - *estimate* contains the estimate provided by the fit of the time
           autocorrelation function

        In addition, the output of the leastsq function is saved into the
        solution dict

         - *infodict*
         - *mesg*
         - *ier*
        """
        from scipy.optimize import leastsq

        if self.solution['results'] is None:
            raise ValueError(
                "Results have not been generated use, the run method first")

        # Prevents an odd bug with leastsq where it expects
        # double precision data sometimes...
        time = self.solution['time'].astype(np.float64)
        results = self.solution['results'].astype(np.float64)

        def within_bounds(p):
            """Returns True/False if boundary conditions are met or not.
            Uses length of p to detect whether it's handling continuous /
            intermittent

            Boundary conditions are:
             0 < A_x < 1
             sum(A_x) < 1
             0 < tau_x
            """
            if len(p) == 3:
                A1, tau1, tau2 = p
                return (A1 > 0.0) & (A1 < 1.0) & \
                       (tau1 > 0.0) & (tau2 > 0.0)
            elif len(p) == 5:
                A1, A2, tau1, tau2, tau3 = p
                return (A1 > 0.0) & (A1 < 1.0) & (A2 > 0.0) & \
                       (A2 < 1.0) & ((A1 + A2) < 1.0) & \
                       (tau1 > 0.0) & (tau2 > 0.0) & (tau3 > 0.0)

        def err(p, x, y):
            """Custom residual function, returns real residual if all
            boundaries are met, else returns a large number to trick the
            leastsq algorithm
            """
            if within_bounds(p):
                return y - self._my_solve(x, *p)
            else:
                return 100000

        def double(x, A1, tau1, tau2):
            """ Sum of two exponential functions """
            A2 = 1 - A1
            return A1 * np.exp(-x / tau1) + A2 * np.exp(-x / tau2)

        def triple(x, A1, A2, tau1, tau2, tau3):
            """ Sum of three exponential functions """
            A3 = 1 - (A1 + A2)
```

```
            return A1 * np.exp(-x / tau1) + A2 * np.exp(-x / tau2) + A3 * np.exp(-x / tau3)

        if self.bond_type is 'continuous':
            self._my_solve = double

            if p_guess is None:
                p_guess = (0.5, 10 * self.sample_time, self.sample_time)

            p, cov, infodict, mesg, ier = leastsq(err, p_guess,
                                                  args=(time, results),
                                                  full_output=True)
            self.solution['fit'] = p
            A1, tau1, tau2 = p
            A2 = 1 - A1
            self.solution['tau'] = A1 * tau1 + A2 * tau2
        else:
            self._my_solve = triple

            if p_guess is None:
                p_guess = (0.33, 0.33, 10 * self.sample_time,
                           self.sample_time, 0.1 * self.sample_time)

            p, cov, infodict, mesg, ier = leastsq(err, p_guess,
                                                  args=(time, results),
                                                  full_output=True)
            self.solution['fit'] = p
            A1, A2, tau1, tau2, tau3 = p
            A3 = 1 - A1 - A2
            self.solution['tau'] = A1 * tau1 + A2 * tau2 + A3 * tau3

        self.solution['infodict'] = infodict
        self.solution['mesg'] = mesg
        self.solution['ier'] = ier

        if ier in [1, 2, 3, 4]:  # solution found if ier is one of these values
            self.solution['estimate'] = self._my_solve(
                self.solution['time'], *p)
        else:
            warnings.warn("Solution to results not found", RuntimeWarning)

    def __repr__(self):
        return ("<MDAnalysis HydrogenBondAutoCorrel analysis measuring the "
                "{btype} lifetime of {n} different hydrogens>"
                "".format(btype=self.bond_type, n=len(self.h)))
```

Bibliography

[1] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, "MDAnalysis: A toolkit for the analysis of molecular dynamics simulations," *Journal of Computational Chemistry*, vol. 32, no. 10, pp. 2319–2327, 2011.

[2] "GNU general public license."

[3] D. Beazley and B. K. Jones, *Python Cookbook*. Beijing, Cambridge, Farnham: O'Reilly, 2013.

[4] W. McKinney, *Python for data analysis*. Beijing, Cambridge, Farnham: O'Reilly, 2013.

Appendix F

Hybridiser Source code listing

The hybridiser tool is ran by a script given below. This follows a structure of reading all inputs, modifying the inputs to adapt them for a dual scale simulation and then writing all output files.

Listing F.1: 'The command line interface to the hybridiser tool.'

```
#!/usr/bin/env python
"""The amazing hybridiser machine

Converts an atomistic input to a hybrid one

Requires:
  gro file of your atomistic system
  topol.top ffnonbonded.itp of your forcefield
  each molecule having a separate itp


Will create outputs for:
  new topol.top
  new ffnonbonded.itp
  new itp for each molecule

All output files will have their original names prefixed by "H_"

Usage:
  hybridise [options]

Options:
  -h --help      Show this screen
  --version      Show version
  -v             Verbose mode
  -s topology    Atomistic topology [default: topol.top]
  -c grofile     Input gro file     [default: conf.gro]
  -m mapfile     Input map file     [default: hybrid.map]

"""
import docopt
import logging

from hybridiser import (
    read_map,
    read_top,
    read_itps,
    read_conf,
    make_itps,
    make_conf,
    write_ndx,
    write_top,
    write_itps,
    write_conf,
)

def main(**kwargs):
    logging.debug(args)
```

```
    # Get a mda Universe
    conf = read_conf(kwargs['-c'])

    # Reading things
    # Get a dict of maps
    maps = read_map(kwargs['-m'], conf)
    # Get a list of ITP objects from the topoloy
    itps = read_itps(kwargs['-s'])
    top = read_top(kwargs['-s'])
    logging.debug(top)


    # Making new things
    new_itps = make_itps(itps, maps)
    new_conf = make_conf(conf, top, maps)

    # Writing things
    # No more work to be done from here
    write_ndx(new_conf, maps)
    write_top(kwargs['-s'])
    write_itps(new_itps)
    write_conf(new_conf)

if __name__ == '__main__':
    args = docopt.docopt(__doc__, version='Hybridise 0.1.0')

    if args['-v']:
        logging.basicConfig(level=logging.DEBUG)
        logging.debug('Debug on')

    main(**args)
```

F.1 Input functions

Listing F.2: 'confreader - Reads an input atomistic GRO file to use as a basis for creating a hybrid starting configuration'

```
import MDAnalysis as mda
import logging


def read_conf(fname):
    """Read a conf file and return a Universe"""
    logging.debug('Reading conf file "{}"'.format(fname))

    u = mda.Universe(fname)

    return u
```

Listing F.3: 'mapreader - Reads a mapping input file.'

```
"""Read .map inputs"""
import logging

from util import (
    parseline,
    strip_squarebracks,
    DefaultKeyDict,
    iter_sect,
)
from mapobjects import (
    Mapping,
    Molmap,
    Bead
)
from xmlreader import read_xml_map


def read_map(fname, u):
    """Reads either a hybridiser or xml map
```

```
    Requires filename and MDA universe of atomistic system
    """
    if fname.endswith('xml'):
        return read_xml_map(fname, u)
    else:
        return read_hybridiser_map(fname)


def read_hybridiser_map(fname)
    logging.debug('Reading map file "{}"'.format(fname))
    d = list(iter_sect(open(fname, 'r').readlines()))

    starts = [i for i, l in enumerate(d) if l.startswith('[')]
    starts.append(None)
    sections = [d[i:j] for i, j in zip(starts[:-1], starts[1:])]

    hybriddef = None
    beadtypes = None
    maps = {}
    skipnext = False
    for i, section in enumerate(sections):
        if skipnext:
            skipnext = False
            continue

        sectname = strip_squarebracks(section[0])
        if sectname == 'hybriddef':
            hybriddef = process_hybrid(section[1:])
        elif sectname == 'beadtypes':
            beadtypes = process_beadtypes(section[1:])
        elif sectname == 'cgbonds':
            raise ValueError("Out of place cgbonds section")
        else:
            molecule = process_molecule(section[1:])
            # peek ahead to see if bonds follow
            try:
                if strip_squarebracks(sections[i+1][0]) == 'cgbonds':
                    skipnext = True
                    bonds = process_cgbonds(sections[i+1][1:])
                else:
                    bonds = None
            except IndexError:
                pass  # last entry might not have cgbonds

            maps[sectname] = Molmap(molecule, bonds)

            prev = sectname

    hybridmapping = Mapping(beadtypes, hybriddef, maps)

    return hybridmapping


def process_molecule(sect):
    """Parse the molecule map"""
    beads = []
    for line in iter_sect(sect):
        line = line.split()
        logging.debug("  {}".format(line))
        idx = int(line[0])
        type = line[1]
        name = line[2]
        indices = map(int, line[3:])
        beads.append(Bead(idx, type, name, indices))
    return beads


def process_cgbonds(sect):
    """Parse the cgbonds section"""
    bonds = [map(int, line.split()) for line in iter_sect(sect)]
    return bonds


def process_hybrid(sect):
    """Process the hybriddef directive

    Allowed values: (not enforced currently)
     atomistic
     coarse
```

```
    Returns
    -------
    A dict of types -> interaction level
    """
    d = DefaultKeyDict()

    for line in iter_sect(sect):
        line = line.split()
        if line[0] == 'default':
            d.defaultvalue = line[1]
        else:
            d[line[0]] = line[1]

    return d


def process_beadtypes(sect):
    return [line for line in iter_sect(sect)]
```

Listing F.4: 'topreader - Reads the topology files of the atomistic system'

```
"""Read itp files"""
import logging
from collections import defaultdict

from util import (
    strip_squarebracks,
    iter_sect,
)
from itp import ITP


def read_top(fname):
    """Read a topology and return the molecules

    Returns
    -------
    A list of (moleculename, quantity) tuples, eg


    [ molecules ]
    ABC 4
    CDF 5
    SOL 100

    Becomes:
    [('ABC', 4),
     ('CDF', 5),
     ('SOL', 100)]
    """
    logging.debug('Read file "{}" for topology'.format(fname))

    d = open(fname, 'r').readlines()

    starts = [i for i, line in enumerate(d)
              if line.startswith('[')]
    starts.append(None)

    sections = [d[i:j] for i, j in zip(starts[:-1], starts[1:])]

    # Sort the sections according to the types
    sects = {strip_squarebracks(s[0]):s[1:] for s in sections}

    # Find the section we want (molecules)
    try:
        molecules = sects['molecules']
    except KeyError:
        raise ValueError("'molecules' section missing from topology")

    return [(l.split()[0], int(l.split()[1])) for l in iter_sect(molecules)]


def read_itps(fname):
    """Read a topology file and return the ITP objects

    Returns
    -------
    A list of ITP instances for each itp file referenced
```

```
    """
    logging.debug('Reading file "{}" for itps'.format(fname))

    itp_files = [l.split('"')[1]
                 for l in iter_sect(open(fname, 'r').readlines())
                 if '#include' in l]

    itps = [ITP(f) for f in itp_files]

    return itps
```

F.2 Processing of inputs

Listing F.5: 'confmaker - Adapts the GRO file to add the VS'

```
"""Make a new Universe for the hybrid system"""
import logging
import MDAnalysis as mda
from MDAnalysis.core.AtomGroup import AtomGroup


def make_conf(u, top, mapping):
    """Take Universe and add in VS coordinates

    Arguments
    ---------
    u - mda Universe of atomistic system
    top - tuple of (molname, number) for each section in topology
    mapping - the mapping scheme for each molecule

    Returns
    -------
    A new Universe for the hybrid system

    Notes
    -----
    Currently VS are appended onto the end of molecules.
    It might be better in the future to append the VS after the atoms that map
    to them
    """
    logging.debug("Making new conf Universe")
    logging.debug(" - Available maps : {}".format(mapping.maps.keys()))
    # Plan of attack
    # - go through molecules in topology
    # - match each molecule to a map in mapping
    # - add in extra atoms to the end of the residue
    # - return the new Universe with VSs in

    # run through the mapping twice
    # once to create the new datastructures, ie allocate
    # everything to the larger new size
    # then second time, rename and rejig things based on
    # the new size

    i = 0  # position in the Universe
    ags = []  # list of the AtomGroups

    for molname, quantity in top:
        logging.debug(" - On section {}, with {} to do"
                      "".format(molname, quantity))
        # select appropriate mapping
        molmap = mapping.maps[molname].beads
        logging.debug(" - Map is {}".format(molmap))

        # The size of each molecule
        # assumes that the mapping is correct, should really
        # check against the ITP
        molsize = sum([len(vs.indices) for vs in molmap])
        logging.debug(" - Molsize is {}".format(molsize))

        # for each molecule in this section, create the VS
        for j in xrange(quantity):
            ag = u.atoms[i:i+molsize]
            i += molsize
```

```
            new_ag = add_VS(ag, molmap)
            ags.append(new_ag)

    new_u = mda.Merge(*ags)

    logging.debug(" - Renaming atoms in new Universe")

    i = 0
    for molname, quantity in top:
        molmap = mapping.maps[molname].beads
        molsize = sum([len(vs.indices) for vs in molmap])
        n_vs = len(molmap)

        for j in xrange(quantity):
            # Include the VS in this grab of the AG
            ag = new_u.atoms[i:i+molsize+n_vs]
            for k, vs in enumerate(molmap):
                subag = ag[[val-1 for val in vs.indices]]

                ag[molsize + k].name = vs.name
                ag[molsize + k].type = vs.type
                ag[molsize + k].position = subag.center_of_mass()

            i += molsize + n_vs

    new_u.dimensions = u.dimensions

    return new_u


def add_VS(ag, molmap):
    """Take a molecule and add in the virtual sites

    ag - the AtomGroup of the molecule
    molmap - the map for this molecule
    """
    suffix = AtomGroup([])

    for idx, vstype, vsname, indices in molmap:
        # Convert to zero based indexing
        indices = [i-1 for i in indices]
        subag = ag[indices]

        vs = subag[-1]  # Copy the last
        suffix += vs
    newag = ag + suffix

    return newag
```

Listing F.6: 'itpmaker - Adapts the topology to a dual scale methodology'

```
"""Make new itps"""
import logging
import itertools

from util import (
    parseline,
    iter_sect,
    RevDict,
    RevSet,
)
from graph import MolGraph
from itp import ITP
from atom import AtomType, Atom


def make_itps(itps, mapping):
    """Modify itps according to mapping

    Arguments
    ---------
    itps - list of ITP objects
    mapping - Mapping object

    Modifies the itp objects and returns the hybrid versions
    """
    molecules = {i.molname:i for i in itps if i.type == 'molecule'}
    # the old nonbonded section
    nonbonded = [i for i in itps if i.type == 'nonbonded'][0]
```

```
    # Forcefield object to generate parameters
    ffnb = NonbondedForcefield(nonbonded['defaults'],
                               nonbonded['atomtypes'],
                               nonbonded['nonbond_params'],
                               nonbonded['pairtypes'])
    ffnb.hybriddef = mapping.hybriddef
    ffnb.beadtypes = mapping.beadtypes

    # Add extra entries required because of new atomtypes
    nonbonded['atomtypes'] += augment_atomtypes(ffnb, molecules, mapping)
    # Add extra nonbond_params for bead-bead interactions
    nonbonded['nonbond_params'] += augment_nonbond_params(ffnb)

    logging.debug("Molecule names: {}".format(molecules.keys()))
    logging.debug("Map names     : {}".format(mapping.maps.keys()))

    # Play with individual itps here
    for molname in molecules:
        augment_itp(molecules[molname], mapping.maps[molname], ffnb)

    return itps


def augment_atomtypes(ffnb, mols, mapping):
    """Based on mapping and molecules, create new atomtypes

    Arguments
    ---------
    ffnb - the forcefield object
    mols - itps for each molecule
    mapping - dict of Map objects

    Returns
    -------
    list of additional atomtypes entries
    """
    logging.debug("Making new atomtypes")
    atomtypes = set([])

    # work through all molecules
    # create mapping of bead type to atom types within that bead
    for molname in mols:
        logging.debug(" - Scanning molecule {}".format(molname))
        atoms = mols[molname]['atoms']

        for vs in mapping.maps[molname].beads:
            logging.debug("  VS is {}".format(vs))
            beadtype = vs.type
            beadname = vs.name
            for i in vs.indices:
                oldtype = atoms[i-1]['type']
                newtype = 'VS_{}'.format(oldtype)
                atomtypes.add(newtype)
                ffnb.new2old[newtype] = oldtype

    ATLINE = "  {at}  {bt}  {m}  {c}  A  {nb[0]}  {nb[1]}\n"
    ats = [';; New atomtypes for hybrid simulation\n',
           ';; These are generally the old atomtypes prefixed by the \n',
           ';; bead that they appear in\n']
    for at in atomtypes:
        oldtype = ffnb.new2old[at]
        oldat = ffnb.atoms[oldtype]

        ats.append(ATLINE.format(
            at=at,
            bt=oldat['btype'],
            m=oldat['m'],
            c=0.0,
            nb=(0.0, 0.0),
        ))
    ats.append(';; Begin the atoms that are actually beads\n')
    for b in ffnb.beadtypes:
        ats.append(ATLINE.format(
            at=b,
            bt=b,
            m=0.0,
            c=0.0,
            nb=(0.00, 0.00),
            )
```

```
        )
    ats.append('\n')
    return ats


def augment_nonbond_params(ffnb):
    """Create the coarse-coarse nonbonded parameters

    Arguments
    ---------
    ffnb - the ForceField object

    Returns
    -------
    a list of extra nonbond_params for the ffnonbonded itp file
    """
    nbs = []
    for A, B in itertools.combinations_with_replacement(
            ffnb.beadtypes, 2):
        if (ffnb.hybriddef[A] == 'coarse' or
            ffnb.hybriddef[B] == 'coarse'):
            nbs.append("{} {} are coarse\n".format(A, B))
            logging.debug("{} {} are coarse".format(A, B))
        else:
            logging.debug("{} {} are atomistic".format(A, B))
    return nbs


def augment_itp(itp, itp_map, ffnb):
    """Augment a single molecule itp to be hybrid

    Arguments
    ---------
    itp - the molecule itp
    itp_map - the mapping for this moleclue
    ffnb - the Forcefield object

    Returns
    -------
    nothing - molecule itp modified in place
    """
    # nrexcl from itp
    for line in iter_sect(itp['moleculetype']):
        nrexcl = int(line.split()[1])

    # count atoms
    atoms = itp['atoms']
    natoms = len(atoms)
    logging.debug("Found {} atoms".format(natoms))

    # Turn the cgbonds section into a graph
    bead_graph = MolGraph([b.index for b in itp_map.beads],
                          itp_map.cgbonds)

    # Construct a graph of the atomistic bonding
    if 'bonds' in itp:
        atom_bonds = [map(int, line.split()[:2])
                      for line in iter_sect(itp['bonds'])]
    else:
        atom_bonds = None
    atom_graph = MolGraph([a['idx'] for a in atoms],
                          atom_bonds)

    # copy existing pairs
    # for all existing pairs:
    # if the pair already has parameters, leave it
    # otherwise, write down the parameters for this pair
    # take into account the fudge LJ and fudge QQ
    # this is because the atom parameters can be 0 in atomtypes
    # so generating pairs won't work
    pairs = []
    for p in iter_sect(itp['pairs']):
        if len(p.split()) > 3:  # if we have parameters, leave it
            pairs.append(p)
            continue
        p = p.split()
        na, nb = int(p[0]), int(p[1])
        ta = atoms[na - 1]['type']  # types haven't been fucked with yet
        tb = atoms[nb - 1]['type']
        V, W = ffnb.gen_pair((ta, tb))
```

```
            pairs.append(" {:<5s} {:<5s}  1  {:E} {:E}\n".format(
                p[0], p[1], V, W))
        pairs.append('; Begin pairs from hybrid method\n')
193     for vs in itp_map.beads:  # for each bead
            for neb in bead_graph.get_neighbours(vs[0], self=True):
                other = itp_map.beads[neb - 1]
                # Don't add pairs if both beads are atomistic
                if (ffnb.hybriddef[vs.type] == 'atomistic'
198                 and ffnb.hybriddef[other.type] == 'atomistic'):
                    logging.debug("Beads {} and {} are both atomistic, "
                                  "skipping manual pairs".format(vs, other))
                    continue

203                 for x, y in itertools.product(vs.indices, other.indices):
                        if y < x:  # avoid duplicate pairs, ie (1,2) and (2,1)
                            continue
                        if atom_graph.get_nexcl(x, y) <= nrexcl:
                            continue
208                     ta = atoms[x - 1]['type']
                        tb = atoms[y - 1]['type']
                        V, W = ffnb.gen_pair((ta, tb))
                        pairs.append(" {:<5d} {:<5d}  1  {:E} {:E}\n".format(
                            x, y, V, W))
213     pairs.append("\n")
        # REPLACE the previous section
        itp['pairs'] = pairs

        # Retype atoms
218     # Add virtual sites between other atoms
        vs2at = {}  # convert virtual site index to atom index
        new_atoms = []
        virt_atoms = []
        # figure out what the highest charge group was in the atoms
223     cgnr_start = max([a['cgnr'] for a in atoms]) + 1
        for i, vs in enumerate(itp_map.beads):
            for at_idx in vs.indices:
                # Rename the atoms in this mapping if coarse type
                if ffnb.hybriddef[vs.type] == 'coarse':
228                 oldtype = atoms[at_idx - 1]['type']
                    atoms[at_idx - 1]['type'] = 'VS_{}'.format(
                        oldtype)
                    # VS atoms can't have charge
                    atoms[at_idx - 1]['charge'] = 0.0
233             new_atoms.append(atoms[at_idx - 1])

            # Take some details off the last atom in my mapping
            atom = atoms[vs.indices[-1] - 1]
            resid = atom['resid']
238         resname = atom['resname']
            cgnr = cgnr_start + i
            virtual_atom = Atom(0, vs.type, resid, resname,
                                vs.name, cgnr, 0.00, 0.00)
            virt_atoms.append(virtual_atom)
243         vs2at[vs.index] = virtual_atom
        # Virtual atoms always come after real atoms
        new_atoms.extend(virt_atoms)

        # Renumber
248     for i, a in enumerate(new_atoms):
            a['idx'] = i + 1

        itp['atoms'] = [str(a) for a in new_atoms]
        itp['atoms'].append("\n")
253
        # Exclusions between virtual sites
        for vs in itp_map.beads:
            nebs = bead_graph.get_neighbours(vs.index, self=False)
            if not nebs:
258             continue
            itp['exclusions'].append(
                "  {}    {}\n"
                "".format(vs2at[vs[0]]['idx'], " ".join(str(vs2at[i]['idx'])
                                                        for i in nebs))
263         )

        # virtual_sitesn - the mapping of atoms into virtual sites
        vsites = []
        for i, vs in enumerate(itp_map.beads):
268         vsites.append(
```

```
            "  {}  2  {}\n"
            "".format(i + 1 + natoms, " ".join(str(i) for i in vs.indices))
        )
    itp['virtualsitesn'].extend(vsites)


def check_map(i, m):
    idx = [a['idx'] for a in i['atoms']]
    idx2 = list(itertools.chain(*[vs.indices for vs in m.beads]))

    if not idx == idx2:
        raise ValueError("Mapping is incorrect")


class NonbondedForcefield(object):
    """The nonbonded parameters for a forcefield"""
    def __init__(self, defaults, atomtypes, nonbond_params, pairtypes):
        for line in iter_sect(defaults):
            try:
                self.nonbondtype = int(line.split()[0])
                self.mixing = int(line.split()[1])
                # genpairs
                # fudgeLJ
                # fudgeQQ
            except IndexError:
                pass

        self.fudgeLJ = 1.0
        self.fudgeQQ = 1.0

        # Map atoms to another dict of their properties
        self.atoms = dict()
        for line in iter_sect(atomtypes):
            a = AtomType(line)
            self.atoms[a['type']] = a

        # Map atom type pairs to parameters
        self.nonbond_params = RevDict()
        for line in iter_sect(nonbond_params):
            line = line.split()
            ta, tb = line[0], line[1]
            params = float(line[-2]), float(line[-1])
            self.nonbond_params[ta, tb] = params

        self.pairs = RevDict()
        for line in iter_sect(pairtypes):
            line = line.split()
            ta, tb = line[0], line[1]
            params = float(line[-2]), float(line[-1])
            self.pairs[ta, tb] = params

        # Convert atom types from new/hybrid to old/atomistic
        # all new types are $BEADTYPE_$ATOMTYPE
        # but maybe there's underscores in the name, so rather
        # than split around them, lets use this dict
        self.new2old = dict()

    def __getitem__(self, pair):
        try:
            return self.nonbond_params[pair]
        except KeyError:
            return self.gen_nonbond(*pair)

    def gen_pair(self, pair):
        try:
            return self.pairs[pair]
        except KeyError:
            V, W = self.gen_nonbond(*pair)
            # add fudgeLJ and fudgeQQ here
            # check mixing rules for application of fudge factors
            V *= self.fudgeLJ
            W *= self.fudgeLJ
            return V, W

    def gen_nonbond(self, ta, tb):
        # mix the parameters for ta and tb according to the mixing
        # rules for this forcefield
        if self.mixing == 1:
            return self._gen_mix_1(ta, tb)
        elif self.mixing == 2:
```

```
            return self._gen_mix_2(ta, tb)

    def _gen_mix_1(self, ta, tb):
        """Geometrix mixing rules"""
        Va, Wa = self.atoms[ta]['nonbonded']
        Vb, Wb = self.atoms[tb]['nonbonded']

        Vab = (Va * Vb)**0.5
        Wab = (Wa * Wb)**0.5

        return Vab, Wab

    def _gen_mix_2(self, ta, tb):
        """Lorentz Berthelot mixing rules"""
        Va, Wa = self.atoms[ta]['nonbonded']
        Vb, Wb = self.atoms[tb]['nonbonded']

        Vab = (Va * Vb)**0.5
        Wab = 0.5 * (Wa + Wb)

        return Vab, Wab
```

F.3 Output functions

Listing F.7: 'confwriter - Writes the new GRO file'

```
"""Write out the new Universe"""
import logging


def write_conf(u):
    fout = 'H_conf.gro'
    logging.debug('Writing new conf file to "{}"'.format(fout))

    u.atoms.write(fout)
```

Listing F.8: 'indexwriter - Writes the ndx file for identifying the dual scale system

```
"""Makes a gromacs index file for a hybrid simulation"""

def write_ndx(u, maps):
    """Ndx file with entries for system and all beads

    Arguments
    ---------
    u - the coarse Universe
    maps - the mapping scheme for this system
    """
    sels = {}

    # select each bead in turn
    for beadtype in maps.beadtypes:
        sels[beadtype] = u.select_atoms('type {}'.format(beadtype))

    # atoms is everything that isn't a bead
    everything = set(u.atoms)
    for other in sels.values():
        everything -= set(other)
    everything = list(everything)
    everything.sort()
    sels['Atoms'] = everything

    # Everything for thermostat
    sels['System'] = u.atoms

    # We want System, then Atoms, then rest in alphabetical order
    names = sels.keys()
    names.remove('System')
    names.remove('Atoms')
    names.sort()
    names = ['System', 'Atoms'] + names
```

```
    with open('H_index.ndx', 'w') as out:
        for selname in names:
            ag = sels[selname]
            out.write("[ {} ]\n".format(selname))
            for i, atom in enumerate(ag):
                if not i % 5:
                    out.write("\n")
                out.write("{} ".format(atom.index + 1))
            out.write("\n")
            out.write("\n")
```

Listing F.9: 'itpwriter - Writes the new itp files out'

```
"""Write itps"""
import logging



def write_itps(mols):
    for mol in mols:
        fout = 'H_' + mol.filename
        mol.write(fout)
```

Listing F.10: 'topwriter - Write out the new topology file'

```
"""Write out a shiny new top file that refers to the hybrid itps"""
import logging

from util import parseline

def write_top(fname):
    fout = 'H_' + fname
    logging.debug('Writing new top file to "{}"'.format(fout))

    d = open(fname, 'r').readlines()

    with open(fout, 'w') as out:
        out.write(";; Hybrid system\n")
        for line in d:
            content, comment = parseline(line)

            if '#include' in content:
                itpfile = content.split('"')[1]
                logging.debug("Itp file is: {}".format(itpfile))
                content = content.replace(itpfile, 'H_' + itpfile)

            out.write(content + comment)
```

F.4 Utility functions and classes

Listing F.11: 'atom - Classes for representing an Atom'

```
"""Class for representing an AtomType line in gromacs

    /* Comments on optional fields in the atomtypes section:
     *
     * The force field format is getting a bit old. For OPLS-AA we needed
     * to add a special bonded atomtype, and for Gerrit Groenhofs QM/MM stuff
     * we also needed the atomic numbers.
     * To avoid making all old or user-generated force fields unusable we
     * have introduced both these quantities as optional columns, and do some
     * acrobatics to check whether they are present or not.
     * This will all look much nicer when we switch to XML... sigh.
     *
     * Field 0 (mandatory) is the nonbonded type name. (string)
     * Field 1 (optional)  is the bonded type (string)
     * Field 2 (optional)  is the atomic number (int)
     * Field 3 (mandatory) is the mass (numerical)
     * Field 4 (mandatory) is the charge (numerical)
     * Field 5 (mandatory) is the particle type (single character)
     * This is followed by a number of nonbonded parameters.
```

```
     *
     * The safest way to identify the format is the particle type field.
     *
     * So, here is what we do:
     *
     * A. Read in the first six fields as strings
     * B. If field 3 (starting from 0) is a single char, we have neither
     *    bonded_type or atomic numbers.
     * C. If field 5 is a single char we have both.
     * D. If field 4 is a single char we check field 1. If this begins with
     *    an alphabetical character we have bonded types, otherwise atomic numbers.
     */
"""
class AtomType(dict):
    """Represents an atomtype line from ffnonbonded.itp"""
    def __init__(self, line):
        """Given an atomtypes line, populate self

        Only does LJ atomtypes

        Provides following keys:
         - type (atom type)
         - btype (bonded type)
         - m (mass)
         - q (charge)
         - nonbonded (nonbonded parameters)
        """
        # This is just a python port of toppush.c
        line = line.strip().split()
        if len(line) < 6:
            raise ValueError("Atom line too small")

        if (len(line[5]) == 1) and line[5][0].isalpha():
            have_bonded_type = True
            have_atomic_number = True
        elif (len(line[3]) == 1) and line[3][0].isalpha():
            have_bonded_type = False
            have_atomic_number = False
        else:
            have_bonded_type = line[1][0].isalpha()
            have_atomic_number = not have_bonded_type

        if have_bonded_type and have_atomic_number:
            type, btype, atomnr, m, q, ptype, V, W = line[:8]
        elif have_bonded_type and not have_atomic_number:
            type, atomnr, m, q, ptype, V, W = line[:7]
            btype = type
        elif not have_bonded_type and have_atomic_number:
            type, btype, m, q, ptype, V, W = line[:7]
        else:
            type, m, q, ptype, V, W = line[:6]
            btype = type

        m, q, V, W = map(float, [m, q, V, W])

        self['type'] = type
        self['btype'] = btype
        self['m'] = m
        self['q'] = q
        self['nonbonded'] = V, W


class Atom(dict):
    """Represents an atom within a molecule itp"""
    def __init__(self, idx, atype, resid, resname, name, cgnr,
                 charge=None, mass=None):
        """
        mass and charge optional, will then default to itp atomtypes values
        """
        self['idx'] = idx
        self['type'] = atype
        self['resid'] = resid
        self['resname'] = resname
        self['name'] = name
        self['cgnr'] = cgnr
        self['mass'] = mass if mass else ''
        self['charge'] = charge if charge else ''

    @classmethod
    def from_line(cls, line):
```

```
        line = line.strip().split()
        idx, atype, resid, resname, name, cgnr = line[:6]
        idx, resid, cgnr = map(int, [idx, resid, cgnr])
        # try and read mass and charge info
        try:
            charge = float(line[6])
        except (ValueError, IndexError):
            charge = None
        try:
            mass = float(line[7])
        except (ValueError, IndexError):
            mass = None

        atom = cls(idx, atype, resid, resname, name, cgnr, charge, mass)

        return atom

    def __str__(self):
        return ' {0:6d} {1:10s} {2:5d} {3:5s} {4:5s} {5:2d} {6} {7}\n'.format(
            self['idx'],
            self['type'],
            self['resid'],
            self['resname'],
            self['name'],
            self['cgnr'],
            self['charge'],
            self['mass'],
        )
```

Listing F.12: 'graph - Graph representations of molecules'

```
"""Graphs of molecules"""
import networkx as nx


class MolGraph(nx.graph.Graph):
    def __init__(self, atoms, bonds=None):
        super(MolGraph, self).__init__()
        self.add_nodes_from(atoms)
        if bonds:
            self.add_edges_from(bonds)

    def get_neighbours(g, idx, dist=2, self=True):
        """Return the neighbours with dist of idx"""
        nebs = nx.single_source_shortest_path_length(
            g, idx, dist).keys()
        if not self:
            nebs.remove(idx)
        return nebs

    def get_nexcl(self, a, b):
        """Return the number of bonds between *a* and *b*"""
        return nx.shortest_path_length(self, a, b)
```

Listing F.13: 'itp - Classes for representing an itp file'

```
from collections import defaultdict
import logging

from atom import Atom
from util import strip_squarebracks, iter_sect


class ITP(defaultdict):
    SECTS = {
        'molecule': (
            'moleculetype',
            'atoms',
            'bonds',
            'angles',
            'dihedrals',
            'pairs',
            'exclusions',
            'virtualsitesn',
        ),
        'bonded': (
            'bondtypes',
```

```
            'angletypes',
            'dihedraltypes',
        ),
        'nonbonded': (
            'defaults',
            'atomtypes',
            'nonbond_params',
            'pair_types',
        ),
    }

    def __init__(self, filename=None):
        super(ITP, self).__init__(list)

        self.filename = filename
        if not filename is None:
            self._read_itp()

    def _read_itp(self):
        d = open(self.filename, 'r').readlines()

        starts = [i for i, line in enumerate(d)
                  if line.startswith('[')]
        starts.append(None)
        sections = [d[i:j] for i, j in zip(starts[:-1], starts[1:])]

        for section in sections:
            name = strip_squarebracks(section[0])
            self[name] += section[1:]

        if 'moleculetype' in self:
            self.type = 'molecule'
            self.molname = self.get_name()
            self['atoms'] = [Atom.from_line(l)
                             for l in iter_sect(self['atoms'])]
        elif 'atomtypes' in self:
            self.type = 'nonbonded'
        else:
            self.type = 'bonded'

    def get_name(self):
        """Figure out who this dude is"""
        for line in iter_sect(self['moleculetype']):
            try:
                return line.split()[0]
            except IndexError:
                pass

    def write(self, fout):
        """Write this itp to file *fout*"""
        with open(fout, 'w') as out:
            for sect in self.SECTS[self.type]:
                if sect in self and not self[sect] is None:
                    out.write('[ {} ]\n'.format(sect))
                    out.write(''.join(str(s) for s in self[sect]))
                    out.write('\n')
```

Listing F.14: 'mapobjects - Classes for representing maps'

```
"""Classes that represent a mapping

Mapping
-------
Entire mapping scheme for system.

Values:
  beadtypes - list of beadtypes in system
  hybriddef - the interaction level between different types
  beads - dict of moleculename:mapping

Molmap
------
The mapping for a single molecule

Values:
  beads - various Bead objects (see below)
  cgbonds - the connectivity between the beads.  Stored as tuple
            of bead indices
```

```
    Bead
    ----
23  A single VS

    Values:
      index - the index of the VS site (in CG terms)
      type - the type of the VS (used in hybriddef)
28    name - the name of the VS (must be unique)
      indices - list of the atom indices that this VS holds (1 based)
    """

    from collections import namedtuple
33
    Mapping = namedtuple('Mapping', ['beadtypes', 'hybriddef', 'maps'])
    Molmap = namedtuple('Map', ['beads', 'cgbonds'])
    Bead = namedtuple('Bead', ['index', 'type', 'name', 'indices'])
```

Listing F.15: 'util - Utility functions for reading files'

```
    """Useful stuff and such"""


4   class DefaultKeyDict(dict):
        """Dict which you can set .defaultvalue attribute for missing keys"""
        def __missing__(self, key):
            try:
                return self.defaultvalue
9           except AttributeError:
                raise KeyError(
                    "No interaction level for pair {} has been defined"
                    " and no default level was defined"
                    "".format(key))
14


    class RevDict(DefaultKeyDict):
        """Allows keys to be reversed"""
        def __getitem__(self, key):
19          if key in self:
                return dict.__getitem__(self, key)
            else:
                return dict.__getitem__(self, key[::-1])


24
    class RevSet(set):
        """A set which checks if the reverse of a key exists"""
        def add(self, other):
            if not other[::-1] in self:
29              set.add(self, other)

        def update(self, others):
            # can't just use update if duplicates exist in others
            for other in others:
34              self.add(other)


    def strip_squarebracks(stuff):
        """Strip the square brackets from around something"""
39      return stuff.rpartition('[')[-1].partition(']')[0].strip()


    def parseline(line):
        """handle comment lines
44
        returns:
          line content, comment section + newline
        """
        parts = line.split(';', 1)
49      try:
            return parts[0], ';' + parts[1]
        except IndexError:
            return line.strip(), '\n'


54
    def iter_sect(lines):
        """Generator which strips comments and ignores empty lines"""
        for line in lines:
            p = parseline(line)[0].strip()
59          if not p:
                continue
```

```
        yield p
```