UNIVERSITY OF SOUTHAMPTON

MASTERS THESIS

Software for the Automated Generation of Coarse-Grained Molecular Dynamics Models

Author:
James A. GRAHAM

Supervisor:
Prof. Jonathan W. ESSEX
Prof. Syma KHALID

A thesis submitted in fulfilment of the requirements for the degree of Master of Philosophy

in the

Computational Systems Chemistry
School of Chemistry

April 25, 2019

Declaration of Authorship

UNIVERSITY OF SOUTHAMPTON

Abstract

Faculty of Engineering and Physical Sciences
School of Chemistry

Master of Philosophy

Software for the Automated Generation of Coarse-Grained Molecular Dynamics Models

by James A. GRAHAM

This work presents a software tool for the automated parametrisation of coarse-grained (CG) molecular dynamics (MD) models and the environment that led to its necessity. It is available at https://github.com/jag1g13/pycgtool and had been registered as doi:10.5281/zenodo.569555.

Numerical simulations are used in chemistry to span the necessary range in scale from atoms to biological systems such that we might understand their behaviour. However, the computational demand to simulate such systems at the atomic level is increasing as we continue to study larger systems over longer timescales. Significant efficiency gains are seen when employing CG MD over traditional all-atom (AA) MD, at the cost of some accuracy.

Sugars are a component of many biomolecule classes such as carbohydrates, glycoproteins and lipopolysaccharides, but are relatively neglected in molecular dynamics simulations. In recent years there has been a trend towards CG models in the field of biomolecular simulation, allowing larger systems to be simulated for longer, but there are currently no satisfactory CG representations of sugars.

The first strand of research described here was to develop a CG model of sugars as part of the existing ELBA forcefield, using a high quality electrostatic model incorporating dipoles as well as point charges. This can be seen in sections 2 and 3.

However, creating an adequate model required a representation of sugars which did not provide benefits over existing AA or united-atom (UA) models. Because of this, the research was taken in a different direction: to address the difficulty of developing CG models, by automation of significant parts of the process. The new aim was to develop a software tool which would allow researchers to more quickly and simply develop their own models, both within the ELBA forcefield, and within the popular MARTINI forcefield.

In order to address this, software was written to automate parts of the process of creating a new coarse-grained molecular dynamics model. The software makes use of a modification to the Boltzmann Inversion used by other software for forcefield parametrisation, which means that the calculated parameters are consistent with established forcefields such as ELBA and MARTINI. Though there are other tools fulfilling a similar purpose, they do not make use of this modification, and thus their output is not consistent with these models. Focus was given to ensuring that best practices in research software engineering (RSE, a term only now becoming established) were followed, ensuring that the software is reliable, usable and extensible.

The software was used in the parametrisation of the ELBA sugar model, and has been used by others for parametrisation of models within the MARTINI forcefield.

Acknowledgements

Emma Nelson
Much moral support & the conical flask analogy

My family
Much moral support

Vicky Randall & Brendan Neville
Much moral support

Nicki Lewin
Sufficient pressure to write a thesis

Marley Samways
Assistance in development of the ELBA sugar model and its continuation

Jon Shearer
Continuing development of PyCGTOOL

Chris Cave-Ayland
Encouragement and guidance in Research Software Engineering

Pin Chia Hsu & Damien Jeffries
Testing and validation of PyCGTOOL and its predecessor CGTOOL

Christopher Reynolds
Assistance in multipole analysis from QM data

Jon Essex & Syma Khalid
Guidance and advice over the course of the project

All members of Jon and Syma's groups & the ICSS DTC

Contents

List of Figures

List of Tables

List of Abbreviations

AA	All Atom (often used to include **UA**)
ALLA / ALLB	α-allose / β-allose
ALTA / ALTB	α-altrose / β-altrose
AMBER	Both widely user molecular dynamics simulator and **AA** biomolecular force field usually used together
BI / IBI	(Iterative) Boltzmann Inversion
CG	Coarse-Grain(ed)
CV	Collective Variables
DFT	Density Functional Theory - a family of **QM** methods
DMA	Distributed Multipole Analysis
ELBA	A **CG** forcefield for simulation of lipids used with **LAMMPS**
FM	Force Matching
GAFF	General **AMBER** Force Field
GALA / GALB	α-galactose / β-galactose
GLCA / GLCB	α-glucose / β-glucose
GLYCAM	A widely used force field for sugars
(GP)GPU	(General Purpose) Graphics Processing Unit
GROMACS	A widely used molecular dynamics simulator
GROMOS	A widely used **UA** biomolecular force field usually used with **GROMACS**
GULA / GULB	α-gulose / β-gulose
IDOA / IDOB	α-idose / β-idose
IO	Input/Output
IUPAC	International Union of Pure and Applied Chemistry
KDE	Kernel Density Estimator / Estimation
LAMMPS	A very flexible molecular dynamics simulator
LPS	LipoPolySaccharide(s)
MANA / MANB	α-mannose / β-mannose
MARTINI	The *de facto* standard **CG** biomolecular force field used with **GROMACS**
MD	Molecular Dynamics
NMR	Nuclear Magnetic Resonance
NPT	A statistical ensemble which fixes the number of particles, pressure and temperature
NVT	A statistical ensemble which fixes the number of particles, volume and temperature
OPLS	A force field for simulation of small molecule liquids available in **AA** and **UA** forms
PCM	Polarizable Continuum Model - a solvent model often used in **QM** calculations
PME	Particle Mesh Ewald - a method for $\mathcal{O}(N \log N)$ calculation of electrostatic interactions

QM	Quantum Mechanics
RDF	Radial Distribution Function
RMSD	Root Mean Square Deviation - a measure of goodness-of-fit
RSE	Research Software Engineering / Engineer
TALA / TALB	α-talose / β-talose
TDD	Test Driven Development
UA	United Atom
VdW	Van der Waals

To Emma

Chapter 1

Introduction

In studying the detailed behaviour of a physical system, science is often limited by the ability of equipment to observe the necessary range in scale in dimensions of both space and time. In chemistry, this limitation is encountered at the smaller end of the spectrum: devices that can observe chemical systems at the resolution of atoms are often unable to observe a large enough or long enough timescale to be useful to biochemists or materials scientists.

To span the necessary range in scale from atoms to biological systems, or materials properties, we may make use of numerical simulations.

1.1 Molecular Dynamics

Molecular dynamics (MD) is the study of the motion and ensemble properties of molecules, in the context of numerical simulation. Since its initial development in the 1950s and 60s the fundamental algorithm has not significantly changed [1] [2].

Two of the major application domains of molecular dynamics are materials science, typically solid state, and biochemistry, typically liquid/aqueous. Materials science is generally concerned with the properties of large metallic or crystalline slabs, whereas biochemical simulations generally consist of a number of discrete organic molecules solvated in water. Both applications use similar methods, although there are optimisations that can be made to each at the loss of some generality.

As the performance per dollar of computer hardware has increased exponentially since the invention of the modern computer, so too has the demand placed on it. It is routine to perform simulations now on desktop hardware which would have been possible only on the largest supercomputers of 20 years ago, with several orders of magnitude performance increase [3]. But even accounting for continual improvement of hardware, there is a push toward ever larger simulations which requires other developments. One major recent improvement has been the offloading of some parts of numerical simulation to GPU (Graphics Processing Unit) accelerator cards which are designed for highly parallel floating point number computation. Most of the significant molecular dynamics simulators (notably AMBER [4], GROMACS [5] and LAMMPS [6]) have been ported to run on GPUs and have been used successfully on both desktop systems and supercomputers.

As well as increasing the performance of the hardware it is beneficial to consider improvements to the simulation protocol. This may be through methods such as replica exchange [7] or metadynamics [8] which attempt to increase the efficiency of sampling but maintain the same physical description of the system, or through methods such as coarse-graining or bond constraints which directly reduce the number of degrees of freedom while allowing larger simulation timesteps to be used.

1.1.1 Forcefields

Whereas *ab initio* Quantum Mechanics (QM) determines the forces acting upon atoms from physical principles, Molecular Mechanics (MM) approaches rely upon an approximation of reality. By treating molecules as a collection of balls (atoms) and springs (bonds), it is possible to apply the equations of classical mechanics instead.

The fundamental equation on which all molecular dynamics models (either QM or MM) are built is Newton's Second Law 1.1.

$$\mathbf{F} = m\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = m\mathbf{a} \tag{1.1}$$

Since the mass of a particle can be considered fixed, if we know the force acting upon the particle, we may calculate its trajectory of motion over time. In turn, the force may be calculated from the energy of the system 1.2

$$\mathbf{F} = -\frac{\mathrm{d}U}{\mathrm{d}\mathbf{r}} \tag{1.2}$$

When using an MM model, the energy U is decomposed using the 'ball and spring' approximation into a set of potential energy functions, known as a 'forcefield'. The forces described by a typical forcefield may be divided into two categories: 'bonded' and 'non-bonded' 1.3. The non-bonded terms comprise the electrostatic forces and the repulsive-dispersive, or Van der Waals (VdW) forces 1.4. The bonded terms typically comprise potentials representing bond lengths, bond angles, and dihedral angles; though some forcefields opt to include more complex combinations of these 1.5.

$$U_{\text{pot}} = U_{\text{bonded}} + U_{\text{non-bonded}} \tag{1.3}$$

$$U_{\text{non-bonded}} = U_{\text{electrostatic}} + U_{\text{VdW}} \tag{1.4}$$

$$U_{\text{bonded}} = U_{\text{bonds}} + U_{\text{angles}} + U_{\text{dihedrals}} \tag{1.5}$$

The potential energy functions chosen to represent the non-bonded terms are typically Coulombs Law 1.6 for the electrostatic component and the Lennard-Jones 6-12 potential 1.7 for the VdW component. There are many alternatives to the Lennard-Jones 6-12 potential which see infrequent use, but its advantage in computational efficiency due to the choice of the factors six and 12, mean that Lennard-Jones has become the default choice.

$$U_{\text{electrostatic}} = \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \tag{1.6}$$

$$U_{\text{VdW}} = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \equiv \frac{A}{r^{12}} - \frac{B}{r^{6}} \tag{1.7}$$

There is much greater variation in the potential energy functions used for bonded potentials, with many forcefields using a different combination [9]. There are two commonly used forms for each of the components of the bond length and angle terms, with one dihedral form generally prefered, but multiple alternatives seeing some use.

For bond lengths there are:

- The simple harmonic potential used by AMBER, MARTINI and ELBA 1.8

- The fourth power potential used by GROMOS 1.9

For bond angles there are:

- The harmonic angle potential used by AMBER 1.10
- The cosine angle potential used by GROMOS, MARTINI and ELBA 1.11

For dihedrals angles there are:

- The periodic proper dihedral used by GROMOS, and sometimes MARTINI and ELBA 1.12
- The Fourier series 1.13
- The Ryckaert-Bellemans function 1.14

$$U_{\text{bonds}} = \frac{1}{2} \sum_{bonds} k_b (l - l_0)^2 \tag{1.8}$$

$$U_{\text{bonds}} = \frac{1}{4} \sum_{bonds} k_b (l^2 - l_0^2)^2 \tag{1.9}$$

$$U_{\text{angle}} = \frac{1}{2} \sum_{angles} k_a (\theta - \theta_0)^2 \tag{1.10}$$

$$U_{\text{angle}} = \frac{1}{2} \sum_{angles} k_a (\cos\theta - \cos\theta_0)^2 \tag{1.11}$$

$$U_{\text{dihedral}} = k_\phi (1 + \cos(n\phi - \phi_s)) \tag{1.12}$$

$$U_{\text{dihedral}} = \frac{1}{2} [C_1(1 + \cos\phi) + C_2(1 - \cos 2\phi) + C_3(1 + \cos 3\phi) + \dots] \tag{1.13}$$

$$U_{\text{dihedral}} = \sum_{i=0}^{n} C_i (\cos\psi)^i \quad \text{where} \quad \psi = \phi - 180\,\text{deg} \tag{1.14}$$

In the case of bond lengths and angles the former of the two potential energy functions is simpler mathematically, whereas the latter is more computationally efficient, usually by excluding costly operations from the derivative such as division or arccos.

Karplus noted that, "although the potentials used in simulations are approximate, they are completely under the control of the user, so that by removing or altering specific contributions their role in determining a given property can be examined" [10].

1.1.2 Coarse-Graining

One of the most significant recent developments in terms of simulation efficiency has been the introduction of coarse-grained (CG) molecular dynamics which has given very large speedups at the cost of some accuracy. The principle behind CG MD is simple: reduce the number of particles being simulated by collecting individual atoms into larger ‘beads’. In this way it can be seen as an extension of the united atom (UA) forcefields such as GROMOS. As with a UA model the improved efficiency comes from both a reduction in the number of particles being simulated and the possibility to use a larger timestep between force evaluations. Since high frequency vibrations, such as that of a C−H bond, are no longer present, the simulation timestep can be increased safely, typically to 10 fs to 40 fs compared to the standard of 1 fs to 2 fs for an all atom (AA) or UA simulation.

When developing a CG forcefield, there is a trade-off to be made of simulation efficiency against replication of the detail seen with an all atom model. It is this decision which determines the number of atoms to be combined into a single bead and also which other details are to be neglected, if any. In practice there is wide agreement on 3-4 heavy (non-hydrogen) atoms to a single bead, which gives a significant gain in efficiency, but retains detail in the structure and conformation of large molecules.

1.1.3 Coarse-Grained Forcefields

Although coarse-graining carries a potentially massive gain in sampling efficiency there has not yet been a proliferation of widely used forcefields as with AA and UA models. While traditional MD has the popular AMBER, CHARMM (both AA) and GROMOS (UA) forcefields and a large number of less widely used forcefields, CG MD has as of yet only one widely used forcefield in biochemical simulation; the MARTINI forcefield [11, 12].

MARTINI

MARTINI uses the common mapping of four heavy atoms into a single bead and is based upon the GROMOS functional form, being used with the GROMACS simulator. It was designed for the study of biomolecules, with parameter sets available for proteins and a wide range of lipids, for proteins [13] and for carbohydrates [14]. In order to simplify the application of the model to new molecules, MARTINI specifies a set of pre-prepared bead types which are designed to be generally applicable to a wide range of contexts. These beads are sorted and named in order of polarity, from the charged 'Q' beads to the apolar 'C' beads with a range of subtypes for finer control. The ease with which MARTINI is used is perhaps the most significant factor in its wide user base. A 'Dry Martini' variant of MARTINI exists which avoids the use of solvent particles by modifying the interaction strengths of existing MARTINI beads [15]; it is currently parametrised only for lipids and is not widely used, but allows much greater computational efficiency than even the standard MARTINI model.

To create a MARTINI parametrisation of a new molecule it is necessary to invent a bead mapping scheme and assign types and charges to each bead. These types determine the Lennard-Jones parameters, so only bonded terms remain to be determined. Bonded terms are generally parametrised by measurements taken from atomistic simulation for equilibrium values, while force constants are set to a 'typical value' dependent on their type. Among the lipid models distributed by the MARTINI group most bonded terms are not individually parametrised, but use, for example, generic values of 0.47 nm and a force constant of 1250 $\mathrm{kJ\,mol^{-1}\,nm^{-2}}$ for the majority of bond lengths. Angle and dihedral terms are similarly generic. Molecules parametrised by other groups do not usually make such significant use of generic parameters.

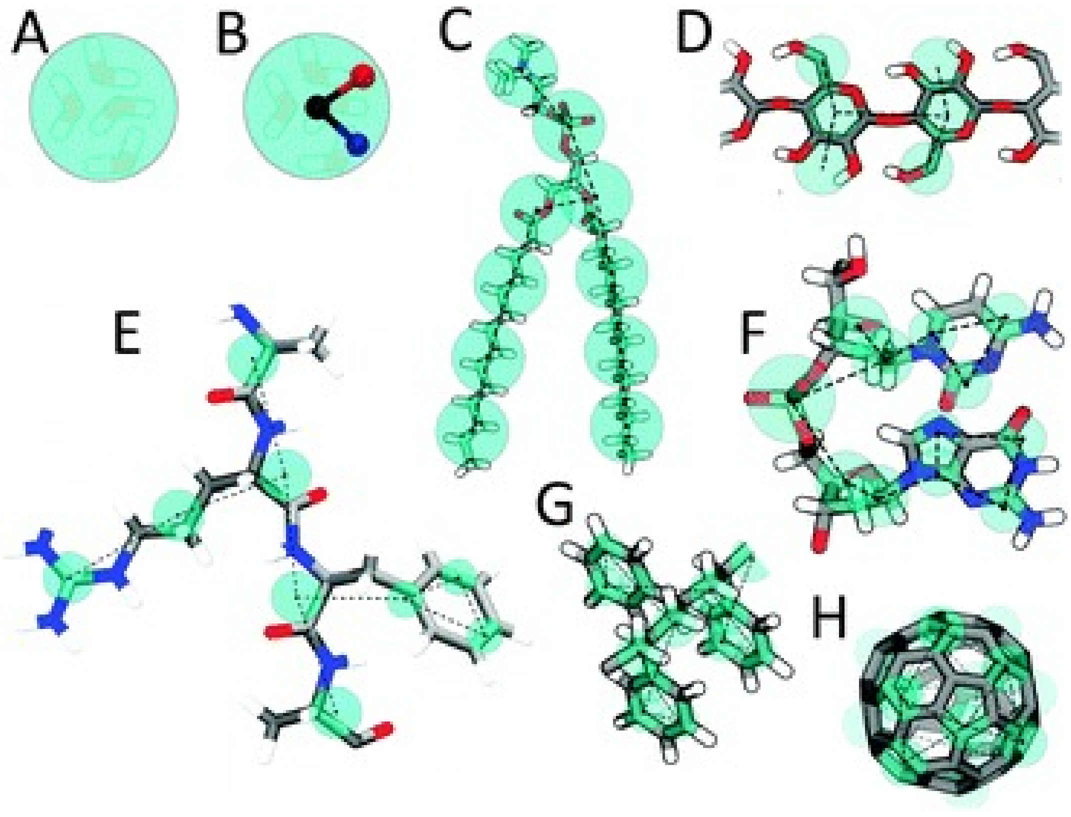


FIGURE 1.1: Example bead mappings used in the MARTINI model [16]

The trade-off made for the convenience of the MARTINI model is that the predefined beads do not allow bead properties to be fit as accurately as may be achieved using a method such as Iterative Boltzmann Inversion (IBI) [17] or Force Matching (FM) [18]. The quality of simulations performed using the MARTINI model is usually satisfactory, though in some cases qualitatively incorrect behaviour may be observed [16].

ELBA

The ELBA coarse-grained force field is more complex than MARTINI, allowing beads to have point monopoles and dipoles and requiring each bead to have custom nonbonded parameters. As a result, it provides a more rigorous physical description of the system, but is more difficult to use. It currently runs only in the LAMMPS molecular dynamics simulator [19], since necessary support for point dipoles does not exist in other packages (e.g. AMBER, GROMACS). It may be possible to port ELBA to other packages in the future either by replacing the point dipoles with a Drude or shell model or by implementing point dipoles and rotational integration, but this would be a major task of its own. The complexity of the electrostatics model, and due to LAMMPS being the only supported simulator, means that equivalent simulation is significantly slower with ELBA as compared to MARTINI.

Inter-molecular interactions in ELBA are modelled with the Stockmayer potential [20] using a shifted force Lennard-Jones and switching dipole-dipole function.

$$U_{ij} = U_{ij}^{LJ} + U_{ij}^{\mu\mu} \tag{1.15}$$

$$U_{ij}^{LJ} = 4\epsilon \left\{ \left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right] + \left[6\left(\frac{\sigma}{r_c}\right)^{12} - 3\left(\frac{\sigma}{r_c}\right)^{6} \right] \left(\frac{r_{ij}}{r_c}\right)^{2} - 7\left(\frac{\sigma}{r_c}\right)^{12} + 4\left(\frac{\sigma}{r_c}\right)^{6} \right\} \tag{1.16}$$

Because of its rigorous electrostatics one of the major strengths of the ELBA force field is an accurate description of highly polar molecules such as water with many calculated water properties being comparable, or sometimes better, in accuracy to common atomistic force fields such as SPC and TIP3P [21]. ELBA is currently used for simulations of lipid bilayers [22] and is compatible with the AMBER force field in dual resolution simulations [23] [24]. Dual-resolution simulations allow an increase in simulation efficiency by coarse-graining the environment while maintaining an atomistic representation of the molecule of interest, typically a protein.

1.2 Sugars

Sugars are a significant component of biomass and have large structural diversity, but are relatively under-represented in the chemical literature. Their absence from many of the common forcefields, both AA and CG, despite approximately 50% of proteins having bound sugars, makes them an interesting target for study.

Glucose, the most common of the hexose monosaccharides is well characterised, but its isomers are not so well studied. There are two major classes of sugars: pentoses and hexoses, the five carbon and six carbon sugars respectively. Each of these classes is split into aldo- and keto-sugars, dependent on the position of the carboxyl C=O in the open chain form.

Of primary interest in this work (and biochemical simulation in general) are the aldo-hexoses in pyranose form. The aldo-hexoses, consisting of a six carbon chain ending in an aldehyde, are able to cyclise as a hemiacetal to form either a six- or five-membered ring. The five-membered form is a furanose, while the six-membered is a pyranose. The relative proportion of these forms is dependent on the stereoisomer, but the pyranose form is always favoured according to NMR studies [25].

The open-chain form of these sugars contains four chiral centres, giving a total of 16 stereoisomers, but an extra chiral centre is formed when in pyranose form, giving 32 steroisomers. The 16 D isomers are shown in fig. 1.3; the L isomers are the enantiomers of these, but are very rarely seen in nature. Sugars are named based on the position of the hydroxyl groups at carbons two, three and four 1.2. The position of the hydroxyl at carbon five determines whether the sugar is D or L, while the position of the stereocentre formed at carbon one determines α or β anomers. In aqueous solution the α and β anomers are in equilibrium through a process known at mutarotation which involves ring opening, bond rotation and a re-closing of the ring.

The conformational space of a single monosaccharide is complex and can be most formally expressed as a pair of angles in spherical coordinates known as the Cremer-Pople angles [26] where certain points on the sphere correspond to the 38 IUPAC (International Union of Pure and Applied Chemistry) named conformations. The conformations are split among classes: chair (C), envelope (E), half-chair (H), skew or twist-boat (S), and boat (B). In all of the aldo-hexo-pyranoses one of the chair forms is the dominant conformation in solution, in the crystal and in vacuum. In solution, some stereoisomers are able to ring flip between the two boat conformations named ${}^4C_1$ and ${}^1C_4$.

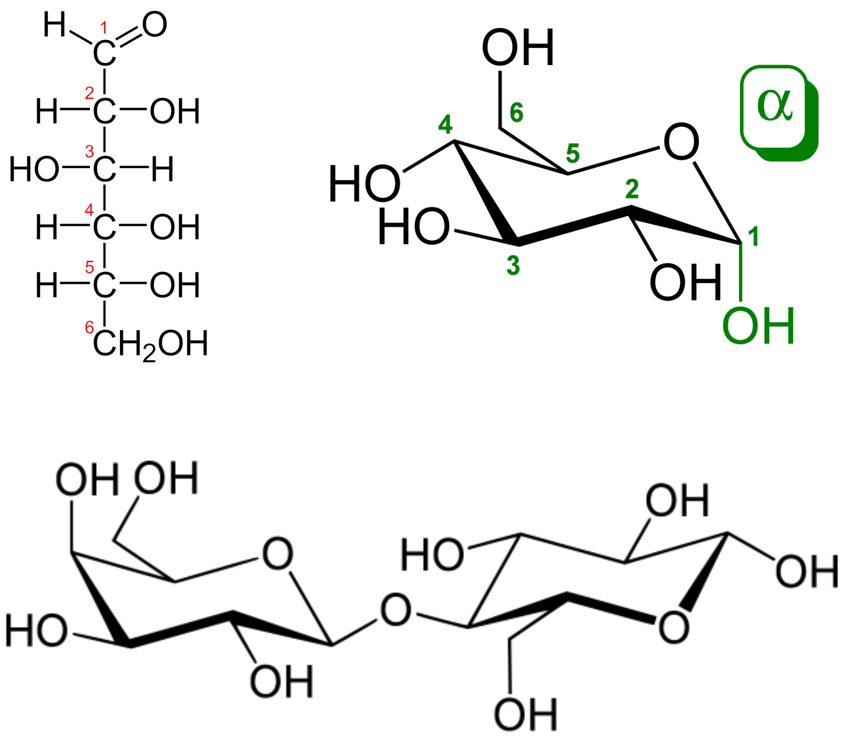


FIGURE 1.2: Top: Open chain and pyranose ring forms of (α)-D-glucose showing numbering convention. Bottom: The disaccharide lactose. Images taken from `https://en.wikipedia.org/wiki/Glucose` and `https://en.wikipedia.org/wiki/Lactose`.

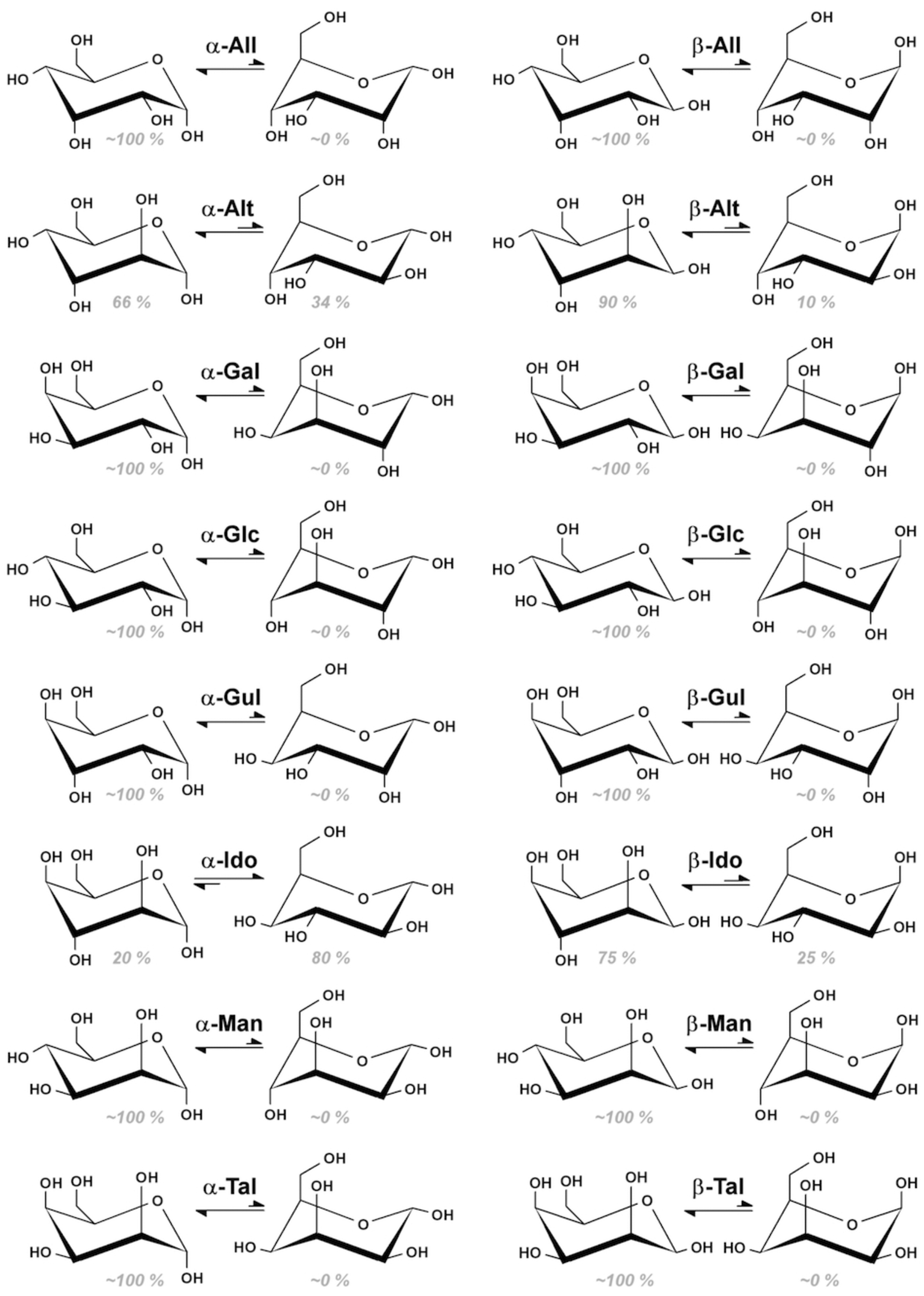


FIGURE 1.3: The 16 D-aldohexopyranoses and their proportions of $^4C_1$ vs $^1C_4$ forms from NMR experiment [27].

Atoms in the sugar ring are numbered following the IUPAC convention starting from the aldehyde carbon in the open-chain conformation which becomes the methoxy adjacent to the ring oxygen in the cyclic forms. Oxygen atoms are assigned the same number as the carbon atom to which they are bonded, with the ring oxygen being O5. Linkages between sugar rings are known as 'glycosidic bonds' and are formally defined by the carbon atoms on either side of the bridging oxygen, so a linkage between the C4 atom of one ring and the C1 atom of another is a $1 \rightarrow 4$ bond. The two dihedral angles in a glycosidic bond are given the Greek letters φ and ψ similarly to the two dihedral angles describing the linkage between two amino acids. Fig. 1.2 shows the disaccharide lactose, containing a $1 \rightarrow 4$ glycosidic bond between the two rings.

Sugars are often encountered in nature as part of oligo- or polysaccharides, long chains of monosaccharides, either linear or branched. One of the major components of biomass is starch, a mixture of amylose and amylopectin, both of which are formed from repeating glucose subunits. The linear polymer amylose is formed entirely from α-D-glucose linked by $1 \rightarrow 4$ glycosidic bonds. Amylopectin is slightly more complex with a branching structure of $1 \rightarrow 6$ bonds linking together amylose chains.

1.2.1 Atomistic Models of Sugars

A number of atomistic models of the hexose sugars exist, some of which are parameters taken directly from general purpose forcefields such as GAFF and some which have been parametrised specifically for sugars [28]. Of these various forcefields, among the most commonly used are GLYCAM [29], based upon AMBER, and GROMOS 53A6$_{\text{GLYC}}$ [27], based upon GROMOS.

The most recent major version of GLYCAM, GLYCAM06 released in 2008, was a significant re-parametrisation of the previous versions intended to improve the accuracy of the force field across a wide range of carbohydrates and gained the ability to describe small organic molecules other than carbohydrates. The new model removed the previous limitation with anomer-specific atom types which prevented an accurate modelling of ring flips. The other major change was a focus on correct conformation of hydroxyl groups with respect to C−O bond rotation. The reference data for the parametrisation was energies and rotation profiles calculated at the B3LYP/6-31++G(2d,2p)//HF/6-31G* level (geometry optimisation using the Hartree Fock method with the 6-31G Gaussian basis set with polarisation functions on heavy atoms; single point energies using the B3LYP DFT method with the 6-31G basis set with diffuse and polarisation functions on all atoms).

The GROMOS 53A6$_{\text{GLYC}}$ force field is a re-parametrisation of the GROMOS 53A6 forcefield focussed on replicating ring puckering parameters, the conformation of hydroxyl groups and glycosidic bond parameters. It does not correctly replicate the behaviour of ring flips as no ring flips were observed during microsecond simulations for any sugar. Both altrose and idose, in α and β forms are present in detectable concentrations of $^4C_1$ and $^1C_4$ in NMR experiments as referenced in the GROMOS 53A6$_{\text{GLYC}}$ paper.

1.2.2 Coarse-Grained Models

There are fewer coarse-grained parametrisations of sugars, and most existing models provide parameters only for glucose. Again MARTINI has a relatively high usage share in the coarse-grained simulation of carbohydrates [14], with an extension

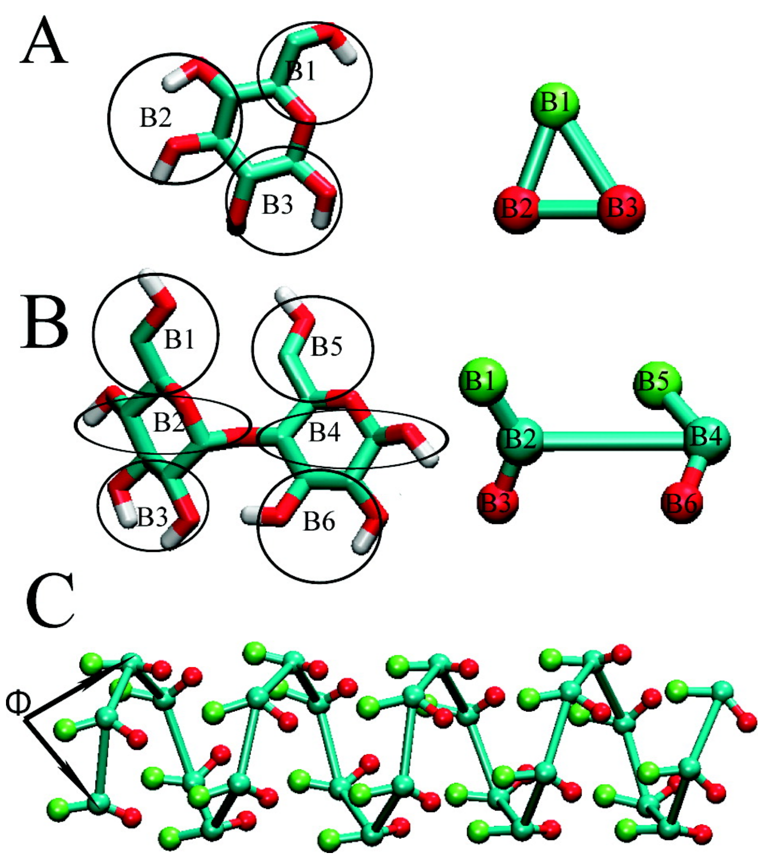


FIGURE 1.4: Coarse-grain mappings used for sugars in the MARTINI model [14].

to glycolipids [30]. Other competing forcefields are PITOMBA [31] and M3B [32].

Both the MARTINI and M3B forcefields use a mapping with three CG beads per ring, but different topologies. For monosaccharides MARTINI uses a triangular mapping, with all three beads being approximately in the plane of the ring, but for polysaccharides uses a linear mapping for each monosaccharide, with glycosidic bonds between the central atoms as in fig 1.4. It is claimed that this linear mapping better replicates the properties and dynamics of amylose and amylopectin polymers; long, linear chains of α-glucose with $1 \rightarrow 4$ glycosidic bonds. The structure of this model does however limit its general application and parametrisations of sugar-containing molecules in MARTINI often use their own custom parameters for the sugar components.

The M3B forcefield uses a triangular mapping for both mono- and polysaccharides, but with beads placed directly on the ring atoms C1, C4 and C6. This allows M3B to be more easily extended to treat glycosidic bonds other than $1 \rightarrow 4$ (although the original paper did not include them) and also allows a relatively easy backward mapping from CG to atomistic, quoting an RMSD of 0.34 Å for a small glucose oligosaccharide. This low RMSD suggests that little information is lost when converting between granularities of representation.

PITOMBA is based upon the GROMOS 53A6$_{\text{GLYC}}$ forcefield, being developed by the same research group, and uses a mapping of four CG beads per ring. This allows it a greater degree of conformational flexibility, as it is not constrained to being planar, and the ability to describe more glycosidic bond types.

By reducing the number of particles in the ring, none of these models are able to properly replicate the conformations accessible to an atomistic model. Additionally, these models do not allow for the full range of glycosidic bonds seen in complex sugar-containing molecules such as lipopolysaccharides. Although M3B is expected to perform better than the others in these cases, these less common glycosidic bonds are not yet part of the model. Finally, these models do not represent the differences between monosaccharides as they have no means to distinguish a hydroxyl group in the 'up' position from one in the 'down' position.

1.2.3 Lipopolysaccharide

Membrane lipids are a diverse class of bio-organic molecules present as a major component of cell membranes. As such they are responsible for the separation of intra- and extra-cellular components and comprise the first line of defence for an individual cell. Lipids are generally composed of a long hydrophobic 'tail' and and small hydrophilic 'head' group, each of which have many alternatives, allowing a wide range of different lipids. Within the membrane of a single cell, many types of lipids may be present in differing concentrations [33].

One class of membrane lipid is lipopolysaccharide (LPS), which is present only in the outer membranes of Gram negative bacteria. LPS can be widely varied, but the template consists of 5-7 hydrophobic tails bound to a disaccharide comprising the 'lipid A' region, a complex sugar 'core' containing several unusual sugar moieties, and a long glucose polysaccharide 'O antigen' [34] [35]. Lipid A is responsible for anchoring the molecule into the membrane, while the O antigen is involved in cell defence and recognition.

1.2.4 Motivations for this Work

Coarse-grained molecular dynamics models are essential for simulations to study molecular processes and interactions past the microsecond scale, yet there are currently no satisfactory coarse-grained models for sugars. Existing models fail to properly describe both the conformational flexibility within a single sugar ring, and the full range of glycosidic bond permutations present in complex molecules such as lipopolysaccharides. These limitations are a result of these models in general being parametrised only for simple glucose polymers. The quality of even atomistic sugar models is below that expected of other classes of biomolecule.

The original intention of this research was to develop a more accurate and more general coarse-grained model of sugars. This ELBA sugar model would have been combined with the existing ELBA lipid models to form a more complex lipopolysaccharide model.

During the project it became clear that the more useful output would be a set of tools to aid in the production of CG models more generally, as the MARTINI model has become the *de facto* standard in CG biomolecular simulation. This strand of work became the main focus and the resulting software, PyCGTOOL, was published in 2017 [37]. PyCGTOOL is available at `https://github.com/jag1g13/pycgtool` as doi:10.5281/zenodo.569555.

The following chapters describe the simulations performed (Chapter 2) and progress made towards an ELBA model of sugars (Chapter 3). Chapter 4 describes the software tool produced to simplify the process of CG model generation (chapter is the PyCGTOOL paper with some modification).

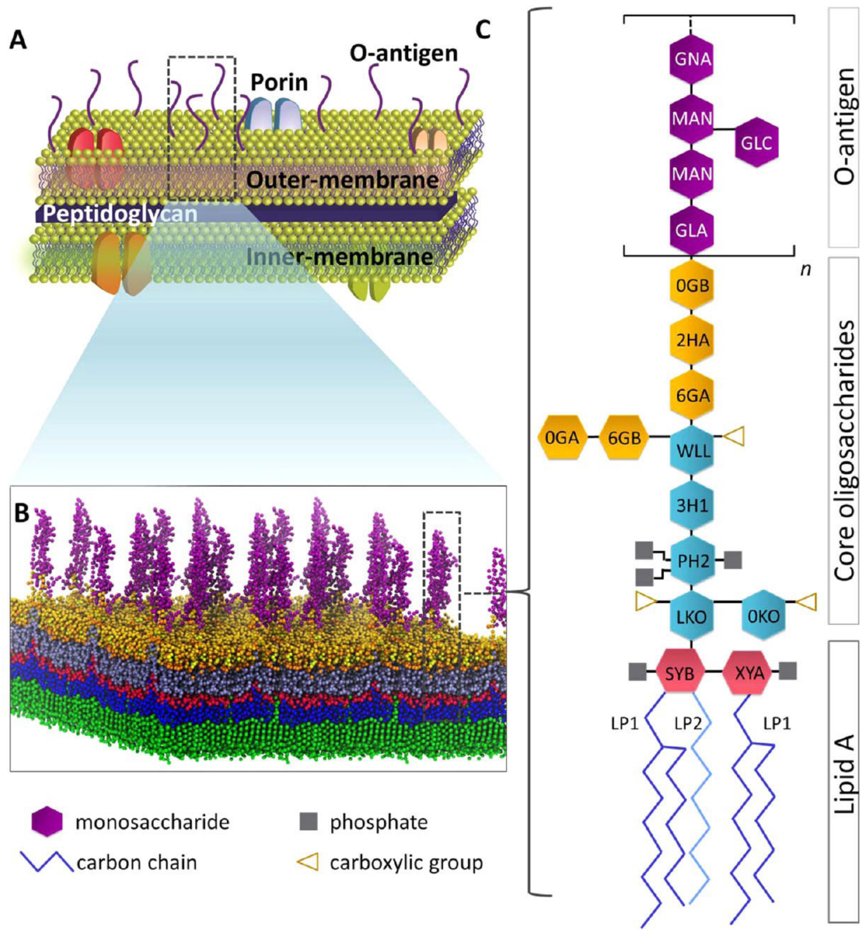


FIGURE 1.5: Lipopolysaccharide structure and location in the Gram-negative cell [36].

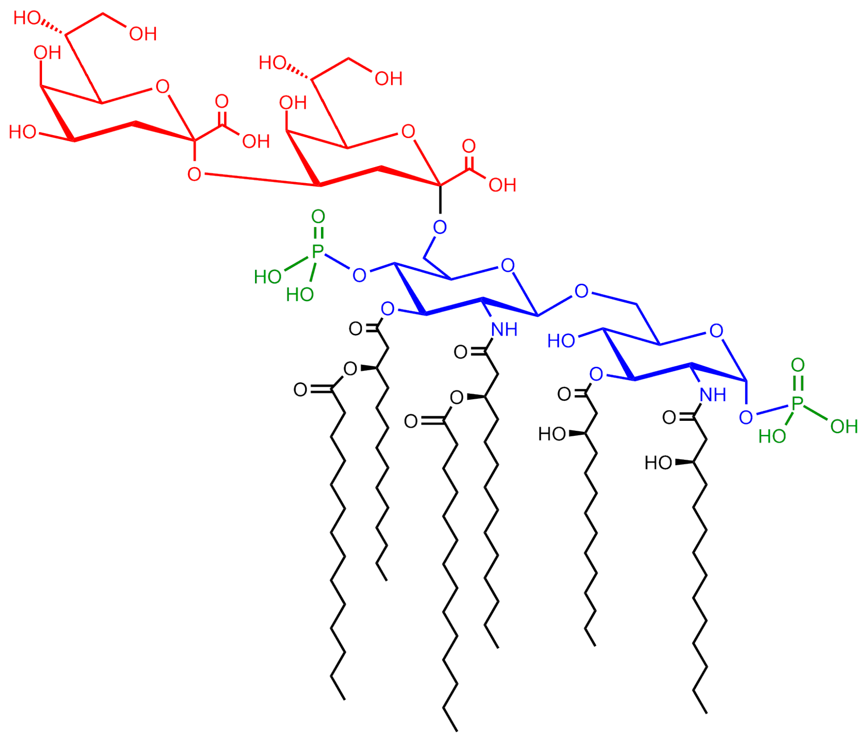


FIGURE 1.6: LPS core region showing non-standard KDO sugars (red). Image taken from https://en.wikipedia.org/wiki/Lipopolysaccharide.

Chapter 2

Atomistic Simulations

In order to parametrise a CG model of sugars, a wide basis of atomistic simulation is required to use as a source both for parametrisation and for validation.

All simulations were performed at typical biological temperatures and pressures; 310 K and 1 atm.

Before each simulation, the system was energy minimised and equilibrated under NVT then NPT ensembles. Equilibration under NVT used the Berendsen thermostat; the Berendsen barostat was added for NPT equilibration. The equilibrated simulation was then run under the v-rescale thermostat and the Parinello-Rahman barostat, both of which provide better energy conservation than the Berendsen equivalent, but are not suitable for equilibration due to instability when far from equilibrium.

Electrostatics were calculated using Particle Mesh Ewald summation (PME), and Van der Waals forces were calculated with a cutoff set the the recommended value for each forcefield.

2.1 Simulation of Small Organics

A range of small organic molecules were simulated to aid in the parametrisation of the ELBA sugar model. Of the table below, methanol and ethanol simulations were used to fit non-bonded parameters of the individual CG beads. Cyclohexane was used as an analogue of the monosaccharides, removing the polar groups, in order to compare conformational behaviour. Atenolol is a β_1 antagonist drug which was used as one of the test cases for automated generation of CG models with PyCGTOOL.

The aqueous simulations consisted of a single molecule, solvated with water to the specified concentration. The size of the solvent box, and thus the number of solvent molecules and concentration, was in each case set to be slightly over twice the short-range electrostatics cutoff distance.

The simulations of neat solvent were again defined by a box size of slightly over twice the short-range cutoff, with the number of molecules set to fill the box.

Forcefield	Molecules	Timescale	Conc. / Num.
GAFF	Methanol (Neat)	100 ns	1000
GAFF	Methanol (Aqueous)	100 ns	0.16 M
GROMOS 53A6	Ethanol (Neat)	100 ns	512
GAFF	Ethanol (Neat)	100 ns	512
GAFF	Ethanol (Aqueous)	500 ns	0.125 M
GAFF	Cyclohexane (Neat)	100 ns	532
GROMOS 54A7	Atenolol (Aqueous)	50 ns	0.045 M

TABLE 2.1: Simulations of small organics

2.2 Simulation of Sugars

2.2.1 Monosaccharides

Each of the monosacharides (with the exception of IDOA / IDOB which do not have parameters in the GLYCAM06 forcefield) was simulated with a single monosaccharide solvated in water with a box size of slightly larger than twice the short-range cutoff. This resulted in a concentration of approximately 0.15 M in each case.

Forcefield	Sugar	Simulation Length
GLYCAM	ALLA	100 ns
GLYCAM	ALLB	100 ns
GLYCAM	ALTA	100 ns
GLYCAM	ALTB	100 ns
GLYCAM	GALA	100 ns
GLYCAM	GALB	100 ns
GLYCAM	GLCA	100 ns
GLYCAM	GLCB	100 ns
GLYCAM	GULA	100 ns
GLYCAM	GULB	100 ns
GLYCAM	MANA	100 ns
GLYCAM	MANB	100 ns
GLYCAM	TALA	100 ns
GLYCAM	TALB	100 ns

TABLE 2.2: Simulations of monosaccharides

2.2.2 Disaccharides

Forcefield	Disaccharide	Simulation Length
GLYCAM	GLCA1-3ALLA	1 µs
GLYCAM	GLCB1-3ALLA	1 µs
GLYCAM	GLCA1-4GLCA	1 µs
GLYCAM	GLCB1-4GLCB	1 µs
GLYCAM	GLCA1-6GLCA	1 µs
GLYCAM	GLCB1-6GLCA	1 µs

TABLE 2.3: Simulations of disaccharides

2.3 Analysis of Glycosidic Angles

2.3.1 Circular Statistics

In order to compare distributions of measurements it is typical to plot histograms. Histograms, however, often provide a poor representation of the underlying probability distribution and do not aid in comparison of multiple probability distributions, either visually or mathematically.

The more rigorous alternative to histograms is Kernel Density Estimation (KDE). When performing a KDE analysis, instead of counting data points into bins, as with histograms, the underlying probability distribution is estimated by summing individual probability kernels for each data point. Typically, each data point is replaced with a normal distribution (equation 2.1), and these normal distributions summed over the coordinates of interest as in equation 2.2; where K is the probability kernel function, and h is the 'bandwidth' parameter. mazon

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

$$f_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.2}$$

KDEs are not limited to application in one dimension, and form the mathematical basis of metadynamics in exploring a complex multi-dimensional energy surface. Nor are KDEs limited to using the normal distribution as their probability kernel. By using the von Mises distibution (equation 2.3) as the probability kernel, it is possible to produce a KDE of data points distributed around a circle, such as encountered when measuring dihedral angles. Like the normal distribution, the von Mises distribution has two parameters: μ, the centre of the distribution, analogous to μ of the normal distribution and κ, the measure of concentration, analogous to $1/\sigma^2$.

$$f(x) = \frac{e^{\kappa \cos x - \mu}}{2\pi I_0(\kappa)} \tag{2.3}$$

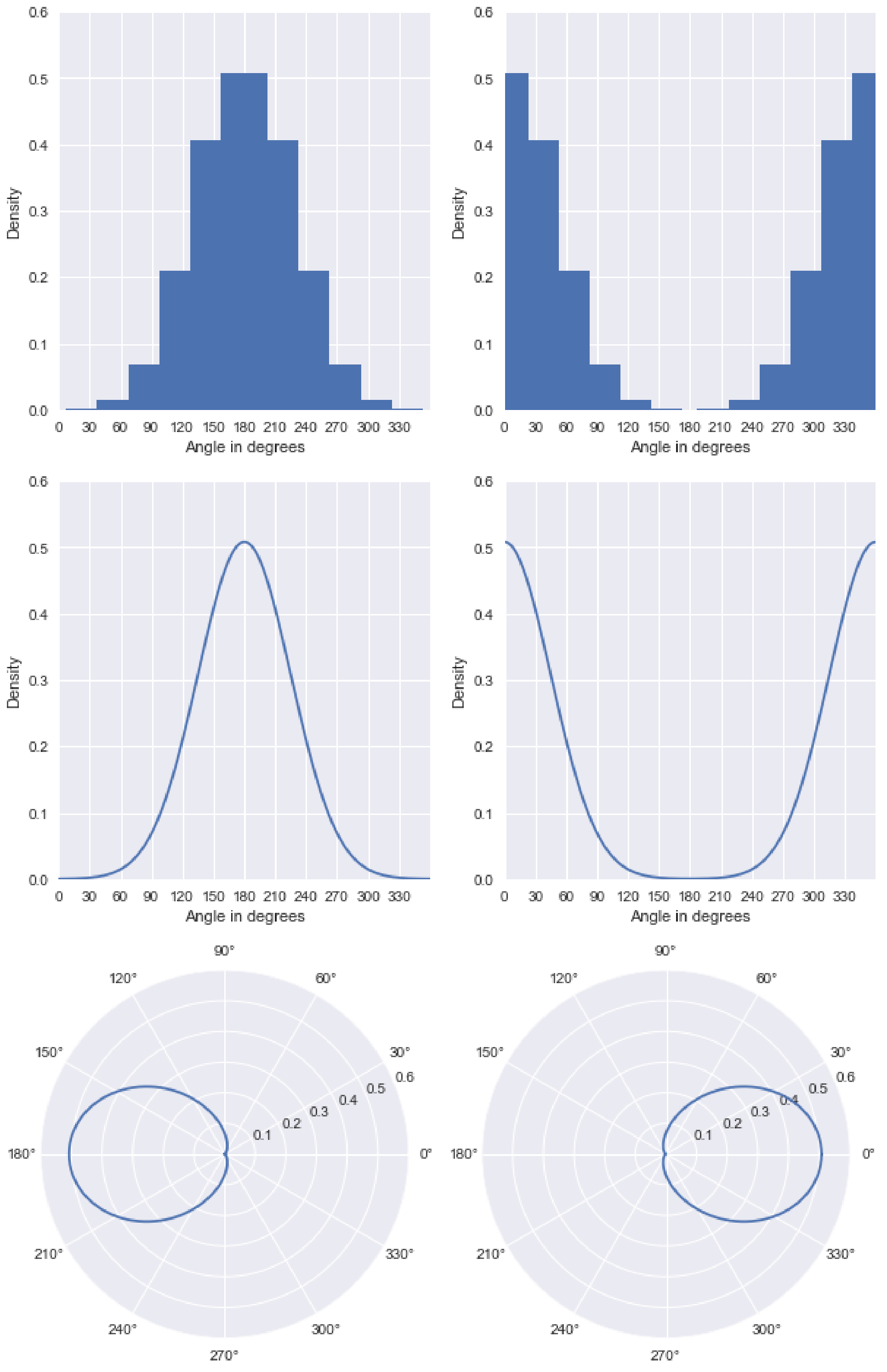


FIGURE 2.1: The three plots in each column show the same sample distribution. In the top row the samples have been binned into a histogram, potentially losing the detail of the underlying distribution. The second row shows a KDE using the von Mises kernel, but plot on a linear axis which breaks the distribution across the boundary of 2π. The bottom row shows the KDE plot on circular axes, avoiding the break in the distribution seen previously.

Chapter 3

Model Parametrisation

3.1 Introduction

It was decided to implement a coarse-grained model of carbohydrates using the ELBA model since a high quality description of electrostatics, and thus interaction with polar solvents such as water, is known to be important to accurately model their behaviour [38]. In addition to this, the model would act as a component of an LPS model, requiring more monosaccharides to be parametrised than is usually seen, with many force fields providing parameters for just glucose. The model should be able to distinguish between monosaccharides, both by electrostatic properties and by conformational properties; i.e. relative proportion of $^{4}C_{1}$ vs $^{1}C_{4}$.

In order to fulfil these goals the model would consist of six beads per monosaccharide, allowing full conformational flexibility and the ability to provide parameters for all possible types of glycosidic linkage. Multiple mappings and bead positions were tested, but the mapping which proved most convenient was simply to place the CG bead directly on the ring atom from the atomistic representation, as shown in figure 3.1. Previous attempted mappings placed the beads at the geometric centre of the atoms they represented which gave a better space-filling representation, but resulted in bimodal distributions of bond lengths and angles as the ring flipped.

The mapping resulted in four equivalent beads (C1, C2, C3, C4) approximated as methanol, one bead (C5) approximated as ethanol, and one bead (O5) containing a single oxygen atom for which the ester oxygen parameters were copied from GAFF. To distinguish between sugars different dipoles and ring dihedral angles would be set.

Parametrisation of the sugar model was performed in several steps: first single beads were parametrised against their atomistic equivalents in isolation, then the beads were combined and parameters tuned with respect to the CG solvent model, finally bonded parameters and dipole angles were tuned.

3.2 Available Data

In order to parametrise a model there must be sufficient data of a level to parametrise against. For atomistic models this is usually quantum mechanical calculations and experimental data; for coarse-grained models the reference data usually comes from atomistic simulations and experiment, although quantum mechanical calculations may still be useful.

In the particular case of sugars there is a relatively large amount of reference data for glucose, but little for the less common sugars. Both the GLYCAM and GROMOS 53A6_GLYC atomistic models provide parameters for the majority of monosaccharides (GLYCAM does not include IDOA and IDOB), but these models both have

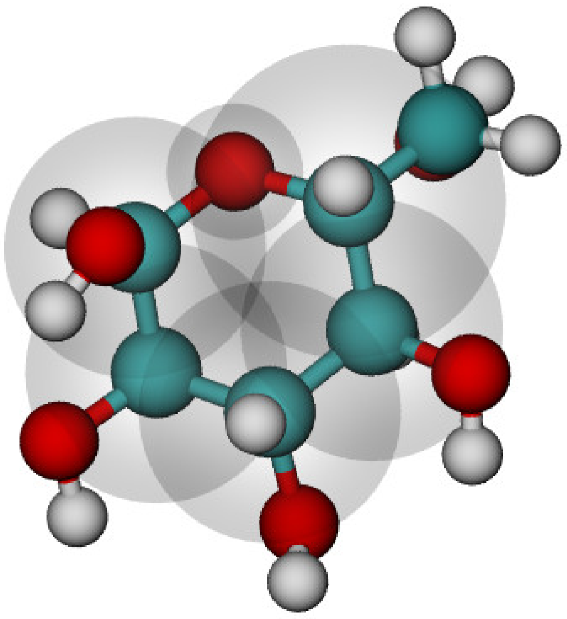


FIGURE 3.1: The six bead per ring mapping used for the model parametrisation.

significant limitations discussed previously. As such these atomistic models are suitable for initial parameter generation, but refinement will have to include data from other sources.

Conformational populations are available from NMR studies collected in the GROMOS 53A6_GLYC paper [27] and shown in fig. 1.3. The values however are averaged over NMR timescales and do not provide data on the energy barrier between conformations or their residence time.

Experimental values for densities of methanol-water [39] and ethanol-water [40] were used in the parametrisation of single bead solvent models. Densities of aqueous glucose solutions were measured by Comesana [41].

Measurements of the dipole of glucose in aqueous solution were made by [42], but no similar data could be found for the other sugars. This value includes contributions from both α and β glucose, but is in reasonable agreement with simulations performed using the GLYCAM atomistic model, so values from GLYCAM simulations must be considered acceptable in the absence of experimental data.

3.3 Lennard-Jones Terms

One of the common methodologies in parametrising a coarse-grained force field is to begin with parametrisation single beads against their hydrogen-capped atomistic equivalents [12]. In the six bead sugar model it is most sensible to consider the beads to be methanol, ethanol and oxygen/water.

Simulations of pure liquids methanol and ethanol were performed using GAFF (General AMBER Force Field) in GROMACS, using ACPYPE [43] for conversion, and Radial Distribution Functions (RDFs) calculated. The initial estimate for the Lennard-Jones σ parameter was taken as the first maximum of the atomistic RDF and the ε parameter was taken to be the same as that of water.

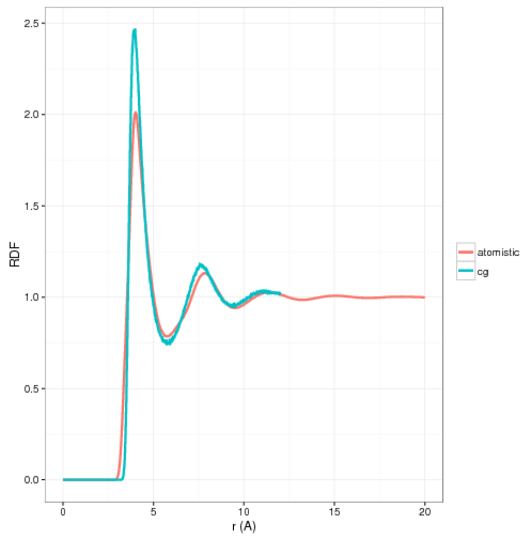


FIGURE 3.2: Overlaid RDFs of methanol from AA reference simulation and CG simulation.

These parameters were used in a single bead coarse grained model with the respective experimental dipoles in a CG simulation from which another RDF was calculated. The σ and ε parameters were iteratively refined until the atomistic and CG RDFs were sufficiently similar. The σ parameter was considered optimal when the CG and AA RDF graphs showed the greatest degree of overlap in the x-axis, and ε when the greatest similarity was seen in the y-axis. These models were further refined by comparing free energies of aqueous solvation and densities to experimental reference values [40]. After these parameters had been determined close to optimal, fine tuning was performed by Marley Samways.

The ELBA forcefield uses the Lorentz-Berthelot combination rules for ε and σ values for interaction between different bead types. These rules use the geometric average of the individual ε parameters and the arithmetic average of the σ parameters as in eqns. 3.1 & 3.2. Using the combination rules as here, free energy calculations indicated that the interaction between the sugar and water was incorrect. Since in atomistic simulations and in nature the sugar is able to form hydrogen bonds with the surrounding water, the strength of the Lennard-Jones interaction needs to be scaled up. This has been done in ELBA previously using a modified version of the Lorentz-Bethelot combination for ε which includes a hydrogen bond scaling parameter, h (eqn. 3.3). Each bead pair has an h constant representing the degree of hydrogen bonding. Values used in the most recent ELBA paper range from $h = 1$ for pairs with no hydrogen bonding to $h = 1.5$ for the water-water interaction. A script was created by Marley Samways to aid in performing bead combinations manually [44]

Interactions between these beads and water were parametrised by comparison to atomistic simulations of a single solute molecule in a box of water. Initially RDFs (see fig 3.2) were compared, then density and free energy. As the parameters were modified, the R^2 deviation between the distribution produced from the CG simulations and the AA reference was measured. In this case, the σ and ε parameters were fixed, but the h-constants for 'bead-bead' and 'bead-water' were varied. The resulting parameters were then tuned by comparing the density of mixtures to experimental reference data [40].

Number	Type	$\varepsilon\prime$	σ
1	H_2O	0.367	3.050
2	MeOH	0.536	3.725
3	EtOH	0.592	4.200
4	O	0.170	1.684

TABLE 3.1: Table of $\varepsilon\prime$ and σ for each bead type. Units of $\varepsilon\prime$ are kJ mol^{-1} and units of σ are Å.

Bead	1	2	3	4
1	3.050	3.388	3.625	2.367
2		3.725	3.963	2.705
3			4.200	2.942
4				1.684

TABLE 3.2: Table of all bead σ combinations. Units of σ are Å.

Bead	1	2	3	4
1	0.550	0.621	0.652	0.262
2		0.670	0.704	0.317
3			0.740	0.333
4				0.170

TABLE 3.3: Table of all bead ε combinations. Units of ε are kJ mol^{-1}.

Bead	1	2	3	4
1	1.5	1.4	1.4	1.05
2		1.25	1.25	1.05
3			1.25	1.05
4				1.00

TABLE 3.4: Table of all bead h-value combinations. H-values are dimensionless.

$$\epsilon_{ij} = \sqrt{\epsilon_i \, \epsilon_j} \tag{3.1}$$

$$\sigma_{ij} = \frac{\sigma_i + \sigma_j}{2} \tag{3.2}$$

$$\epsilon_{ij} = h_{ij} \sqrt{\epsilon_{i'} \, \epsilon_{j'}} \tag{3.3}$$

The individual h values were estimated by analogy with existing beads in the ELBA forcefield for lipids then iteratively tuned by measuring the density of solvent mixtures.

3.4 Bonded Terms

Bonded terms within a single monosaccharide was calculated by Boltzmann Inversion using the program CGTOOL; the predecessor to PyCGTOOL, discussed in chapter 4. Atomistic simulations were performed in GROMACS versions 5.0.4 to 2016 using the GLYCAM06 force field with parameters from the GLYCAM carbohydrate builder and converted using ACPYPE. Simulation systems consisted of a single monosaccharide solvated with an 8 Å TIP3P water buffer at one atmosphere of pressure and at a temperature of 310 K. Electrostatic forces were calculated with PME at a cutoff of 8 Å, with a shifted-force potential using the same cutoff for Van der Waals forces.

Analysis of these atomistic trajectories with CGTOOL gave equilibrium values and force constants for all bond lengths, angles and dihedral angles. The calculated terms for bond length and angles were used without modification throughout development of the coarse-grained model, but dihedral angle terms were used as initial estimates and later modified to give the flexibility required for ring flips to occur. CGTOOL also gave bead charges, calculated simply by summing the charges of the component atoms. It was originally intended that these charges be replaced with charges taken from QM calculations, but since these failed to adequately replicate the molecular dipole of glucose (see 3.5) they were not used.

3.4.1 Dihedral Angles

While bond lengths and angles were taken directly from CGTOOL output without modification, the calculated dihedral angles within the sugar ring were not suitable. One of the aims of the model is to accurately represent the ring-flip behaviour of each monosaccharide, while CGTOOL is limited in dihedrals to replicating only a single conformation or an average over multiple conformations.

Initial simulations were performed using the dihedrals angles taken directly from CGTOOL, but these were replaced by a Fourier series consisting of a period three cosine to produce minima 120° apart overlaid with a period one cosine to provide an bias toward the experimentally observed conformation. The relative weights of the Fourier terms to replicate NMR conformational biases were determined by another student in the group.

3.5 Dipoles

Since the main distinguishing feature of the ELBA forcefield is an accurate representation of electrostatics, it is important to have high quality dipole parameters.

3.5.1 Analysis

Bead dipoles were calculated using two strategies: from atomistic MD simulations and from individual geometry optimised QM structures.

Classical

Classical dipoles were calculated from the all-atom simulations using GLYCAM. Initial analysis was performed using the GROMACS tool `gmx dipole` which calculates dipoles of molecules or fragments from atomic charges. Molecular dipoles measured in this way are shown in table 3.5.

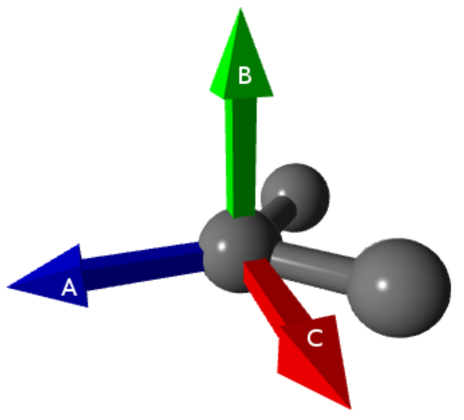
In order to place dipoles onto the coarse-grain beads, individual bead dipoles and their relative orientations were required. Firstly, all-atom trajectories from GROMACS were pseudo-coarse-grained and exported into ELBA LAMMPS format which supports particle dipoles. Dipoles for each bead were calculated from atomic charges using the same algorithm as `gmx dipole`. For charged beads a counter-charge is added to each of the component atoms weighted by mass to ensure all beads are neutral. The weighting of the counter-charge is arbitrary, but mass weighting was chosen so that the counter-charge would reside toward the centre of the bead on the carbon atom. Once beads have been made neutral, bead dipoles can be calculated in the usual manner. This manipulation is shown in 3.4 and is required since the dipole of set of particles with net charge is not well defined: the dipole of a charged group is dependent on the reference point from which it is measured. By setting the total charge to zero dipoles may be calculated normally, giving the same value regardless of the reference point. For convenience this reference point is almost always chosen to be the coordinate origin.

$$\mu_B = \sum_{i \in B} \mathbf{x}_i \left(q_i - \frac{q_B m_i}{m_B} \right) \tag{3.4}$$

From the resulting pseudo-CG trajectory, dipole orientations were calculated relative to the ring. In order to fully capture the range of motion of the dipoles a three dimensional representation was required. The chosen coordinates were the spherical polar representation relative to the plane formed with the two neighbouring beads as the polar angle (θ) and the angle relative to the neighbour in the positive direction as the azimuthal angle (ϕ).

By coarse-graining a GLYCAM simulation trajectory from GROMACS to LAMMPS format using CGTOOL with dipoles calculated from atomic charges, dipoles at the pseudo-coarse-grain level could be visualised. A Python program was created to perform analysis of dipoles by angles relative to the plane of the ring [45]. A second Python program was created by Marley Samways to allow visualisation of dipoles in VMD by converting the point dipole into two reference particles [44]. This allowed both the direction and magnitude of the dipoles to be easily seen, as well as allowing the VMD measuring tools to be used for analysis.

These analysis tools showed that bead dipoles trace circles in the space of the two measured angles; elevation above the plane and angle with the adjacent bond (azimuthal angle). This dipole rotation corresponded to rotation of the hydroxyl groups



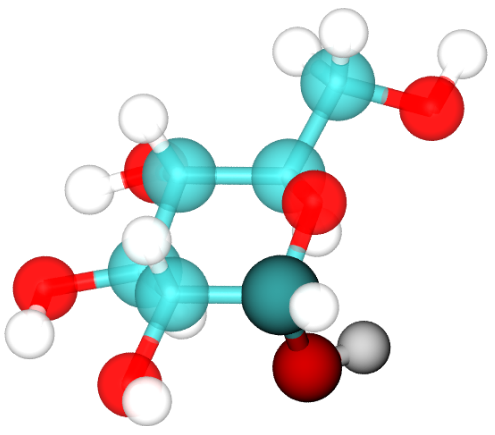


FIGURE 3.3: Left: The polar angle θ is the angle between the dipole and the 'up' axis B; the azimuthal angle ϕ is the angle between the dipole and the bond vector joining the reference atom and the next atom in the ring. Right: Rotation of a hydroxy group causes the bead dipole to precess in a circle/cone.

about the C$-$O bond. Although the exact shape of the distribution varied between beads, with distinct minima being seen around the circle, it was judged that a circle in the two dimensional space would be a sufficiently accurate approximation while not overly complex to implement. These circles were of the same radius, but the circle centre was at a different angle of elevation depending on whether the hydroxyl group was axial or equatorial, and in an 'up' or 'down' position. An example is shown in figure 3.4 for a simulation of α-glucose.

Quantum Mechanical

QM dipoles were calculated using the programs Gaussian09, GDMA and Mulfit. Initial calculations were performed using the PBE functional with the 6-311+G** basis set (as recommended by [46]), test calculations were performed using HF 6-31G, before returning to PBE 6-311+G** with the PCM implicit solvent model.

From the geometry optimisation performed in Gaussian, the electron density was reduced to a set of bead multipoles using Distributed Multipole Analysis (DMA) with GDMA. These multipoles up to rank 4 (up to hexadecapoles) were used as input for Mulfit and reduced to rank 1 (monopoles and dipoles). Mulfit is a tool for multipole reduction which performs a least squares fit of the electric field around a molecule to replicate the field using multipoles of lower rank. This multipole reduction was performed by Prof. Chris Reynolds for several reduction schemes.

The bead monopoles (point charges) and dipoles fitted by Mulfit gave a very poor fit the the expected molecular dipole of over 12 Debye compared to the expected value of 4.9 Debye. As a result the charges and dipoles from QM calculations were never applied to the model, instead remaining with charges and dipoles calculated from the constituent atoms of the CG beads.

3.5.2 Functional Forms

The initial implementation of the model set dipoles at a fixed position relative to the plane of the ring, as this made control relatively simple. Since LAMMPS only

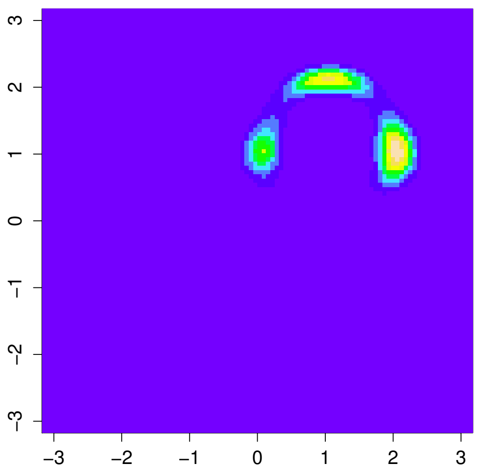


FIGURE 3.4: Orientation of C1 dipole in pseudo-coarse-grained trajectory for GLCA. The x axis is elevation above the plane in radians; the y axis is angle from the adjacent atom. The colour scale is from blue (low density) to red (high density).

Type	Molecular Dipole	SD
Experimental	4.9 ± 0.2	
GLYCAM fudge 0.5	4.37	1.65
GLYCAM fudge 1	4.30	1.64
GLYCAM GLCB vacuum	3.46	0.75
GLYCAM GLCB aq	4.30	1.56
PBE 6-311+G**	0.69 to 1.45	
Mulfit all atoms	0.64 to 0.87	
Mulfit carbon	12.04 to 12.86	

TABLE 3.5: Measured and calculated molecular dipoles in Debye. Mulfit ranges are the smallest and largest values obtained from testing a range of conformers. Mulfit values were collected by Prof. Christopher Reynolds. All calculations are for α-glucose unless otherwise specified.

Sugar	Dipole	SD	Sugar	Dipole	SD
ALLA	4.73	1.82	ALLB	4.59	1.68
ALTA	5.08	1.91	ALTB	4.51	1.88
GALA	4.15	1.59	GALB	4.42	1.74
GLCA	4.27	1.63	GLCB	4.26	1.56
GULA	4.45	1.89	GULB	4.78	1.87
IDOA	-	-	IDOB	-	-
MANA	4.48	1.67	MANB	4.25	1.67
TALA	4.40	1.78	TALB	4.34	1.70

TABLE 3.6: Calculated molecular dipoles using the GLYCAM model. Simulations were 100 ns and consisted of a single monosaccharide in a box of approximately 320 water molecules.

includes control of dipoles by a single angle (as previously used with the ELBA forcefield), a second control angle was required. For simplicity, the second angle was added as a dihedral (as `dihedral_dipole`), so that the position of the dipole would be specified in the same manner as the position of an atom.

Since this simple form was seen to be insufficient, a further new form of dipole control was added to LAMMPS as `improper_dipole_cone` which sets a cone at a certain elevation from the plane of the ring with a given internal angle. All points on the surface of the cone are of equal energy, but a harmonic restoring torque is applied along the radius. Throughout this development bead dipole magnitudes were fixed as the experimental dipole magnitudes of their atomistic equivalents; i.e. methanol/ethanol.

The style `improper_dipole_cone` is calculated as follows (visualisation in fig. 3.5): The atom to which the dipole is attached (atom 0) and the atoms to the right (atom 1) and left (atom 2) are passed into the function. The vectors between bonded pairs (eqn. 3.5) are used to define orthogonal local axes by calculating the normal to both vectors and the sum of both vectors, giving 'up' and 'out' vectors relative to the ring as shown in eqn. 3.6. The third axis is the cross product of these two vectors. A cone centre vector is created by rotating vector B around the axis C by a specified angle θ (eqn. 3.8) using rotation matrix eqn. 3.7. The angle γ between the dipole and the cone centre is calculated as eqn. 3.9 and a torque applied to the dipole along the radius of the cone, with force calculated from a cos-harmonic potential as eqn. 3.11.

$$\vec{r_1} = \mathbf{x}_0 - \mathbf{x}_1 \qquad \vec{r_2} = \mathbf{x}_0 - \mathbf{x}_2 \tag{3.5}$$

$$\vec{A} = \vec{r_1} + \vec{r_2} \qquad \vec{B} = \vec{r_1} \times \vec{r_2} \qquad \vec{C} = \vec{A} \times \vec{B} \tag{3.6}$$

$$R = \begin{pmatrix} \hat{C}_x^2(1-\cos\theta)+\cos\theta & \hat{C}_x\hat{C}_y(1-\cos\theta)-\hat{C}_z\sin\theta & \hat{C}_x\hat{C}_z(1-\cos\theta)+\hat{C}_y\sin\theta \\ \hat{C}_y\hat{C}_x(1-\cos\theta)+\hat{C}_z\sin\theta & \hat{C}_y^2(1-\cos\theta)+\cos\theta & \hat{C}_y\hat{C}_z(1-\cos\theta)-\hat{C}_x\sin\theta \\ \hat{C}_z\hat{C}_x(1-\cos\theta)-\hat{C}_y\sin\theta & \hat{C}_z\hat{C}_y(1-\sin\theta)+\hat{C}_x\sin\theta & \hat{C}_z^2(1-\cos\theta)+\cos\theta \end{pmatrix} \tag{3.7}$$

$$\vec{D} = \vec{A} \times R \qquad \vec{E} = \mu \times \vec{D} \tag{3.8}$$

$$\gamma = \operatorname{atan2}(\det(\mu, \vec{D}, \vec{E}),\ \mu \cdot \vec{D}) \tag{3.9}$$

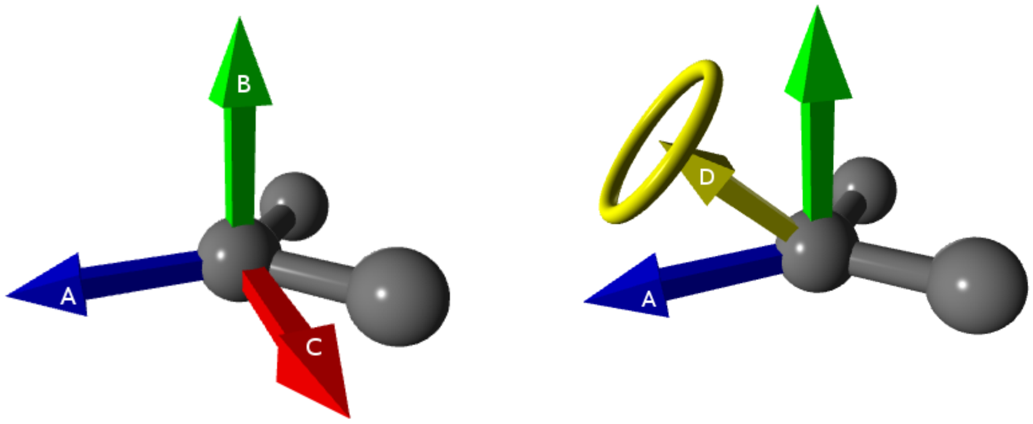


FIGURE 3.5: Visualisation of vectors use in the dipole control function. Vectors $\vec{A}$ and $\vec{B}$ are the reference 'out' and 'up' vectors. Vector $\vec{C}$ is the cross product of these used in the rotation matrix R where θ is the elevation of $\vec{D}$ above $\vec{A}$. Vector $\vec{D}$ is the cone centre. The ring around vector $\vec{D}$ represents the resultant range of free motion of the dipole constrained to this vector by the angle γ. The atom upon which these vectors are centred is 'atom 0'.

$$\det(a,b,c) = \begin{vmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{vmatrix} \tag{3.10}$$

$$\tau = 2k\,(\cos(\gamma) - \cos(\gamma_0))\,(\vec{D} \times \mu) \tag{3.11}$$

This functional form does not allow the cone centre to be rotated out of the plane containing the vectors A and B, although this could be added as a second rotation if required in the future. The dipoles calculated from atomistic simulations had cone centres in this plane so it is expected that this extension will not be required for the unsubstituted hexoses.

Investigation of energy conservation in a series of small test systems revealed an issue with the implementation of the original `angle_dipole` functional form whereby a dipolar particle would continually gain angular momentum if no rotational thermostat was applied. Since torque was applied to only a single atom, angular momentum was not conserved during the integration. This issue was fixed upstream by resolving an equal and opposite torque to forces on the adjacent atoms, in such a way as to conserve both angular and linear momentum. The additional functional forms implemented here have not yet been adapted.

Later, it became evident that this functional form for control of the dipoles did not move correctly when the sugar undergoes a ring flip, so needed to be replaced. The only alternative found which could correctly represent a ring flip, was for each bead to place a dummy bead above or below the plane of the ring, opposite the location of the hydroxy oxygen. This dummy bead was bound the the closest ring bead and to the two adjacent beads to fix its position. The dipole was then restrained to the vector between the ring bead and the dummy bead using the existing `angle_dipole` potential. While the dipole now behaved correctly during ring flips, the model now used 11 beads to represent a single monosaccharide; only six fewer particles than used in the united-atom representation.

3.6 Disaccharide Model

Once the monosaccharide model was sufficiently complete with only minor tuning remaining the first glycosidic terms could be added, allowing sugar oligo- or polymers to be modelled. With 16 different monosaccharides and five hydroxyl substituents per ring there are a total of 6400 glycosidic bond permutations if all were to be parametrised individually. Since this is infeasible, it was decided to reduce the number of permutations by considering only if the bond is in an ‘up’ or ‘down’ configuration and the bead types involved. This reduces the number of permutations to 16; two bead types on each end of the bond, each in one of two configuations.

Simulations of several linkage types for the ALTA-ALTA dimer indicate that the potential energy surface for rotation about the two glycosidic angles is relatively simple, often just a single minimum. However, it was not clear from just these simulations whether this single minimum was a true minimum, or an effect of kinetic trapping. To find the true energy landscape metadynamics simulations were run across a range of linkages.

Metadynamics is an enhanced sampling method in molecular dynamics whereby an energy penalty is applied to locations in the phase space which have already been visited, resulting in a much more complete exploration. This is carried out by defining one or more degrees of freedom to be ‘collective variables’ (CVs) which are used to define a subset of the complete phase space. As the simulation runs, at a set interval a Gaussian energy penalty is laid down at the current position in the reduced CV phase space. Since this penalty will tend to push the trajectory away from previously visited areas it acts to reduce the depth of wells in the potential energy surface and promote a more complete exploration of the phase space. A commonly used analogy is to compare metadynamics to pouring sand into energy wells.

Metadynamics simulations were performed using the PLUMED package version 2.2 [47] which integrates into several of the most commonly used MD simulators; here used with GROMACS 5.1.1. The glycosidic φ and ψ angles of the disaccharide were used as the two collective variables. Gaussians were deposited every 500 MD steps (every 1 ps) with a height of 0.12 kJ mol^{-1} and a width of 0.35 rad on a square grid ranging from $-\pi$ to π rad in accordance with recommendations from the authors of PLUMED.

Initial simulations of the dissaccharide ALLA1-4ALLA confirm that the single minimum seen in previous ALTA-ALTA simulations is reasonable. The potential energy landscape over the two glycosidic angles is shown in figure 3.6. Figure 3.7 shows the occupancy probability over the energy landscape using the Boltzmann probability distribution eq 3.12. This figure shows that it is reasonable to approximate the potential energy surface as a single minimum in the case of the ALLA1-4ALLA disaccharide, allowing a single, simple harmonic dihedral to be used in the CG representation.

$$p_i = \frac{e^{-\epsilon_i/kT}}{\sum_{i=1}^{M} e^{-\epsilon_i/kT}} \tag{3.12}$$

Four disaccharides were selected to represent the combinations of glycosidic bonds in ‘up’ and ‘down’ positions around the ring at each end and two to represent the two configurations of a 1-6 linkage. Each of these was simulated for 1 µs using

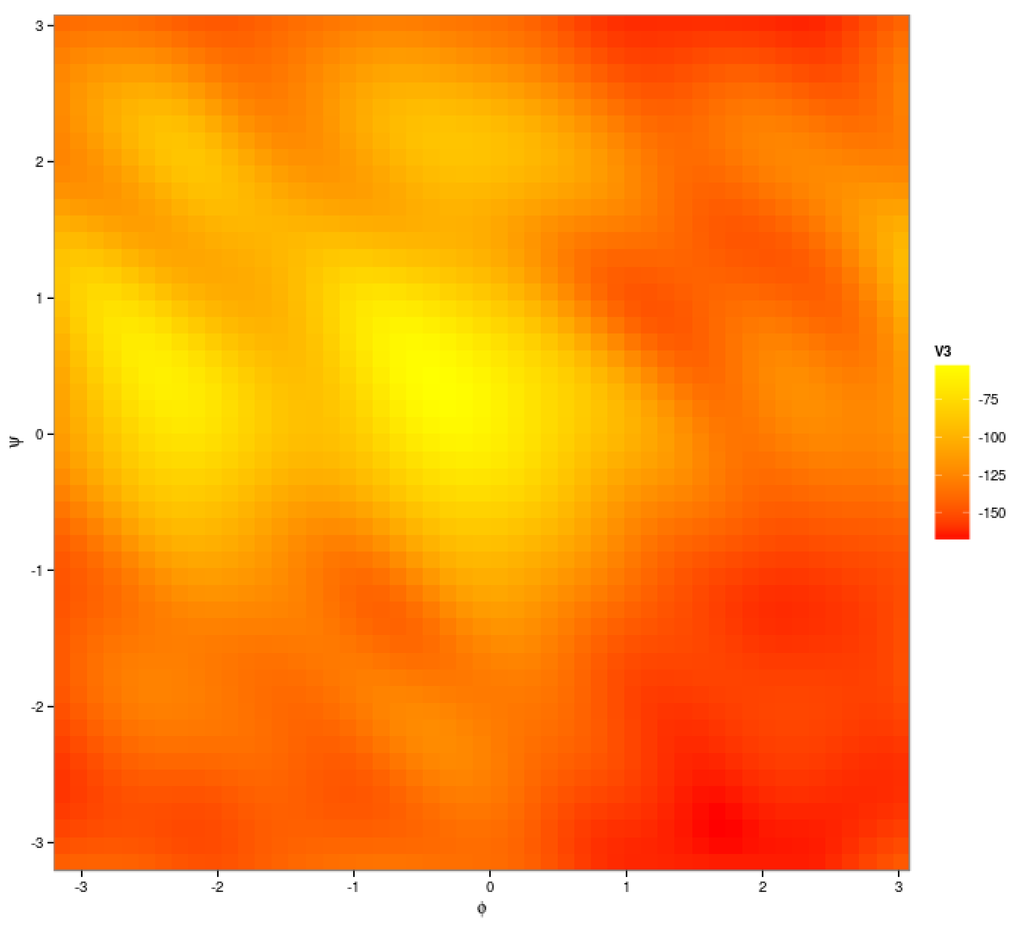


FIGURE 3.6: Free energy surface over the two glycosidic angles for the atomistic disaccharide ALLA1-4ALLA calculated by metadynamics. Scale in kcal mol^{-1} and angles in radians. Red is low energy and favourable.

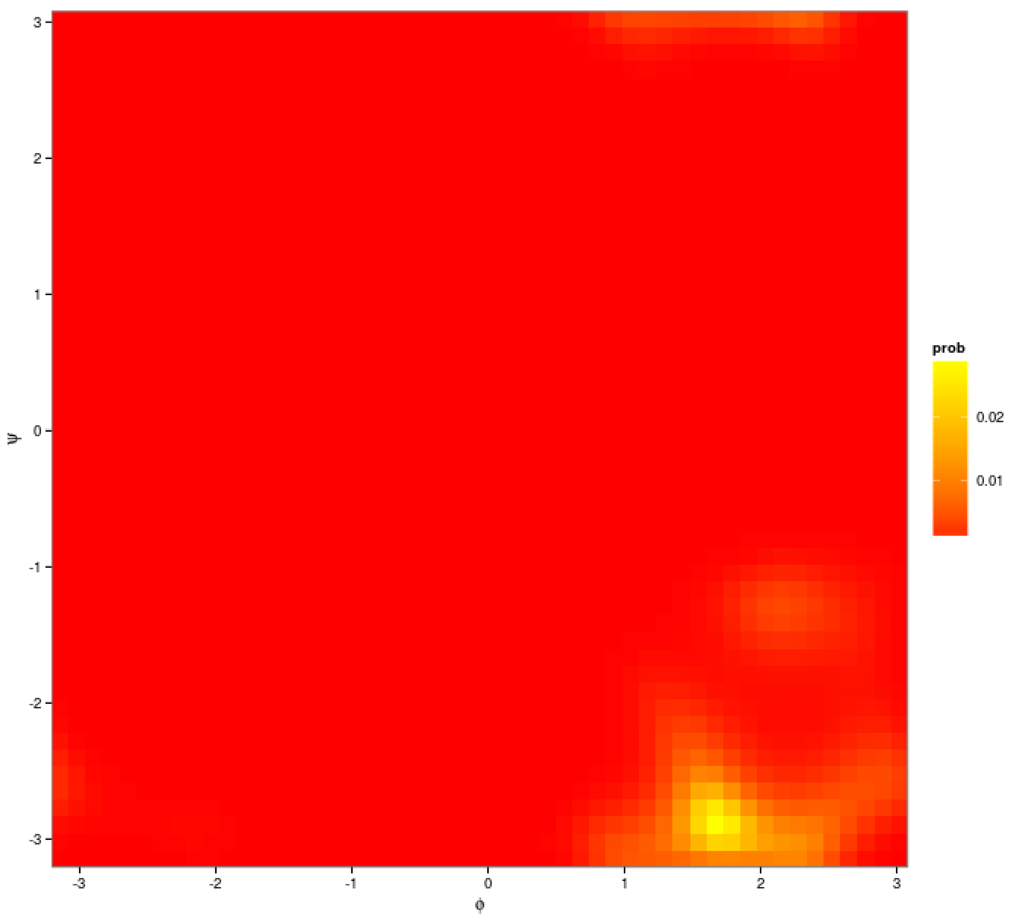


FIGURE 3.7: Boltzmann weighted probability for free energy surface over the two glycosidic angles of ALLA1-4ALLA. Scale is unitless, angles in radians. Red is low probability and unfavourable.

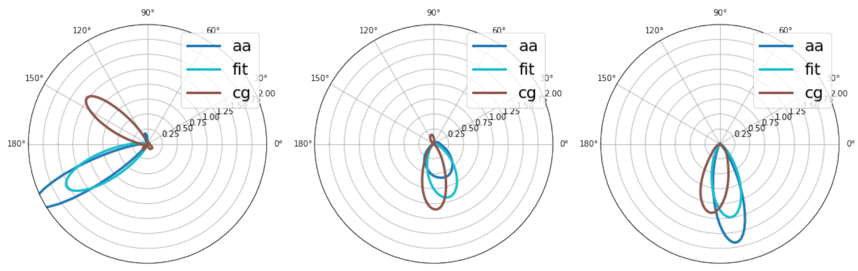


FIGURE 3.8: KDE of the three glycosidic angles in the disaccharide GLCA1-3ALLA comparing the fitted coarse-grained model to atomistic

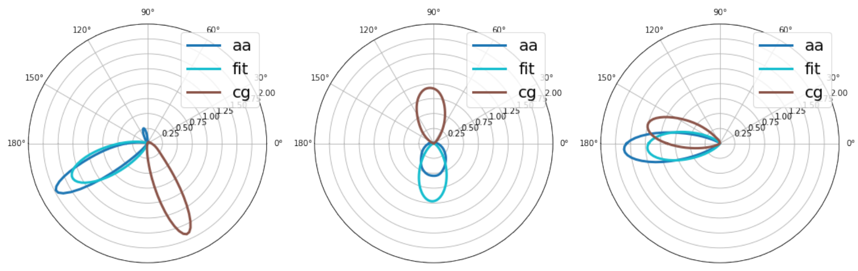


FIGURE 3.9: KDE of the three glycosidic angles in the disaccharide GLCB1-3ALLA comparing the fitted coarse-grained model to atomistic

the GLYCAM forcefield. Figures 3.8 to 3.13 show circular KDE plots overlaying the glycosidic angles measured from these atomistic simulations, with a fitted Gaussian and the angles measured from a CG simulation of the same disaccharides using the fitted parameters.

3.7 Simulation Methods

All atomistic simulation data for sugars and solvent models was collected using GROMACS versions 5.0.4 and later running either locally using GPU acceleration, on a small scale GPU workstation cluster, or remotely on the Iridis4 compute cluster. A typical production simulation was 100 ns long using a timestep of 2 fs using PME electrostatics and cut-off van der Waals interactions. When using AMBER based models (GAFF and GLYCAM), both cut-off distances were 8 Å, whereas a cut-off of 12 Å was used for GROMOS based models. The v-rescale thermostat at 310 K was

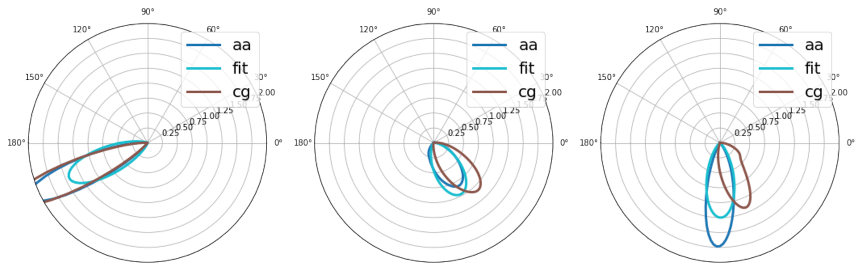


FIGURE 3.10: KDE of the three glycosidic angles in the disaccharide GLCA1-4GLCA comparing the fitted coarse-grained model to atomistic

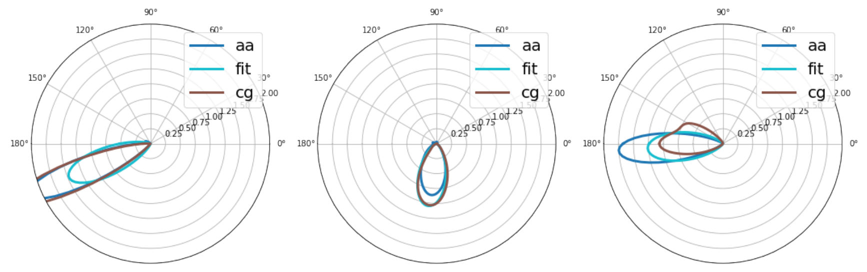


FIGURE 3.11: KDE of the three glycosidic angles in the disaccharide GLCB1-4GLCB comparing the fitted coarse-grained model to atomistic

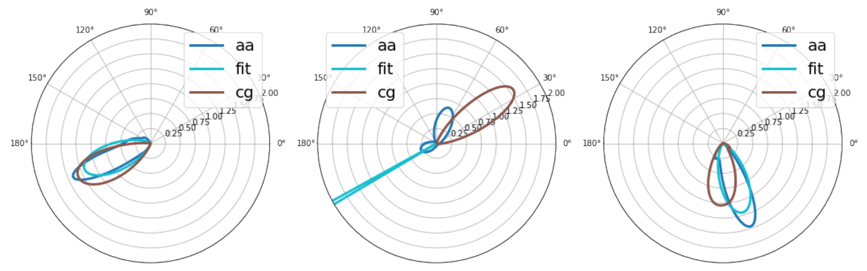


FIGURE 3.12: KDE of the three glycosidic angles in the disaccharide GLCA1-6GLCA comparing the fitted coarse-grained model to atomistic

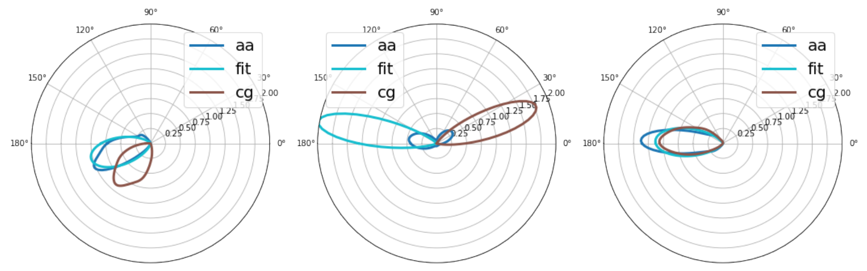


FIGURE 3.13: KDE of the three glycosidic angles in the disaccharide GLCB1-6GLCA comparing the fitted coarse-grained model to atomistic

used in conjunction with the isotropic Parrinello-Rahman barostat at 1 bar with a compressibility of 4.5×10^{-5} bar^{-1}. LINCS constraints were used on all bonds.

Input configurations for all atomistic production simulations had been pre-equilibrated using a protocol of: initial energy minimisation, NVT equilibration for 50 ps with the Berendsen thermostat, and finally NPT equilibration for 500 ps using the Berendsen thermo- and isotropic barostat.

Coarse-grained simulations were performed using LAMMPS the current version as of Autumn 2014 and later with added code for several dipole control potentials. The later versions of LAMMPS included the corrected angle dipole potential.

LAMMPS simulations were performed locally in a single input script with initial energy minimisation, NVT equilibration with the Langevin thermostat, typically at 303 K with rotational thermostatting, followed by NPT equilibration using the isotropic Berendsen barostat. Production simulation was performed under the same conditions. Timesteps were typically 5 fs to 10 fs with equilibration typically being of the order of 100 ps and production simulation of the order of 100 ns.

Quantum mechanical calculations were performed in Gaussian09 on the Iridis4 cluster using a range of DFT functionals: B3LYP, PBEPBE and PBE0.

3.8 Conclusions and Future Work

The ELBA sugar model as developed here goes some way towards the aim, but has significant limitations. Through the attachment of dipoles to each ring bead, the model provides a level of detail not present in other CG models of sugars. It is able to distinguish and transition between the range of conformations of the monosaccharide ring and able to distinguish each of the 16 stereoisomers.

However, the representation that was required to fix the positions of the dipoles means that the model uses only slightly fewer particles per monosaccharide than existing UA models. The computational expense of the electrostatics model, and the limitation that ELBA may only be run with the LAMMPS simulator, mean that the speed of simulation is significantly slower than existing AA and UA models.

Simulations of disaccarides are possible with this model, though parameters for the linkages between rings are not yet finalised. This constitutes part of the further work which is being undertaken by others. The performance limitations may be mitigated by restructuring the dipole restraints to reduce the number of particles, at the cost of accuracy, or by making such modifications that ELBA is able to run in another, more performant simulation engine.

Chapter 4

PyCGTOOL

4.1 Introduction

Development of coarse-grained (CG) molecular dynamics models is often a laborious process which commonly relies upon approximations to similar models, rather than systematic parametrisation. PyCGTOOL automates much of the construction of CG models via calculation of both equilibrium values and force constants of internal coordinates directly from atomistic molecular dynamics simulation trajectories.

The derivation of bespoke parameters from atomistic simulations improves the quality of the CG model compared to the use of generic parameters derived from other molecules, while automation greatly reduces the time required. The ease of configuration of PyCGTOOL enables the rapid investigation of multiple atom-to-bead mappings and topologies. Although PyCGTOOL is here used in combination with the GROMACS molecular dynamics engine its use of standard trajectory input libraries means that it is in principle compatible with other software.

PyCGTOOL is available from the URL `https://github.com/jag1g13/pycgtool` as doi:10.5281/zenodo.569555.

4.2 Software Engineering

In order for software to be considered of good quality, it should satisfy several criteria [48, 49]. It should be:

1. correct
2. easy to use (usability)
3. efficient
4. extensible

The primary concern of any software should be correctness. If it produces incorrect results it is not fit for purpose and cannot be considered 'good software'. Correctness may be ensured by software testing, of which there are several kinds: The lowest level is unit testing: tests of individual units of code, often classes or functions that ensure they behave correctly in isolation. One level up is integration testing: do the individual components behave correctly when brought together. The two highest levels are system testing and acceptance testing: does the program as a whole perform the tasks laid out in its requirements correctly, and does it do so in an acceptable manner. Acceptance testing is therefore not strictly a test of correctness, but rather a combination of the other desirable properties of 'good software'; it also serves as a test that the requirements specification correctly matched the true requirements of the users.

Modern software engineering has extended the concept of software testing in the form of 'continuous integration'. This is a process whereby automated tests are run with every change to the codebase, meaning that component failures may be identified immediately. Combined with good test coverage, this helps to reduce the risk of distribution of incorrect code, particularly in cases where members of a team may be making independent changes.

Testing, taken to its furthest point, becomes Test Driven Development (TDD), whereby the tests are written before the code. This has several significant advantages: it provides a concrete description of what the software will do before code is written, and it ensures that test coverage is high so bugs are less likely to remain unnoticed.

The remaining properties of 'good software' are more flexible. A piece of software may be forgiven for performing poorly by one criterion if it performs well on others. For instance, within the field of molecular dynamics, the program LAMMPS performs poorly on usability, but is more easily extensible than its competitors.

Extensibility may be promoted by attempting to reduce coupling between components to achieve a modular design.

4.3 Introduction

As described previously, the construction of a CG model of a novel (one for which a model does not yet exist) molecule is time-consuming and involves repetitive measurements from atomistic simulation data for iterative refinement of these parameters. In addition, to test alternate mappings (i.e. which atoms are subsumed into each CG bead) it is often necessary to completely reparametrise large parts of the molecule in cases where there is no single obvious mapping or bonded topology.

In order to streamline and automate the process as much as possible, the PyCGTOOL software has been developed. This is a tool for the automated generation of bonded parameters for CG models within the MARTINI forcefield or as a part of a more complex forcefield such as the ELBA [24] CG forcefield. Whilst the GROMACS MD engine is used throughout, it is in principle replaceable by any MD engine able to output trajectories in one of the common MD trajectory formats. Similarly, the target forcefield is MARTINI, though PyCGTOOL is able to generate bonded terms for any forcefield using the same functional forms.

Key features are:

- Generation of coarse-grained internal coordinate parameters directly from atomistic simulation trajectories
- Creation of initial system coordinates for CG simulations
- Simple configuration file syntax enabling changes to the model to be made quickly and easily
- Functionality to aid in CG model validation and comparison back to atomistic data
- 'Mapping only' mode to aid in simulation setup using existing parameters
- Open source with all dependencies available in the Python Package Index

Note that this software does not intend to replace existing MARTINI parametrisations, but rather to reduce the difficulty and effort in studying a molecule which

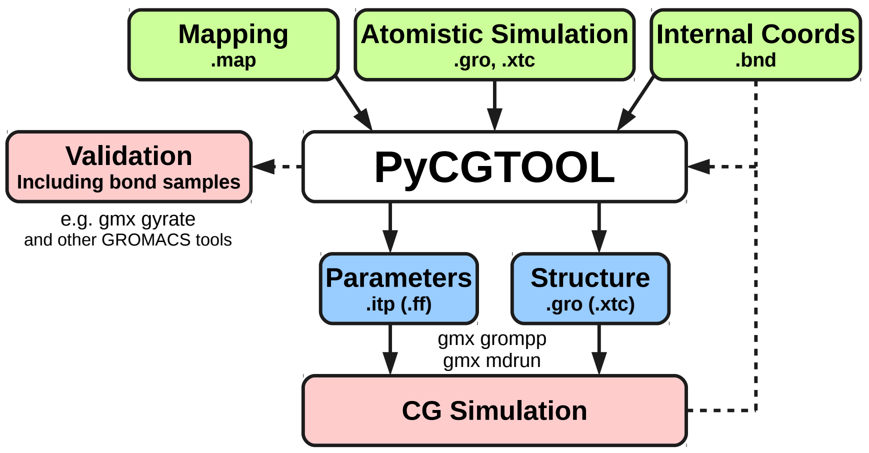


FIGURE 4.1: The PyCGTOOL workflow. Green: An atomistic simulation trajectory is fed into PyCGTOOL, along with the atomistic-to-CG mapping (as *.map* and *.bnd* files). Blue: Output is a calculated parameter set and system structure. Red: Initial validation may be performed by passing a CG trajectory through PyCGTOOL once more to collect samples of internal coordinate values.

does not yet have a MARTINI parametrisation. This is particularly anticipated to be useful in the simulation of small, drug-like molecules.

4.4 Workflow

PyCGTOOL formalises the workflow (fig 4.1) for development of a coarse-grained model to allow rapid creation of a single CG model, but also to simplify iterative improvement in cases where there may be multiple plausible atomistic-to-CG mappings which should be investigated. The workflow enabled by PyCGTOOL consists of three major stages: preparation, generation, and validation.

The main task for the user during the preparation stage is to devise an atomistic-to-CG mapping. The mapping process specifies for each CG bead in the generated model: which atoms will be combined to create the single CG bead, the name of the CG bead, its MARTINI bead type, and optionally the bead charge. This mapping is provided to PyCGTOOL in a *.map* file which may contain mappings for more than one molecule. Next, the internal coordinates to be parametrised must be defined. PyCGTOOL takes as input a *.bnd* file, containing a list of pairs of named beads (i.e. matching the names defined in the mapping file) between which bonds will be measured. Optionally, triplets of beads may be listed to define angles, although this is not strictly necessary, as all valid triplets will be created by PyCGTOOL stepping through the molecular graph if none are defined. Both of these files are easily modifiable plain text files with the expected format described both towards the end of this chapter and in the documentation (available at `http://pycgtool.readthedocs.io/`). The final input to the parameter generation phase is a reference simulation trajectory in standard GROMACS [5] *.gro* and *.xtc* formats, or other standard formats if using the optional MDTraj library [50].

The second stage, generation, is defined by the options provided to PyCGTOOL. For each residue in each frame of the input simulation trajectory, the mapping defined in the *.map* file is applied; if a mapping is not found for a residue the residue will be skipped. The resulting mapped trajectory frame is referred to as the pseudo-CG representation; it has a topology matching that of the final CG model, but was not the output of a CG simulation. The position of a bead in the resulting pseudo-CG frame is by default the centre of geometry of its component atoms in the reference frame, though this may be configured to use the centre of mass if atom masses in the reference trajectory can be guessed from their atom names.

From the pseudo-CG representation, for each residue in each simulation frame, the defined internal coordinates are measured. Mean values and standard deviations are calculated at the end of the process for each internal coordinate, which are in turn used to calculate force constants using the equations in the following section. Optionally, all force constants may be assigned the default MARTINI values of 1250 kJ mol^{-1} nm^{-2} for bond lengths and 25 kJ mol^{-1} for angles, if there is a strict requirement that the model conform to this aspect of the MARTINI convention.

To aid in validation, a random sample of measurements of each internal coordinate may be output at this stage; by default $N = 10\,000$, a sample size which was found to be large enough to be statistically similar to the population. The full pseudo-CG trajectory may also be exported for analysis with standard GROMACS tools.

To improve the stability of simulations using the generated CG model and to allow a larger timestep to be used, two modifications to the resulting parameters are made. First, all bonds with force constants higher than a threshold value are converted to constraints. By default the threshold is set at 100 000 kJ mol^{-1} nm^{-2}, although this is user-configurable. Second, angles defined within a closed triangle of beads are removed. These angle definitions are redundant with the definition of the three associated bond lengths, and were found during testing to greatly decrease the stability of simulations. These modifications permitted a timestep of 20 fs to be used in all tested cases, with some of the models being stable up to 40 fs.

The validation stage is the most complex from the perspective of the user, during which the generated model is tested for satisfactory replication of physical properties with respect to the reference simulation. The most basic analysis used in validation is to compare the distributions of bond lengths and angles from simulations using the generated models with the samples of internal coordinates measured from the reference trajectory during the parametrisation stage. These distributions were compared in the validation section of this chapter using a series of Tukey boxplots to allow comparison of both median values, and interquartile ranges.

Further validation is performed by analysis of properties relevant to the class of molecule in question: for instance, models of membrane lipids are commonly assessed by how well they replicate the values of membrane thickness and surface area per lipid. For small drug-like molecules, suitable criteria may be radius of gyration, an indirect measure of conformation, and their interaction with systems such as proteins or membranes.

4.5 The Model and Methodological Considerations

Since the major target forcefield for PyCGTOOL is MARTINI, the internal coordinate model follows MARTINI convention in using the simple harmonic potential for bond lengths (GROMACS bond type 1) and a cos-harmonic potential for angles (GROMACS angle type 2), while dihedrals are usually not defined. Since the required functional form is known, it is possible to calculate a mean value and force constant for each internal coordinate.

The assumption of normality and use of simple functional forms for internal coordinate potentials precludes an accurate representation of multi-modal bond length or angle distributions. A proper representation of multi-modal distributions would require use of GROMACS's tabulated potentials, as is common for non-bonded terms in more complex CG models [51], but this would undermine the simplicity of the MARTINI model and comes at a potentially significant simulation performance penalty. Although dihedrals are not common in MARTINI parametrisations, with the exception of the protein forcefield, PyCGTOOL is able to generate them if they are defined in the input bond file. The same assumption of a unimodal normal distribution is used, meaning that they should only be used in cases where there is a single favoured conformation. Dihedrals were not used in any of the validation models presented here. It is hoped that a future version will include the ability to fit dihedrals with a multiplicity greater than one.

4.5.1 Modified Boltzmann Inversion

Measurements from the pseudo-CG trajectory are used to calculate a mean and standard deviation for each internal coordinate, relying on the assumption that the measurements are normally distributed. A modification of the Boltzmann Inversion technique [52] is used whereby the Boltzmann Inversion transformation $-RT \log f(x)$ is applied to both the target functional form and the assumed normal distribution of measured values. This method allows force constants to be calculated directly from the collected bond sample. Boltzmann Inversions for the default length and angle functional forms used in the MARTINI forcefield are built into the software, while alternative potentials may be added by other developers relatively simply at a later date. These potentials used in the MARTINI forcefield use the same functional form as those in the ELBA forcefield, allowing bonded parameters to be used after a simple conversion to the LAMMPS [19] unit system.

By applying the transformation eqn 4.2 to the standardised normal distribution it can be made equivalent to the target functional form, making the assumption of independent degrees of freedom, and hence no correlation. If this transformation is applied to the standard deviation, the target force constant can be obtained as in eqns 4.4, 4.6. GROMACS gives a conversion factor between the harmonic and cos-harmonic forms of the angle potential (eqn 4.5) which is used to calculate force constants for the cos-harmonic functional form. This method is similar to the Boltzmann Inversion technique [52], but uses the assumption of normality of the measured values to obtain a force constant rather than a tabulated potential.

$$f(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \equiv e^{-\frac{k(x-x_0)^2}{2RT}} \tag{4.1}$$

$$-RT \log f(x) = \frac{RT(x-\mu)^2}{2\sigma^2} \tag{4.2}$$

$$k(x-x_0)^2 \equiv \frac{RT(x-\mu)^2}{\sigma^2} \tag{4.3}$$

$$k = \frac{RT}{\sigma^2} \quad \text{simple harmonic} \tag{4.4}$$

$$k_{\cos} = \frac{k_{\text{harm}}}{\sin^2 x_0} \tag{4.5}$$

$$k_{\cos} = \frac{RT}{\sigma^2 \sin^2 \mu} \quad \text{cos harmonic} \tag{4.6}$$

Where x is the internal coordinate, x_0 is the equilibrium value of x, μ is the mean value of x, and σ is the standard deviation of x. The resultant force constants are k for a harmonic functional form, as used for bond lengths, and $k_{\cos}$ for a cosine-harmonic functional form, as used for angles.

In contrast with a standard Boltzmann Inversion, which results in a tabulated potential, the parameters obtained from the modified Boltzmann Inversion fit into the standard MARTINI conventions, making them more easily comparable and able to be combined with existing models. Tabulated potentials in GROMACS come with a large decrease in simulation performance due to a less optimised code path, and mean significantly more effort for the researcher in both managing and validating the potential. Since the central benefit of MARTINI is ease of use (even at the expense of accuracy), it does not make sense to sacrifice this for more accurate bonded potentials.

4.5.2 MARTINI Convention

One of the first stages of the standard workflow for simulation setup when using atomistic models in GROMACS is the program *gmx pdb2gmx*, which takes as input a coordinate file containing atom and residue names, usually in *.pdb* or *.gro* format, and outputs the system topology containing both bonded and non-bonded terms. This is performed by lookup of residue names within pre-packaged forcefield database directories. For each residue the appropriate residue record is identified, and atom names are checked before copying parameters into a *.top* topology file which is then used as part of the simulation input. The tool *gmx pdb2gmx* also allows topologies for polymers, such as proteins, to be constructed from monomer records. Since this polymer functionality has proved useful for atomistic models, PyCGTOOL is able to export a GROMACS style forcefield directory, allowing *gmx pdb2gmx* to be used with coarse-grained models, which has not previously been possible.

The method of parameter generation in PyCGTOOL differs from the existing *Auto_MARTINI* [53] method in that PyCGTOOL makes use of a complete reference simulation trajectory as opposed to a single geometry-optimised snapshot, meaning that dynamic fluctuations are explicitly considered in the form of individual force constants. The *Forcebalance* program [54] also has the potential to be used in the generation of parameters for CG models although it would require significant setup work and modification; PyCGTOOL is designed to be easy to use and fit within the framework of relatively simple CG models such as MARTINI. Additionally, since the parametrisation process in *Forcebalance* performs additional simulations, while this code does not, PyCGTOOL would be able to more quickly generate a series of alternate CG mappings. Both *Auto_MARTINI* and *Forcebalance* do however have their own advantages: *Auto_MARTINI* is able to generate its own mapping rather than requiring one be provided by the user; *Forcebalance* has greater flexibility in fitting more complex functional forms and allows additional target data to be used in the fitting process.

For convergence and accuracy of calculated parameters, the most significant limitation is sampling of the conformational landscape, as is true for MD simulations in general [55]. A relatively rigid molecule may require only a few tens of nanoseconds of atomistic simulation for parameters to converge, whereas a more flexible molecule is likely to require longer. Many of the validation models were generated using atomistic simulation trajectories of 50 ns. The analysis requires only snapshots taken from an ensemble and has no time dependence, so it is possible to use hybrid trajectories compiled from multiple simulations, or with the use of enhanced sampling methods, to aid sampling in difficult cases. When building the ensemble using an enhanced sampling method, it is important that the ensemble be correctly re-weighted so as not to skew the distributions from which the bond measurements will be taken. This sampling limitation is greatly reduced in the case of membrane lipids where a single simulation may contain hundreds of replicated molecules, thus sampling many conformations at once and giving better confidence in the equilibrium conformation.

Corrections for periodic boundaries are applied both during the construction of the pseudo-CG particles and during the calculation of bond vectors in measuring bonded terms. This means that molecules crossing the periodic boundary present no issue so trajectory pre-processing using the GROMACS tools is not required.

PyCGTOOL operates only on atom names provided *via* the input coordinate *.gro* file and the user defined atomistic to CG mapping, and thus its operation is independent of forcefield and simulation parameters used in the reference simulation. The single exception is the temperature at which the reference simulation was performed, which is required for correct calculation of force constants. PyCGTOOL has been used largely to derive parameters from simulations with the GROMOS [56] united-atom forcefield, but also with the AMBER-based GLYCAM06 [29] fully atomistic carbohydrate model in the generation of parameters for the ELBA CG forcefield.

4.6 Simulation Methods

4.6.1 Atomistic

GROMOS

All atomistic simulations performed in the validation of PyCGTOOL, with the exception of the capsaicin reference which was provided by the Sansom group [57] and the polyalanine reference, were performed using the GROMOS 53A6 force field in SPC water using GROMACS 2016. A simulation timestep of 2 fs was used with PME electrostatics and both direct-space electrostatic and Lennard-Jones cutoffs of 1.2 nm with LJ dispersion correction. Constraints were applied to all bond lengths using LINCS. All simulation systems were equilibrated by NVT using the v-rescale thermostat at 310 K for 100 ps, then NPT using the Nose-Hoover thermostat and Parrinello-Rahman barostat at 1 bar for 1 ns. The reference simulation trajectory was collected over 50 ns under the same conditions as the NPT equilibration.

AMBER

The atomistic simulation used as the reference model of polyalanine was performed using the AMBER03 force field in TIP3P water using GROMACS 2016. A simulation timestep of 2 fs was used with PME electrostatics and both direct-space electrostatic and Lennard-Jones cutoffs of 0.8 nm with LJ dispersion correction. Constraints were applied to all bond lengths using LINCS. All simulation systems were equilibrated by NVT using the Berendsen thermostat at 310 K for 100 ps, then NPT using the Nose-Hoover thermostat and Parrinello-Rahman barostat at 1 bar for 1 ns. The reference simulation trajectory was collected over 50 ns under the same conditions as the NPT equilibration.

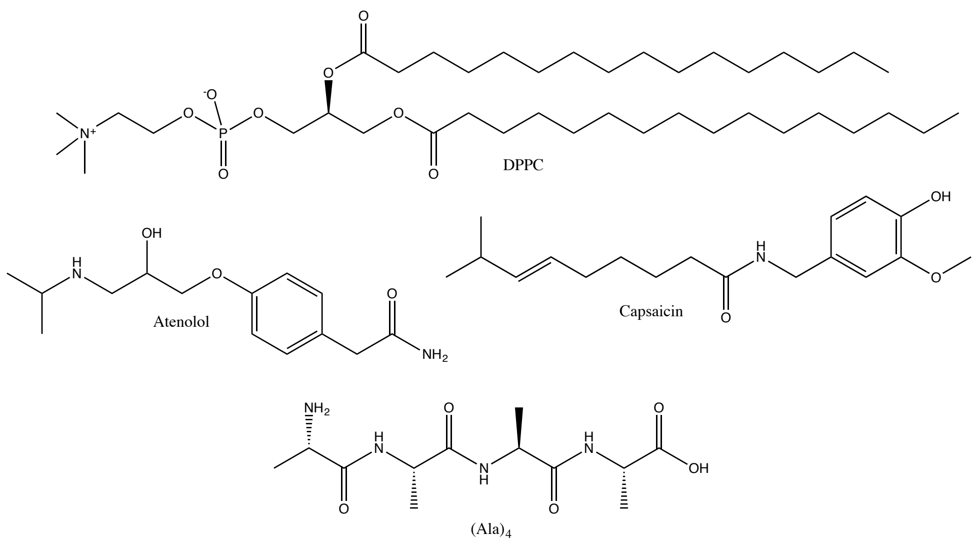


FIGURE 4.2: The molecules used for validation of PyCGTOOL.

4.6.2 MARTINI CG

All MARTINI CG models were simulated using GROMACS 2016 using the standard simulation settings provided on the MARTINI website. Electrostatic interactions were calculated using the reaction field method with a short range cutoff of 1.1 nm and screening constant of 15. Lennard-Jones interactions used a potential-shifted cutoff at 1.1 nm. Only bond lengths explicitly listed in the topology as constraints were constrained using LINCS.

Simulation systems were equilibrated first in NVT using the v-rescale thermostat at 310 K for 1 ns with a timestep of 5 fs, then in NPT using the v-rescale thermostat and Berendsen barostat at 1 bar for 100 ns with a timestep of 10 fs. The simulation trajectory was collected over 500 ns with a timestep of 20 fs using the v-rescale thermostat and Parinello-Rahman barostat at 1 bar.

4.7 Validation

Validation of PyCGTOOL was performed using a range of target molecules comprising: two drug-like molecules, atenolol and capsaicin; the membrane lipid dipalmitoylphosphatidylcholine (DPPC); and a short four-residue strand of polyalanine as a test of the polymer functionality. The structures of these molecules are shown in figure 4.2.

All CG models were taken directly as output by PyCGTOOL; hand-tuning may be desired in practice in certain cases to achieve a better fit to reference properties, but often the model may be taken unmodified.

Several models were created for each of these molecules: a model using the default settings of PyCGTOOL; a model using default MARTINI force constants *via* PyCGTOOL, but still using the entire simulation trajectory for calculation of equilibrium values of parameters; and a model created using only a static snapshot (referred to in figures as 'naive MARTINI'). This final model was created using only the final trajectory frame of the reference atomistic simulation for measurement of equilibrium values, and using the default MARTINI force constants; this model is not intended to represent a typical manually generated model, but rather a model generated in the simplest possible manner.

From simulations with each of these CG models, the internal coordinates were measured and compared to the atomistic reference model. Further measurements were made for each target, relevant to the class of molecule to which they belong.

4.7.1 Drug-like Molecules

The two drug-like molecules tested were capsaicin, with topology and simulation trajectory provided by the Sansom group [57], and atenolol, using the GROMOS 54A7 forcefield taken from the ATB database [58].

For each of these molecules, a range of models were generated, as described previously. In addition to the general measurements of bonded terms, radii of gyration were measured for each model and the reference atomistic model using the GROMACS tool *gmx gyrate*. To allow direct comparison, the atomistic reference model was first mapped to a pseudo-CG representation. The default PyCGTOOL model for each molecule was also inserted into a pre-equilibrated MARTINI POPC membrane simulation, starting in the solvent a short distance away from the membrane surface.

Atenolol

Tukey Box-and-Whisker plots show the distribution of a sample. The bounds of the box are the upper and lower quartiles while line across the middle of the box represents the median. The whiskers extend to the farthest value within 1.5× the interquartile range. Outliers may be represented by dots, but they are omitted here.

The Tukey boxplots shown in fig 4.3 list each of the bond lengths and angles defined in the CG atenolol model, numbered in the order they are present in the input and output files. The upper series of boxplots shows the distribution for each defined bond length in each model with clearly strong agreement between the atomistic reference and default PyCGTOOL models, both in median and in spread (box width corresponds to interquartile range). This is a result of the individually calculated force constants which are disabled in the model generated using default MARTINI force constants. That bonds with high force constants are automatically converted to constraints during the parametrisation process is useful for molecules containing small, relatively rigid groups such as the phenyl ring in atenolol, as it allows them to be kept rigid without reducing the simulation timestep from a standard 20 fs as may be required without using constraints. The lower figure shows the distributions for the defined angles showing generally good agreement in median values across all generated models although the spread is noticeably under-represented in the PyCGTOOL model for the first two angles.

The radius of gyration is compared between the models in fig 4.4, showing that the model generated using PyCGTOOL with default settings most closely replicates the median radius of gyration of the atomistic simulation by a small margin.

Generation of each CG model by PyCGTOOL required a total of 45 s to process 25 000 reference trajectory frames of the 3671 atom simulation. This time is reduced to 11 s if the solvent is stripped from the trajectory in advance, showing that for reference trajectories containing only a single target molecule, the speed of the software is primarily limited by file access.

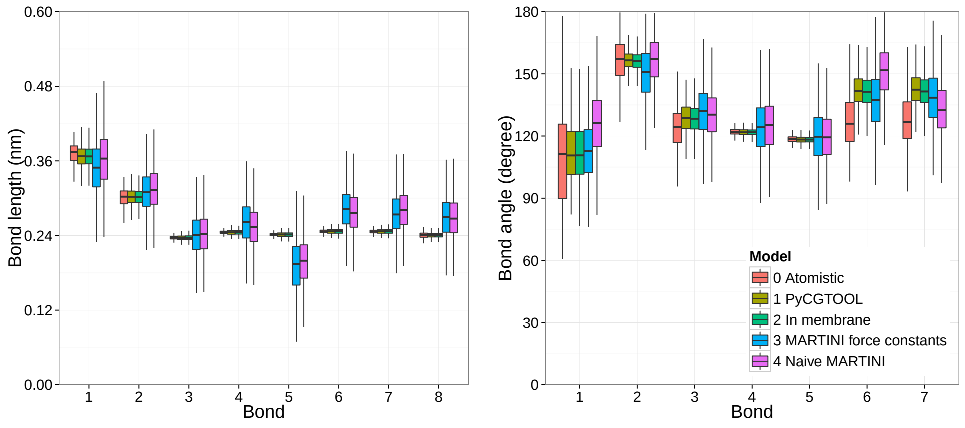


FIGURE 4.3: Tukey boxplots showing bond length and angle distributions for the molecule atenolol. Note here that default force constants differ significantly from the atomistic reference, producing much wider distributions for many of the bonds. For angle seven in the bottom panel note that the model using a calculated force constant is slightly less accurate in median and interquartile range. The mean and standard deviations for each of these bond lengths and angles are shown in tables 4.1 and 4.2. These tables are omitted for the other validation molecules.

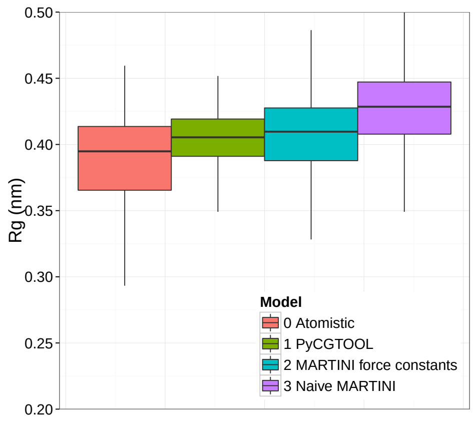


FIGURE 4.4: Radius of gyration for atenolol. The default PyCGTOOL generated model gives a median radius within the interquartile range of the reference atomistic model.

Bond	Model	Mean	Std dev
1	Atomistic	0.3710	0.016880
1	PyCGTOOL	0.3674	0.017370
1	In membrane	0.3671	0.017390
1	MARTINI force constants	0.3493	0.045030
1	Naive MARTINI	0.3637	0.046610
2	Atomistic	0.3005	0.014110
2	PyCGTOOL	0.3022	0.013750
2	In membrane	0.3018	0.013240
2	MARTINI force constants	0.3113	0.035240
2	Naive MARTINI	0.3150	0.035140
3	Atomistic	0.2364	0.002950
3	PyCGTOOL	0.2363	0.004108
3	In membrane	0.2365	0.004148
3	MARTINI force constants	0.2417	0.035340
3	Naive MARTINI	0.2430	0.035650
4	Atomistic	0.2452	0.002682
4	PyCGTOOL	0.2452	0.004148
4	In membrane	0.2453	0.004093
4	MARTINI force constants	0.2620	0.037390
4	Naive MARTINI	0.2542	0.034160
5	Atomistic	0.2414	0.002614
5	PyCGTOOL	0.2415	0.004075
5	In membrane	0.2415	0.004089
5	MARTINI force constants	0.1910	0.043970
5	Naive MARTINI	0.1987	0.040790
6	Atomistic	0.2465	0.003129
6	PyCGTOOL	0.2467	0.004161
6	In membrane	0.2468	0.004141
6	MARTINI force constants	0.2825	0.034990
6	Naive MARTINI	0.2778	0.035240
7	Atomistic	0.2463	0.003146
7	PyCGTOOL	0.2465	0.004106
7	In membrane	0.2466	0.004124
7	MARTINI force constants	0.2747	0.035430
7	Naive MARTINI	0.2813	0.034460
8	Atomistic	0.2405	0.004948
8	PyCGTOOL	0.2405	0.004125
8	In membrane	0.2405	0.004126
8	MARTINI force constants	0.2701	0.035800
8	Naive MARTINI	0.2685	0.034880

TABLE 4.1: Mean and standard deviations of measured bond lengths in atenolol models.

Bond	Model	Mean	Std dev
1	Atomistic	109.5	23.170
1	PyCGTOOL	112.7	14.790
1	In membrane	112.7	14.830
1	MARTINI force constants	113.3	14.580
1	Naive MARTINI	125.9	16.100
2	Atomistic	156.3	10.760
2	PyCGTOOL	156.5	4.623
2	In membrane	156.2	4.526
2	MARTINI force constants	150.0	13.390
2	Naive MARTINI	156.2	11.740
3	Atomistic	123.8	9.713
3	PyCGTOOL	128.7	7.091
3	In membrane	128.3	7.108
3	MARTINI force constants	132.1	13.230
3	Naive MARTINI	130.4	12.200
4	Atomistic	122.0	1.603
4	PyCGTOOL	121.8	1.683
4	In membrane	121.8	1.645
4	MARTINI force constants	124.5	13.770
4	Naive MARTINI	125.6	13.430
5	Atomistic	118.5	1.589
5	PyCGTOOL	118.3	1.606
5	In membrane	118.3	1.593
5	MARTINI force constants	119.8	13.520
5	Naive MARTINI	119.7	12.810
6	Atomistic	127.9	14.080
6	PyCGTOOL	142.1	7.883
6	In membrane	141.7	7.653
6	MARTINI force constants	137.3	14.190
6	Naive MARTINI	150.8	12.660
7	Atomistic	128.9	14.380
7	PyCGTOOL	142.6	7.838
7	In membrane	141.7	7.947
7	MARTINI force constants	138.5	13.590
7	Naive MARTINI	133.1	12.900

TABLE 4.2: Mean and standard deviations of measured bond angles in atenolol models.

Capsaicin

Similar boxplots are presented for the capsaicin validation (4.5, 4.6, 4.7). The default PyCGTOOL generated model is again seen to closely replicate bond lengths, particularly those within the aromatic ring. In the angle distributions there is little difference between the models using calculated or default MARTINI force constants. The median value for radius of gyration is marginally less accurate than the model using default MARTINI force constants, though both are similar and within the interquartile range of the reference model.

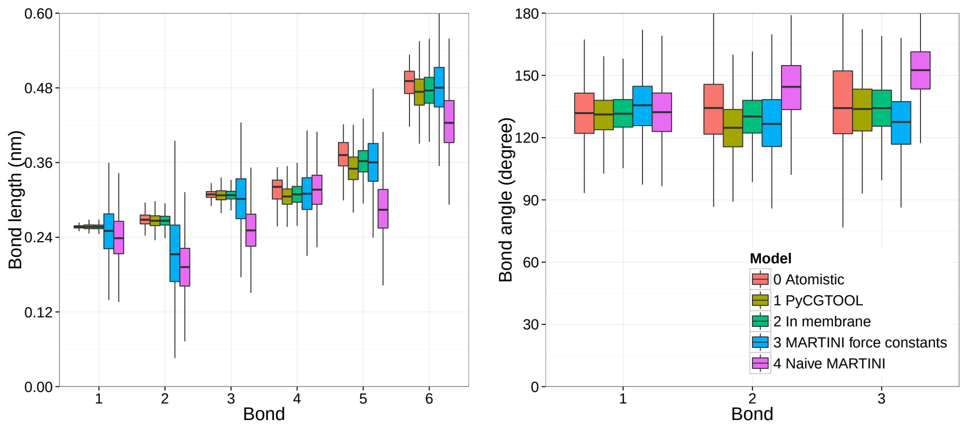


FIGURE 4.5: Bond length and angle distributions for the molecule capsaicin using a range of models.

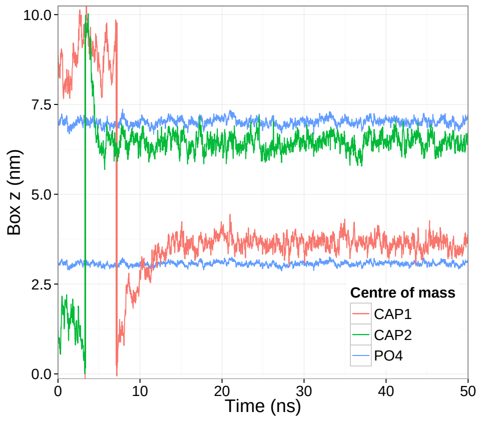


FIGURE 4.6: Centre of mass in z profile plot of a capsaicin-popc simulation. Both capsaicin molecules can be seen to enter the membrane within 15 ns where they remain until the end of the 1 μs simulation. Only the first 50 ns are shown here. The large jumps at approximately 3 ns and 7 ns are artefacts of both molecules crossing the periodic box boundary.

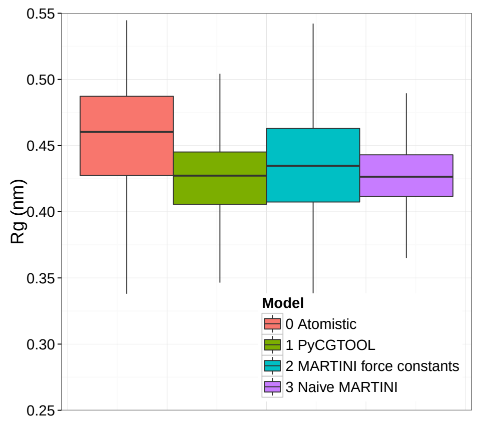


FIGURE 4.7: Radius of gyration for capsaicin.

4.7.2 Lipid: DPPC

The DPPC (1,2-dipalmitoyl-sn-glycero-3-phosphocholine) validation was used to test if, by using a one-to-one mapping, the reference forcefield parameters could be recovered from the simulation trajectory. The reference simulation consisted of a small membrane patch containing 128 molecules of DPPC using the MARTINI forcefield version 2.2. The output trajectory from this simulation was used as the reference model input to PyCGTOOL in an attempt to reconstruct the MARTINI parameters.

Interestingly, bond angles (shown in fig 4.8) measured from the MARTINI simulation trajectory were not as would be expected given the topology file. Tail angle potentials for saturated MARTINI lipids are set with an equilibrium angle of 180° and a force constant of 25 kJ mol^{-1}, while the median value of each tail angle during the simulation was consistently within the range of 140° to 150°. As a result of this, the angle potential equilibrium values calculated by PyCGTOOL do not match the reference MARTINI topology. However, simulations performed using this output do reproduce the properties of the reference simulation to a reasonable degree. Average membrane thickness differs from the reference by −2.6 %, while the difference in surface area per lipid (APL) is slightly larger at −5.2 %. Plots of these measurements are presented in the supplementary information. Measurements of membrane thickness and surface area per lipid were both performed using *RAMSi*, a component of the previous implementation of *CGTOOL* (available at `https://bitbucket.org/jag1g13/cgtool`) using a similar algorithm to the *GridMAT-MD* [59] tool.

Generation of each CG model by PyCGTOOL required approximately 700 s to process 10 000 reference trajectory frames of the 2585 bead simulation, containing 128 replica lipids. This time is reduced to 465 s if the solvent is stripped from the trajectory in advance, 230 s by running with the optional Python module dependency Numba, or 170 s using both optimisations.

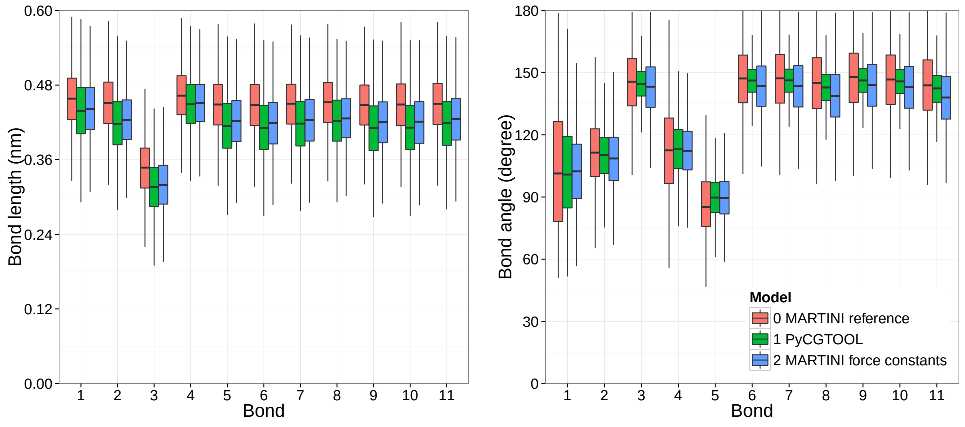


FIGURE 4.8: Bond length and angle distributions for the lipid DPPC as compared to a standard MARTINI reference simulation. Note here that the default force constant produces an accurate distribution as lipids are the class of molecules for which they were designed.

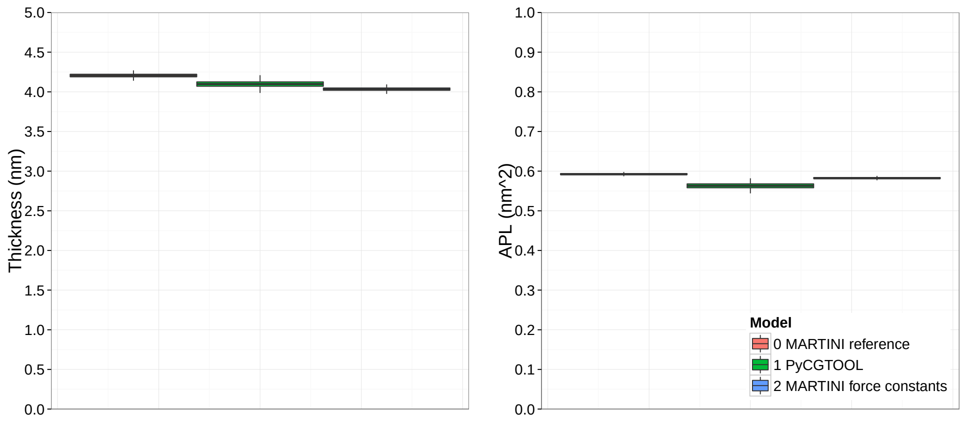


FIGURE 4.9: Membrane thickness and area per lipid as measured by RAMSi for the lipid DPPC.

4.7.3 Polymer: Polyalanine

For validation of the polymer functionality, a small, four-residue strand of polyalanine was used. A reference simulation was performed using the AMBER03 forcefield [60] and the TIP3P water model [61] in GROMACS 2016. The standard MARTINI protein mapping was used, whereby an alanine residue is represented by a single 'P4' bead. By use of the *output_forcefield* setting in the advanced options of PyCGTOOL, a full GROMACS forcefield directory with a *.rtp* residue topology file can be created. This enables the use of the GROMACS tool *gmx pdb2gmx* to construct the topology of alanine polymers of arbitrary length.

For the short alanine strand, the values measured were the same as those measured for the drug-like molecules. For measurement of bond lengths and angles, since an alanine monomer is mapped as a single residue containing a single bead, each bond length in the chain is equivalent, as is each angle. The upper plot in fig 4.11 shows very close agreement between the model generated using default settings and the reference model, both in median and spread. In the angle plot there is very little difference between models, indicating that the default MARTINI force constant is satisfactory in this case. Radius of gyration in fig 4.12 also shows that

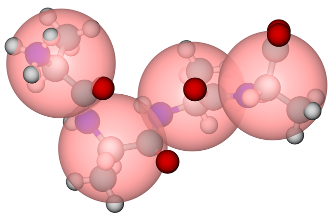


FIGURE 4.10: Coarse-grain mapping for the short strand of polyalanine. The atomistic frame was mapped to its pseudo-CG representation for simple visualisation.

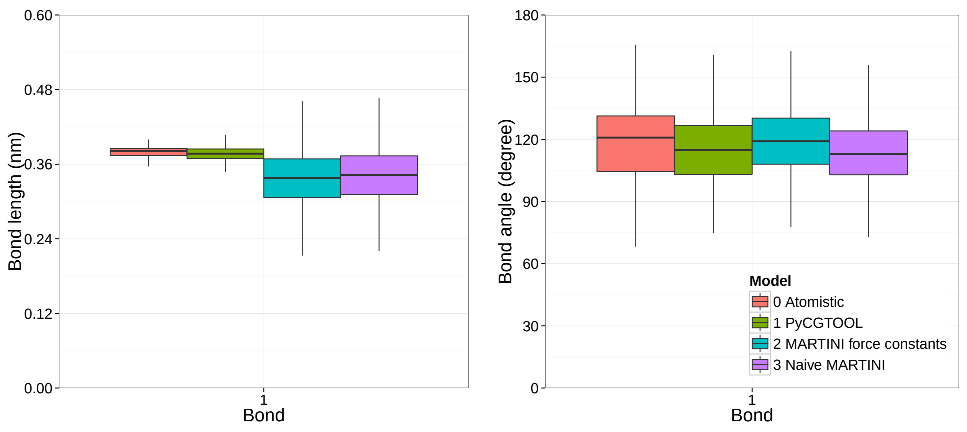


FIGURE 4.11: Tukey boxplots showing bond length and angle distributions for an alanine tetramer. All three bond lengths in the tetramer are considered equivalent, so are represented by a single box, as are both bond angles.

the model generated using default settings provides the best agreement with the reference model.

4.8 Conclusions

PyCGTOOL is a user-friendly program for automated generation of coarse-grained (CG) models of molecules, compatible with the popular MARTINI forcefield for biomolecular simulations, and others. The CG models that are produced by PyCGTOOL are generated from atomistic simulation trajectories, rather than fitting to a single snapshot and therefore provide a more accurate representation of the distribution of bond lengths and angles from their atomistic reference models. A small modification of the Boltzmann Inversion technique is used to generate the CG parameters from atomistic simulation trajectories, with only the atomistic-to-CG mapping and bond topology provided by the user. The output is a set of files ready to be used within the GROMACS simulation package.

PyCGTOOL has been used here to generate parameters for a test set of molecules including lipids, drug molecules and a short peptide. The results show improved performance compared to using a static snapshot with the MARTINI default parameters in the case of small molecules, while also decreasing the work of the user as compared to manual parametrisation by repeated measurement. PyCGTOOL is freely available on GitHub at `https://github.com/jag1g13/pycgtool` and has been registered as doi:10.5281/zenodo.569555.

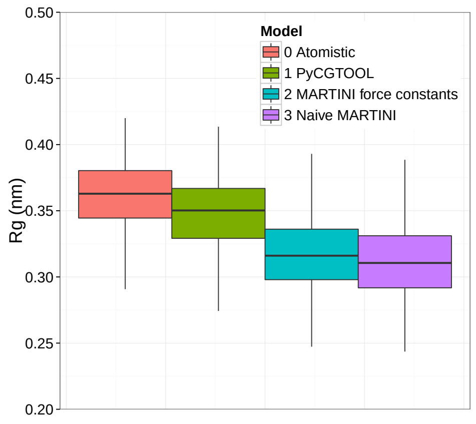


FIGURE 4.12: Radius of gyration for an alanine tetramer. The default PyCGTOOL generated model gives a median radius within the interquartile range of the reference model.

4.9 Software Dependencies

The software requires Python3 with the following packages available from the Python Package Index via *pip*:

- Numpy - http://www.numpy.org/
- simpletraj - https://github.com/arose/simpletraj

The following modules are optional dependencies:

- MDTraj for pseudo-CG XTC output - http://mdtraj.org/1.7.2/
  - Scipy - https://www.scipy.org/
  - Cython - http://cython.org/
- Numba for increased performance - http://numba.pydata.org/
- Sphinx to generate local documentation - http://www.sphinx-doc.org/en/stable/
- A Python testing framework such as Nose2 or py.test

4.10 Input File Formats

The format of the input mapping and bond files is fully described in the documentation at http://pycgtool.readthedocs.io/. Sample files used for the atenolol case are provided:

Mapping file, defining bead name, type and component atoms with residue name in square brackets:

```
[36KB]
N1  P1  N1  C12 C13 C14
O1  P1  O3  C1  C2  C3
O2  SP3 O2  C4
C1  SC3 C5  C6
```

```
C2  SC3 C8  C9
C3  SC3 C7  C10
N2  P5  C11 O1  N2
```

Bond definition file, listing bond lengths and optionally angles to be measured:

```
[36KB]
N1 O1
O1 O2
O2 C1
O2 C2
C1 C2
C1 C3
C2 C3
C3 N2
```

4.11 Parameters of Generated Models

Bond	PyCGTOOL		Default MARTINI		Naive MARTINI	
	eqm.	f. const.	eqm.	f. const.	eqm.	f. const.
B1	0.370 99	8975.978 46	0.370 99	1250	0.396	1250
B2	0.300 44	12 960.932 05	0.300 44	1250	0.314	1250
B3	0.236 41	-	0.236 41	1250	0.236	1250
B4	0.245 16	-	0.245 16	1250	0.239	1250
B5	0.241 37	-	0.241 37	1250	0.243	1250
B6	0.246 49	-	0.246 49	1250	0.250	1250
B7	0.246 36	-	0.246 36	1250	0.250	1250
B8	0.240 47	-	0.240 47	1250	0.248	1250
A1	109.293 03	17.614 61	109.293 03	25	131.33	25
A2	156.319 64	453.896 88	156.319 64	25	170.96	25
A3	123.849 43	130.752 85	123.849 43	25	118.33	25
A4	122.014 16	4604.874 87	122.014 16	25	120.84	25
A5	118.499 38	4341.751 59	118.499 38	25	119.41	25
A6	128.069 03	68.442 58	128.069 03	25	155.59	25
A7	128.989 10	67.572 42	128.989 10	25	117.80	25

TABLE 4.3: Parameters generated for alternate models of Atenolol.

Bond	PyCGTOOL		Default MARTINI		Naive MARTINI	
	eqm.	f. const.	eqm.	f. const.	eqm.	f. const.
B1	0.256 55	-	0.256 55	1250	0.27	1250
B2	0.268 64	28 104.363 91	0.268 64	1250	0.27	1250
B3	0.308 26	46 366.917 23	0.308 26	1250	0.27	1250
B4	0.316 03	7265.650 83	0.316 03	1250	0.33	1250
B5	0.371 56	4197.408 80	0.371 56	1250	0.38	1250
B6	0.484 86	3014.761 22	0.484 86	1250	0.50	1250
A1	131.471 61	78.363 08	131.471 61	25	120	25
A2	133.328 44	57.540 41	133.328 44	25	147	25
A3	135.969 69	38.406 85	135.969 69	25	168	25

TABLE 4.4: Parameters generated for alternate models of Capsaicin.

Bond	PyCGTOOL		Default MARTINI		Orig. MARTINI	
	eqm.	f. const.	eqm.	f. const.	eqm.	f. const.
B1	0.459 34	1059.133 97	0.459 34	1250	0.47	1250
B2	0.450 33	1069.727 76	0.450 33	1250	0.47	1250
B3	0.347 33	1126.240 56	0.347 33	1250	0.37	1250
B4	0.463 33	1175.279 00	0.463 33	1250	0.47	1250
B5	0.448 89	1116.399 29	0.448 89	1250	0.47	1250
B6	0.447 79	1100.749 21	0.447 79	1250	0.47	1250
B7	0.449 66	1104.576 30	0.449 66	1250	0.47	1250
B8	0.452 05	1133.663 48	0.452 05	1250	0.47	1250
B9	0.448 17	1109.521 26	0.448 17	1250	0.47	1250
B10	0.447 87	1099.023 87	0.447 87	1250	0.47	1250
B11	0.449 94	1105.567 86	0.449 94	1250	0.47	1250
A1	104.165 25	10.370 76	104.165 25	25	-	-
A2	111.600 67	35.233 05	111.600 67	25	120	25
A3	144.842 03	101.121 43	144.842 03	25	180	25
A4	112.859 55	19.330 28	112.859 55	25	-	-
A5	88.028 34	31.882 89	88.028 34	25	-	-
A6	146.602 12	114.459 88	146.602 12	25	180	25
A7	146.611 59	106.839 56	146.611 59	25	180	25
A8	143.950 77	86.925 24	143.950 77	25	180	25
A9	146.617 44	108.750 88	146.617 44	25	180	25
A10	146.188 42	103.189 04	146.188 42	25	180	25
A11	143.421 88	83.924 30	143.421 88	25	180	25

TABLE 4.5: Parameters generated for alternate models of DPPC.

Bond	PyCGTOOL eqm.	f. const.	Default MARTINI eqm.	f. const.	Orig. MARTINI eqm.	f. const.
B1	0.378 65	27 968.796 45	0.378 65	1250	0.378	1250
A1	118.628 92	31.343 35	118.628 92	25	111.5	25

TABLE 4.6: Parameters generated for alternate models of Ala4.

Chapter 5

Other Software

5.1 RAMSi

The development of CGTOOL, the predecessor to PyCGTOOL, was planned such that its core components, the Frame class and other supporting classes, could be applied not just to coarse-grain mapping, but to general analysis of trajectories generated by GROMACS. Simulation analysis is a significant part of the computational process with many standard analysis methods being performed on every simulation. Since some such analysis methods are repetitive and require little individual variation these were another target for automation.

RAMSi developed from CGTOOL as an extension for calculating average membrane thickness due to limitations with the existing tool GridMAT-MD [59]. GridMAT is a Perl script which performs membrane thickness and surface area calculations on a grid using GROMACS `.gro` or `.pdb` files as input. There are two major problems with GridMAT which make it unsuitable for analysis of large simulations. Firstly GridMAT is inefficient in terms of disk space as `.gro` and `.pdb` files are plain text files. The `.xtc` format is much more efficient for long MD trajectories, with a filesize 10-15 times smaller, and is the default GROMACS trajectory output format. The trajectory of a typical membrane simulation may be several tens or hundreds of GB when stored in a plain text format. Secondly GridMAT is slow, requiring around a minute to perform a full surface area analysis for a single frame of a typical membrane simulation, dependent on system size. With a typical simulation producing several tens or hundreds of thousands of trajectory frames, this means that to fully analyse an entire trajectory would require weeks so in practice only single frames or small subsets are analysed. Because of this limitation, much of the available data is discarded from the analysis.

RAMSi relies upon the CGTOOL core written in C++ for trajectory input and uses a similar algorithm to GridMAT but is more efficient and parallelised with OpenMP. The essential steps in performing a thickness and surface area analysis using the GridMAT/RAMSi algorithm are:

Some small differences exist in the algorithm between GridMAT and RAMSi. Since it is build on top of the CGTOOL core, RAMSi is able to perform analysis of atomistic, coarse-grained and pseudo-coarse-grained membranes, allowing direct comparison to be made in the evaluation of a new CG model. RAMSi also has experimental support for calculating local Gaussian or mean curvature of a membrane, so to allow highly curved membranes to be sorted into leaflets the sorting process is performed in blocks. One advantage of GridMAT over RAMSi is the ability to identify portions of the bilayer occupied by proteins and account for these in the surface area calculation; this is not yet present in RAMSi, but is planned for the future. The ability to calculate lipid order parameters is also planned.

Algorithm 1 Major points of the GridMAT/RAMSi algorithm

```
1: Find reference particles in each lipid
2: Sort reference particles into upper and lower bilayer
3: for all Frames in trajectory do
4:     for all Grid points do
5:         Find closest reference particle to grid point in upper
6:         Find closest reference particle to grid point in lower
7:         Find z distance between reference particles
8:     end for
9:     for all Reference particles do
10:        Sum area assigned to reference particle
11:    end for
12: end for
```

The major advantage of RAMSi over GridMAT is its speed. By caching the results of the membrane thickness calculation for each lipid, the calculation does not need to be performed for each grid point. The algorithm still scales $\mathcal{O}(n^2)$ with the number of grid points however, as it is still required to find the lipid closest to each point. The improvements to the algorithm, plus the performance of optimised C++ vs Perl mean that RAMSi performs vastly better than GridMAT. For timing comparison: calculation of the lipid surface area and membrane thickness for a single frame of a 128 lipid DPPC bilayer simulation takes 39 seconds with GridMAT on an Intel i7 4790 CPU, whereas RAMSi is able to calculate the same properties at 87 simulation frames per second on a single core or 230 frames per second with four threads in parallel. This is a relative speedup of 3400:1 for the serial case or 9000:1 in parallel. As a result it is entirely possible to use the complete simulation trajectory in the analysis, preventing wasted data. This also allows trends in properties over time to be calculated much more easily and with less uncertainty.

5.2 xtc-length

There is no efficient way to find the number of frames in a GROMACS `.xtc` trajectory using standard GROMACS tools. The recommended method is to use the program `gmx check` to scan through the file and count the number of simulation frames. Since `gmx check` reads every simulation frame into memory for error checking this can require significant amounts of time when being used just to find the simulation length. The program `xtc-length` allows the simulation length to be found more quickly.

The difficulty in counting the number of frames in a GROMACS `.xtc` file is due to the variable compression algorithm used to create it. A single frame in an `.xtc` trajectory is composed of a header and the compressed atomic coordinates, with the degree of compression being dependent on the instantaneous state of the system and thus having a variable frame size. The header contains the number of atoms stored in the frame, the size of the frame in bytes, the simulation timestep and the simulation box size, as well as several other pieces of information.

One simple approach is to estimate the number of frames present by finding the size of the initial frame, and dividing the total file size by this, assuming that all frames are of a similar size. This gives a correct result to within a few percent and is the cheapest possible method.

A more accurate approach is to read the header of every frame in the `.xtc` file and count each. This approach is slower than the filesize estimation approach, but gives an exact number of frames. It is also possible to extract other information from the headers such as the number of atoms and the total simulation time. This is the method used in the tool xtc-length and is typically around three times faster than `gmx check` for the same file when the file has not recently been opened and near instant if a reference to the file persists in RAM.

Chapter 6

Reflections

The invention of the conical flask was probably science. Making one contributes to the science of others.

(Emma Nelson)

6.1 A Coarse-Grained Model of Sugars

The original aim of this research, to develop a coarse-grained model of sugars within the ELBA forcefield, was not completed. Significant work towards the objective was made, though several limitations present in the model at the time of my leaving the project prevent the model from being useful in the research of others.

The most significant of these reasons is the speed of simulations performed using the model, in terms of $\mathrm{ns\,d^{-1}}$. This is partially due to a limitation of the ELBA forcefield. Since its inclusion of point dipoles necessitates the use of LAMMPS, rather than one of the more performant simulation engines such as GROMACS or AMBER, as well as the extra computational work inherent to calculations with the dipoles themselves, a model based upon the ELBA forcefield must be significantly coarse-grained in order to maintain parity with even a united-atom model. Compounding this, the monosaccharide representation used by the model developed here, represented each ring atom with a pair of CG beads: one for the ring atom itself and one to restrain the point dipole. This means that the model required almost as many particles to represent a monosaccharide as the standard united-atom representation and thus competes with models such as the GLYCAM and GROMOS-GLYC parameter sets.

There is a trade-off to be made between a fast simulation and an accurate simulation, which is how the ELBA forcefield is able to maintain relevance against the *de facto* standard MARTINI forcefield. However, the accuracy of the ELBA sugar model as represented in this thesis is not sufficient to justify the loss of speed.

Development of the ELBA sugar model is being continued by others, so it is possible that with significant restructuring these limitations may be mitigated.

6.2 The Automated Generation of Coarse-Grained Models

During the development of the ELBA sugar model, it became clear that the majority of the work, in the sense of time rather than complexity, was taking repeated measurements from reference models and refining parameters in an attempt to match the target distribution. This process was repeated by each researcher in their own work, when using the MARTINI forcefield also.

The creation of the software PyCGTOOL to automate this relied on a single approximation and subsequent manipulation, namely the approximate equivalence of the Boltzmann weighted population of a harmonic oscillator and the normal distribution within the typical range seen within biomolecular simulation. This step, the 'science' component of the conical flask invention, supported the rest of the process, software engineering, the production of many conical flasks.

Though this section of the work was not itself largely science, it is the larger contribution and has been used by others in their own research.

Bibliography

(1) Alder, B. J.; Wainwright, T. E. *The Journal of Chemical Physics* **1959**, *31*, 459.

(2) Verlet, L. *Physical Review* **1967**, *159*, 98–103.

(3) LAMMPS Benchmarks http://lammps.sandia.gov/bench.html.

(4) Salomon-Ferrer, R.; Götz, A. W.; Poole, D.; Le Grand, S.; Walker, R. C. *Journal of Chemical Theory and Computation* **2013**, *9*, 3878–3888.

(5) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. *SoftwareX* **2015**, *1-2*, 19–25.

(6) Brown, W. M.; Wang, P.; Plimpton, S. J.; Tharrington, A. N. *Computer Physics Communications* **2011**, *182*, 898–911.

(7) Sugita, Y.; Kitao, A.; Okamoto, Y. *The Journal of Chemical Physics* **2000**, *113*, 6042.

(8) Laio, A.; Parrinello, M. *Proceedings of the National Academy of Sciences* **2002**, *99*, 12562–12566.

(9) Lindahl; Abraham; Hess; van der Spoel **2019**, DOI: 10.5281/ZENODO.2564761.

(10) Karplus, M.; McCammon, J. A. *Nature Structural Biology* **2002**, *9*, 646.

(11) Marrink, S. J.; de Vries, A. H.; Mark, A. E. *The Journal of Physical Chemistry B* **2004**, *108*, 750–760.

(12) Marrink, S. J.; Risselada, H. J.; Yefimov, S.; Tieleman, D. P.; De Vries, A. H. *Journal of Physical Chemistry B* **2007**, *111*, 7812–7824.

(13) Monticelli, L.; Kandasamy, S. K.; Periole, X.; Larson, R. G.; Tieleman, D. P.; Marrink, S. J. *Journal of Chemical Theory and Computation* **2008**, *4*, 819–834.

(14) López, C. A.; Rzepiela, A. J.; De Vries, A. H.; Dijkhuizen, L.; Hünenberger, P. H.; Marrink, S. J.; López, C. A.; Rzepiela, A. J.; De Vries, A. H.; Dijkhuizen, L.; Hünenberger, P. H.; Marrink, S. J. *Journal of Chemical Theory and Computation* **2009**, *5*, 3195–3210.

(15) Arnarez, C.; Uusitalo, J. J.; Masman, M. F.; Ingólfsson, H. I.; de Jong, D. H.; Melo, M. N.; Periole, X.; de Vries, A. H.; Marrink, S. J. *Journal of Chemical Theory and Computation* **2014**, *11*, 260–275.

(16) Marrink, S. J.; Tieleman, D. P. *Chemical Society reviews* **2013**, *42*, 6801–6822.

(17) Reith, D.; Meyer, H.; Müller-Plathe, F. *Macromolecules* **2001**, *34*, 2335–2345.

(18) Ercolessi, F.; Adams, J. B. *Europhysics Letters* **1994**, *26*, 583–588.

(19) Plimpton, S. *Journal of Computational Physics* **1995**, *117*, 1–19.

(20) Stockmayer, W. H. *The Journal of Chemical Physics* **1941**, *9*, 863–870.

(21) Orsi, M.; Essex, J. W. *Faraday Discussions* **2013**, 1–24.

(22) Orsi, M.; Essex, J. W. *PLoS ONE* **2011**, *6*, e28637.

(23) Orsi, M.; Ding, W.; Palaiokostas, M. *Journal of Chemical Theory and Computation* **2014**, *10*, 4684–4693.

(24) Genheden, S.; Essex, J. W. *Journal of Chemical Theory and Computation* **2015**, *11*, 4749–4759.

(25) Zhu, Y. P.; Zajicek, J.; Serianni, A. S. *Journal Of Organic Chemistry* **2001**, *66*, 6244–6251.

(26) Cremer, D.; Pople, J. A. *Journal of the American Chemical Society* **1975**, *97*, 1354–1358.

(27) Pol-Fachin, L.; Rusu, V. H.; Verli, H.; Lins, R. D. *Journal of Chemical Theory and Computation* **2012**, *8*, 4681–4690.

(28) Stortz, C. A.; Johnson, G. P.; French, A. D.; Csonka, G. I. *Carbohydrate Research* **2009**, *344*, 2217–2228.

(29) Kirschner, K. N.; Yongye, A. B.; Tschampel, S. M.; González-Outeiriño, J.; Daniels, C. R.; Foley, B. L.; Woods, R. J. *Journal of Computational Chemistry* **2008**, *29*, 622–655.

(30) López, C. A.; Sovova, Z.; van Eerden, F. J.; de Vries, A. H.; Marrink, S. J. *Journal of Chemical Theory and Computation* **2013**, *9*, 1694–1708.

(31) Rusu, V. H.; Baron, R.; Lins, R. D. *Journal of Chemical Theory and Computation* **2014**, *10*, 5068–5080.

(32) Molinero, V.; Goddard, W. a. *J. Phys. Chem. B* **2004**, *108*, 1414–1427.

(33) Saka, S. K.; Honigmann, A.; Eggeling, C.; Hell, S. W.; Lang, T.; Rizzoli, S. O. *Nature Communications* **2014**, *5*, 14554–14559.

(34) Caroff, M.; Karibian, D.; Cavaillon, J. M.; Haeffner-Cavaillon, N. *Microbes and Infection* **2002**, *4*, 915–926.

(35) Erridge, C.; Bennett-Guerrero, E.; Poxton, I. R. *Microbes and Infection* **2002**, *4*, 837–851.

(36) Ma, H.; Irudayanathan, F. J.; Jiang, W.; Nangia, S. *The Journal of Physical Chemistry B* **2015**, *119*, 14668–14682.

(37) Graham, J. A.; Essex, J. W.; Khalid, S. *Journal of Chemical Information and Modeling* **2017**, *57*, 650–656.

(38) Kirschner, K. N.; Woods, R. J. *Proceedings of the National Academy of Sciences of the United States of America* **2001**, *98*, 10541–10545.

(39) Mikhail, S. Z.; Kimel, W. R. *Journal of Chemical & Engineering Data* **1961**, *6*, 533–537.

(40) Perry, R., *Chemical engineers' handbook*, 5th ed.; McGraw-Hill: 1973.

(41) Comesaña, J. F.; Otero, J. J.; García, E.; Correa, A. *Journal of Chemical & Engineering Data* **2003**, *48*, 362–366.

(42) Shiraga, K.; Suzuki, T.; Kondo, N.; Tajima, T.; Nakamura, M.; Togo, H.; Hirata, A.; Ajito, K.; Ogawa, Y. *The Journal of Chemical Physics* **2015**, *142*, 234504.

(43) Sousa da Silva, A. W.; Vranken, W. F. *BMC research notes* **2012**, *5*, 367.

(44) Samways, M. ELBA tools https://github.com/marleysamways/ELBA., 2015.

(45) Graham, J. A. dipoleAnalysis https://github.com/jag1g13/dipoleAnalysis., 2015.

(46) Csonka, G. I.; Kaminsky, J. *Journal of Chemical Theory and Computation* **2011**, *7*, 988–997.

(47) Tribello, G. A.; Bonomi, M.; Branduardi, D.; Camilloni, C.; Bussi, G. *Computer Physics Communications* **2014**, *185*, 604–613.

(48) Wilson, G.; Aruliah, D. A.; Brown, C. T.; Chue Hong, N. P.; Davis, M.; Guy, R. T.; Haddock, S. H. D.; Huff, K. D.; Mitchell, I. M.; Plumbley, M. D.; Waugh, B.; White, E. P.; Wilson, P. *PLoS Biology* **2014**, *12*, e1001745.

(49) Wilson, G.; Bryan, J.; Cranston, K.; Kitzes, J.; Nederbragt, L.; Teal, T. K. **2016**.

(50) McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P. P.; Lane, T. J.; Pande, V. S. *Biophysical Journal* **2015**, *109*, 1528–1532.

(51) Izvekov, S.; Voth, G. A. *The Journal of Physical Chemistry B* **2005**, *109*, 2469–2473.

(52) Tschöp, W.; Kremer, K.; Batoulis, J.; Bürger, T.; Hahn, O. *Acta Polymerica* **1998**, *49*, 61–74.

(53) Bereau, T.; Kremer, K. *Journal of Chemical Theory and Computation* **2015**, *11*, 2783–2791.

(54) McKiernan, K. A.; Wang, L.-P.; Pande, V. S. *Journal of Chemical Theory and Computation* **2016**, *12*, 5960–5967.

(55) Lindorff-Larsen, K.; Trbovic, N.; Maragakis, P.; Piana, S.; Shaw, D. E. *Journal of the American Chemical Society* **2012**, *134*, 3787–3791.

(56) Oostenbrink, C.; Villa, A.; Mark, A. E.; Van Gunsteren, W. F. *Journal of Computational Chemistry* **2004**, *25*, 1656–1676.

(57) Hanson, S. M.; Newstead, S.; Swartz, K. J.; Sansom, M. S. *Biophysical Journal* **2015**, *108*, 1425–1434.

(58) Koziara, K. B.; Stroet, M.; Malde, A. K.; Mark, A. E. *Journal of Computer-Aided Molecular Design* **2014**, *28*, 221–233.

(59) Allen, W. J.; Lemkul, J. A.; Bevan, D. R. *Journal of Computational Chemistry* **2009**, *30*, 1952–1958.

(60) Duan, Y.; Wu, C.; Chowdhury, S.; Lee, M. C.; Xiong, G.; Zhang, W.; Yang, R.; Cieplak, P.; Luo, R.; Lee, T.; Caldwell, J.; Wang, J.; Kollman, P. *Journal of Computational Chemistry* **2003**, *24*, 1999–2012.

(61) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *The Journal of Chemical Physics* **1983**, *79*, 926.