PyCGTOOL: Automated Generation of Coarse-grained Molecular Dynamics Models from Atomistic Trajectories.

Journal:	*Journal of Chemical Information and Modeling*
Manuscript ID	ci-2017-00096t.R1
Manuscript Type:	Application Note
Date Submitted by the Author:	n/a
Complete List of Authors:	Graham, James; University of Southampton, Chemistry Essex, Jonathan; University of Southampton, School of Chemistry Khalid, Syma; University of Southampton, Chemistry

PyCGTOOL: Automated Generation of Coarse-grained Molecular Dynamics Models from Atomistic Trajectories

James A. Graham, Jonathan W. Essex, and Syma Khalid*

School of Chemistry, University of Southampton, Hampshire, SO17 1BJ, United Kingdom

E-mail: S.Khalid@soton.ac.uk

Abstract

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 we present PyCGTOOL 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.

The software is available from the URL `https://github.com/jag1g13/pycgtool` as doi:10.5281/zenodo.259330.

Introduction

In recent years there has been a steep rise in the popularity of coarse-grained (CG) molecular dynamics (MD) simulations within the biomolecular simulation community. Perhaps the most popular CG forcefield is MARTINI,[1] having been employed to study: lipids,[2] proteins,[3] nucleic acids[4,5] and nano-materials.[6,7] The MARTINI framework consists of a set of predefined particles with a range of polarities, allowing the user to bypass the difficult manual parametrisation of the non-bonded Lennard-Jones terms when creating a model of a new molecule, by simply adopting the pre-existing parameter set. Additionally, MARTINI provides a recommended set of equilibrium values for bond lengths and force constants for both bond lengths and angles. In the simplest cases all that then remains is to determine appropriate particle types, bond topology, and angle equilibrium values, to give an adequate representation of the target molecule. In practice, the default values are applicable only to molecules similar to those from which the parameters were generated, so equilibrium values for bond lengths must also be generated on a molecule-by-molecule basis. Default force constants however, are regularly used in user-generated models, for example as in Ma *et al.*[8]

The construction of a CG model of a novel molecule is thus 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.

Here we present PyCGTOOL, a tool for the automated generation of bonded parameters for CG models within the MARTINI force field or as a part of a more complex forcefield such as the ELBA[9] (ELectrostatics BAsed) 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 used in this paper 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 we do not intend to replace existing MARTINI parametrisations, but rather to reduce the difficulty and effort in studying a molecule which does not yet have a MARTINI parametrisation. We particularly anticipate this to be useful in the simulation of small, drug-like molecules.

Workflow

PyCGTOOL formalises the workflow (fig 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.

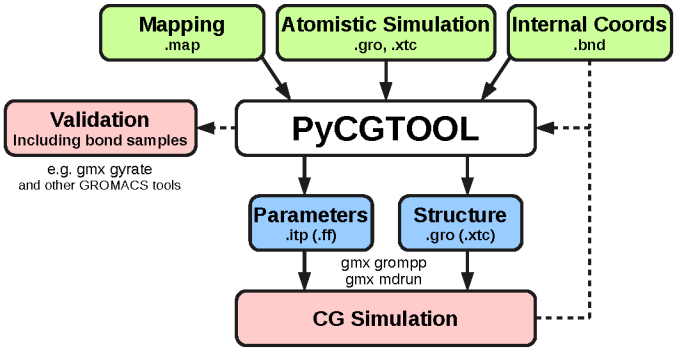

Figure 1: The PyCGTOOL workflow. Green: An atomistic simulation trajectory is fed into PyCGTOOL, along with the atomistic-to-CG mapping (as *.map* and *.bnd* files). Blue: Output is a calculated parameter set and system structure. Red: Initial validation may be performed by passing a CG trajectory through PyCGTOOL once more to collect samples of internal coordinate values.

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 in the supplementary in-

formation 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[10] *.gro* and *.xtc* formats, or other standard formats if using the optional MDTraj library.[11]

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 $1250\,\mathrm{kJ\,mol^{-1}\,nm^{-2}}$ for bond lengths and $25\,\mathrm{kJ\,mol^{-1}}$ 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 $100{,}000\,\mathrm{kJ\,mol^{-1}\,nm^{-2}}$, although this is again 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.

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 paper 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.

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,[12] 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.

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[13] 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 as described in the supplementary information. 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 provided. 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[14] unit system.

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.

The method of parameter generation in PyCGTOOL differs from the existing *Auto_MARTINI*[15] 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[16] 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.[17] A relatively rigid molecule may require only a few tens of nanoseconds of atomistic simulation for parameters to converge, whereas a more flexi-

ble 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. 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 preprocessing 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[18] united-atom forcefield, but also with the AMBER-based GLYCAM06[19] fully atomistic carbohydrate model in the generation of parameters for the ELBA CG forcefield.

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 the supplementary information as well as the capsaicin and polyalanine results.

All simulations were performed using GROMACS 2016 (details provided in supplementary information), with the exception of the capsaicin reference simulation for which data was provided by the Sansom group.[20]

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.

Drug-like Molecules

The two drug-like molecules tested were capsaicin, with topology and simulation trajectory provided by the Sansom group,[20] and atenolol, using the GROMOS 54A7 forcefield taken from the ATB database.[21]

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

The Tukey boxplots shown in fig 2 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 3, 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.

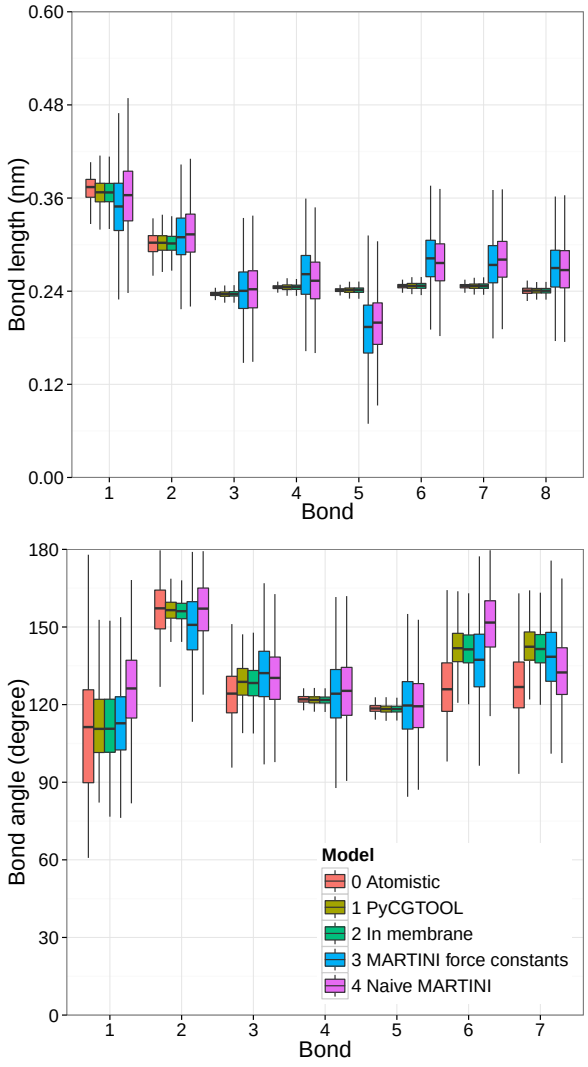

Figure 2: Tukey boxplots showing bond length and angle distributions for the molecule atenolol . Note here that default force constants differ significantly from the atomistic reference, producing much wider distributions for many of the bonds. For angle seven in the bottom panel we note that the model using a calculated force constant is slightly less accurate in median and interquartile range.

Capsaicin

Similar boxplots are presented in the supplementary information for the capsaicin validation. The default PyCGTOOL generated model is again seen to closely replicate bond lengths, particulary those within the aromatic ring. In the angle distributions there is little dif-

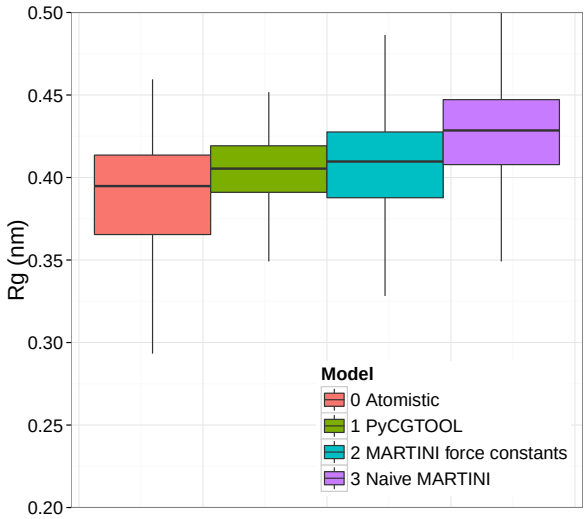

Figure 3: Radius of gyration for atenolol. The default PyCGTOOL generated model gives a median radius within the interquartile range of the reference atomistic model.

ference 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.

Lipid: DPPC

The DPPC 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 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 $25\,\mathrm{kJ\,mol^{-1}}$, 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 *CG-TOOL* (available at `https://bitbucket.org/jag1g13/cgtool`) using a similar algorithm to the *GridMAT-MD*[22] 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.

Polymer: Polyalanine

A short four-residue strand of polyalanine was simulated using the AMBER03 forcefield. As a test of the polymer functionality of PyCGTOOL, each alanine residue was mapped to a single bead and a GROMACS residue topology (*.rtp*) file was output, defining a single residue and its inter-residue bonds, which allows the use of the standard GROMACS tool *gmx pdb2gmx* to build the complete topology for the short polyalanine strand.

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 5 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

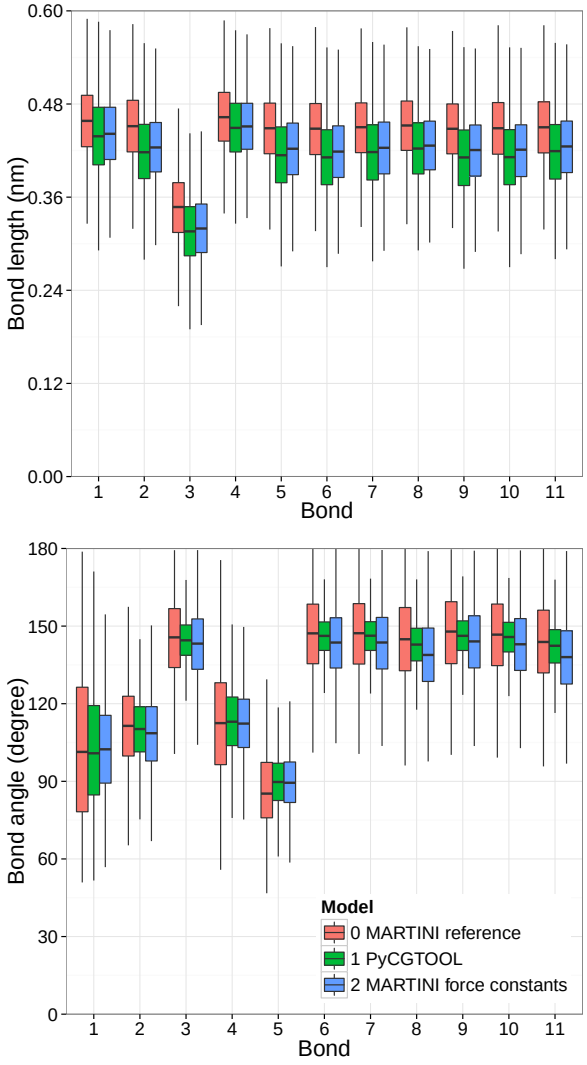

Figure 4: Bond length and angle distributions for the lipid DPPC as compared to a standard MARTINI reference simulation. Note here that the default force constant produces an accurate distribution as lipids are the class of molecules for which they were designed.

models, indicating that the default MARTINI force constant is satisfactory in this case. Radius of gyration in fig 6 also shows that the model generated using default settings provides the best agreement with the reference model.

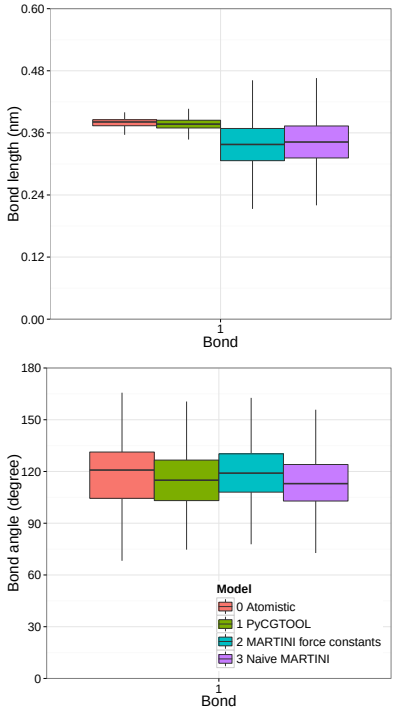

Figure 5: Tukey boxplots showing bond length and angle distributions for an alanine tetramer. All three bond lengths in the tetramer are considered equivalent, so are represented by a single box, as are both bond angles.

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. Uniquely, 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

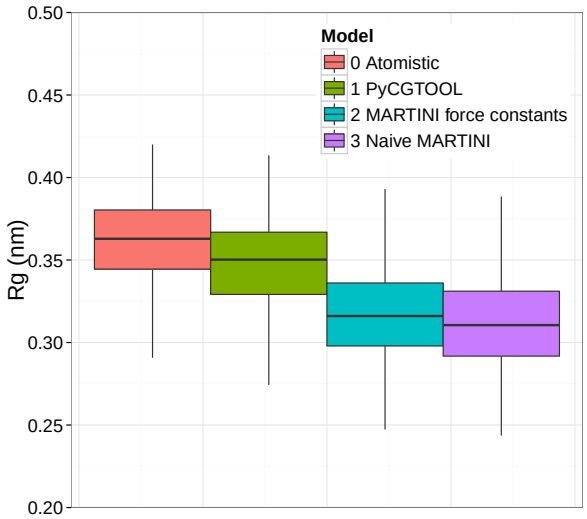

Figure 6: Radius of gyration for an alanine tetramer. The default PyCGTOOL generated model gives a median radius within the interquartile range of the reference model.

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.

In the this work we have used PyCGTOOL to generate parameters for a test set of molecules including lipids, drug molecules and a short peptide (present in SI). 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. Thus we present a freely available code to easily generate accurate coarse-grain models from their atomistic counterparts.

Acknowledgement JAG acknowledges funding from the EPSRC via the ICSS DTC (grant code (EP/G03690X/1), assistance in testing during development from Pin-Chia Hsu and Damien Jefferies (University of Southampton), and the reference simulation trajectory for capsaicin from Phill Stansfield and Nick Michelarakis (University of Oxford).

Supporting Information Available: This material is available free of charge via the Internet at `http://pubs.acs.org/`.

References

(1) Marrink, S. J.; Tieleman, D. P. Perspective on the Martini Model. *Chem. Soc. Rev.* **2013**, *42*, 6801–6822.

(2) Ingólfsson, H. I.; Melo, M. N.; van Eerden, F. J.; Arnarez, C.; Lopez, C. A.; Wassenaar, T. A.; Periole, X.; de Vries, A. H.; Tieleman, D. P.; Marrink, S. J. Lipid Organization of the Plasma Membrane. *J. Am. Chem. Soc.* **2014**, *136*, 14554–14559.

(3) Bond, P. J.; Holyoake, J.; Ivetac, A.; Khalid, S.; Sansom, M. S. P. Coarse-Grained Molecular Dynamics Simulations of Membrane Proteins and Peptides. *J. Struct. Biol.* **2007**, *157*, 593–605.

(4) Corsi, J.; Hawtin, R. W.; Ces, O.; Attard, G. S.; Khalid, S. DNA Lipoplexes: Formation of the Inverse Hexagonal Phase Observed by Coarse-Grained Molecular Dynamics Simulation. *Langmuir* **2010**, *26*, 12119–12125.

(5) Khalid, S.; Bond, P. J.; Holyoake, J.; Hawtin, R. W.; Sansom, M. S. DNA and Lipid Bilayers: Self-Assembly and Insertion. *J. R. Soc. Interface* **2008**, *5*, 241–250.

(6) Wallace, E. J.; Sansom, M. S. P. Carbon Nanotube Self-Assembly with Lipids and Detergent: A Molecular Dynamics Study. *Nanotechnology* **2009**, *20*, 045101.

(7) Wong-Ekkabut, J.; Baoukina, S.; Triampo, W.; Tang, I.-M.; Tieleman, D. P.; Monticelli, L. Computer Simulation Study of Fullerene Translocation Through Lipid Membranes. *Nat. Nanotechnol.* **2008**, *3*, 363–368.

(8) Ma, H.; Irudayanathan, F. J.; Jiang, W.; Nangia, S. Simulating Gram-Negative Bacterial Outer Membrane: A Coarse Grain Model. *J. Phys. Chem. B* **2015**, *119*, 14668–14682.

(9) Genheden, S.; Essex, J. W. A Simple and Transferable All-Atom/Coarse-Grained Hybrid Model to Study Membrane Processes. *J. Chem. Theory Comput.* **2015**, *11*, 4749–4759.

(10) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High Performance Molecular Simulations Through Multi-Level Parallelism from Laptops to Supercomputers. *SoftwareX* **2015**, *1-2*, 19–25.

(11) McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hern??ndez, C. X.; Schwantes, C. R.; Wang, L.-P. P.; Lane, T. J.; Pande, V. S.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P. P.; Lane, T. J.; Pande, V. S. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophys. J.* **2015**, *109*, 1528–1532.

(12) Izvekov, S.; Voth, G. A. A Multiscale Coarse-Graining Method for Biomolecular Systems. *J. Phys. Chem. B* **2005**, *109*, 2469–2473.

(13) Tschöp, W.; Kremer, K.; Batoulis, J.; Bürger, T.; Hahn, O. Simulation of Polymer Melts. I. Coarse-Graining Procedure for Polycarbonates. *Acta Polym.* **1998**, *49*, 61–74.

(14) Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comput. Phys.* **1995**, *117*, 1–19.

(15) Bereau, T.; Kremer, K. Automated Parametrization of the Coarse-Grained Martini Force Field for Small Organic Molecules. *J. Chem. Theory Comput.* **2015**, *11*, 2783–2791.

(16) McKiernan, K. A.; Wang, L.-P.; Pande, V. S. Training and Validation of a Liquid-Crystalline Phospholipid Bilayer Force Field. *J. Chem. Theory Comput.* **2016**, *12*, 5960–5967.

(17) Lindorff-Larsen, K.; Trbovic, N.; Maragakis, P.; Piana, S.; Shaw, D. E. Structure and Dynamics of an Unfolded Protein Examined by Molecular Dynamics Simulation. *J. Am. Chem. Soc.* **2012**, *134*, 3787–3791.

(18) Oostenbrink, C.; Villa, A.; Mark, A. E.; Van Gunsteren, W. F. A Biomolecular Force Field Based on the Free Enthalpy of Hydration and Solvation: The GROMOS Force-Field Parameter Sets 53A5 and 53A6. *J. Comput. Chem.* **2004**, *25*, 1656–1676.

(19) Kirschner, K. N.; Yongye, A. B.; Tschampel, S. M.; González-Outeiriño, J.; Daniels, C. R.; Foley, B. L.; Woods, R. J. GLYCAM06: A Generalizable Biomolecular Force Field. Carbohydrates. *J. Comput. Chem.* **2008**, *29*, 622–655.

(20) Hanson, S. M.; Newstead, S.; Swartz, K. J.; Sansom, M. S. Capsaicin Interaction with TRPV1 Channels in a Lipid Bilayer: Molecular Dynamics Simulation. *Biophys. J.* **2015**, *108*, 1425–1434.

(21) Koziara, K. B.; Stroet, M.; Malde, A. K.; Mark, A. E. Testing and Validation of the Automated Topology Builder (ATB) Version 2.0: Prediction of Hydration Free Enthalpies. *J. Comput. Aided. Mol. Des.* **2014**, *28*, 221–233.

(22) Allen, W. J.; Lemkul, J. A.; Bevan, D. R. GridMAT-MD: A Grid-Based Membrane Analysis Tool for Use with Molecular Dynamics. *J. Comput. Chem.* **2009**, *30*, 1952–1958.

Graphical TOC Entry