Moltemplate: A Tool for Coarse-Grained Modeling of Complex Biological Matter and Soft Condensed Matter Physics

Andrew I. Jewett[1], David Stelter[2], Jason Lambert[3], Shyam M. Saladi[4], Otello M. Roscioni[5], Matteo Ricci[5], Ludovic Autin[1], Martina Maritan[1], Saeed M. Bashusqeh[6], Tom Keyes[2], Remus T. Dame[7], Joan-Emma Shea[8], Grant J. Jensen[4,9], David S. Goodsell[1,10]

1 Department of Integrative Structural and Computational Biology, The Scripps Research Institute, La Jolla, CA, USA

2 Department of Chemistry, Boston University, MA, USA

3 Department of Chemistry, University of Tennessee, Knoxville, TN, USA

4 Division of Biology and Biological Engineering, California Institute of Technology, Pasadena, CA, USA

5 MaterialX LTD, Bristol, UK

6 School of Mechanical Engineering, College of Engineering, University of Tehran, Tehran, Iran

7 Leiden Institute of Chemistry, Leiden University, Leiden, Netherlands

8 Departments of Chemistry and Biochemistry and Physics, University of California, Santa Barbara, CA, USA

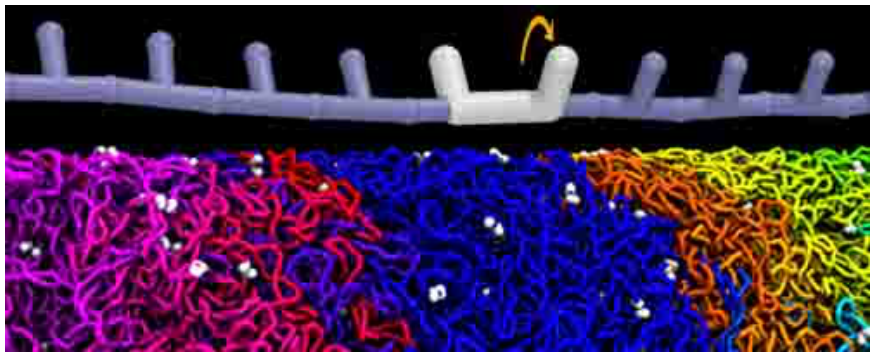
9 Howard Hughes Medical Institute, California Institute of Technology, Pasadena, CA, USA

10 RCSB Protein Data Bank and Institute for Quantitative Biomedicine, Rutgers, the State University of New Jersey, Piscataway, NJ, USA

Corresponding Authors:

Andrew I. Jewett jewett.aij@gmail.com, David S. Goodsell goodsell@scripps.edu

GRAPHICAL ABSTRACT



ABSTRACT

Coarse-grained models have long been considered indispensable tools in the investigation of biomolecular dynamics and assembly. However, the process of simulating such models is arduous because unconventional force fields and particle attributes are often needed, and some systems are not in thermal equilibrium. Although modern molecular dynamics programs are highly adaptable, software designed for *preparing* all-atom simulations typically makes restrictive assumptions about the nature of the particles and the forces acting on them. Consequently, the use of coarse-grained models has remained challenging. Moltemplate is a file format for storing coarse-grained molecular models and the forces that act on them, as well as a program that converts moltemplate files into input files for LAMMPS, a popular molecular dynamics engine. Moltemplate has broad scope and an emphasis on generality. It accommodates new kinds of forces as they are developed for LAMMPS, making it a popular tool with thousands of users in computational chemistry, materials science, and structural biology. To demonstrate its wide functionality, we provide examples of using moltemplate to prepare simulations of fluids using many-body forces, coarse-grained organic semiconductors, and the motor-driven supercoiling and condensation of an entire bacterial chromosome.

Keywords: coarse-grained simulation, molecular modeling, LAMMPS, molecular dynamics

INTRODUCTION

Simulation methods are currently used to explore biological systems from atoms to entire organisms and beyond. Given current computational capabilities, the representation must be tailored to the scale level being studied. Detailed quantum mechanical and molecular mechanics simulations explore the details of covalent bonding, molecular geometry and interaction, and other sub-molecular properties [1]. Molecular dynamics simulations are widely employed to look at the dynamic properties of entire biomolecules and their complexes [2], effectively providing a “computational microscope” to probe biomolecular processes [3,4]. Moving to larger systems and/or longer time scales often requires coarse-graining methods that represent collections of atoms with simpler primitives to make the simulation computationally tractable. For example, popular force fields such as MARTINI [5] replace individual amino acids with a small number of spheres, and are widely used to explore protein folding and other dynamic processes. Currently, even simpler representations, where individual beads represent multiple amino acids/nucleotides, domains, subunits or entire molecules, become necessary when simulating even larger systems such as entire genomes or whole cells [6]. These “coarse-grained” representations have many advantages: they form a bridge between the atomic properties of molecules and the continuum properties of cell biology and materials science. Some coarse-grained models are designed with a small number of parameters, making it easier to pinpoint the aspects of the model or force field that lead to emergent functional properties [7]. Some coarse-grained models are carefully parameterized, matching the placement of beads, force-fields parameters, and (if applicable) internal state transition rates, with atomistic simulation and experimental data [6,8–12].

The loss of resolution during coarse-graining often results in the design and use of complex and unconventional force fields not available in conventional molecular modeling software.

Consequently, these coarse-grained representations are often designed and implemented in a one-off manner, typically by writing large custom simulation programs to address a system of interest. For example, REMODELER is a specialized tool written in 20000 lines of FORTRAN for simulating the growth of bacterial cell walls [13]. Similarly, POLYCHROM (https://zenodo.org/record/3579473) is a tool for building coarse-grained simulations of eukaryotic chromosomes. Specialized tools like these typically require a large investment of skill and labor for each system that the investigators wish to study.

Numerous tools exist for building fully atomistic simulations [14–16], however, tools for specifying coarse-grained simulations are not as widely available. Specialized tools are available for building simulations of membranes and proteins using the MARTINI force field, including PyCGTool, a method for adding new molecules to MARTINI [17], MERMAID [18] and CHARMM-GUI Martini Maker [19]. Several general tools are also available with goals similar to the work described here. For example, TopoTools (https://doi.org/10.5281/zenodo.3845031) is a VMD plugin that streamlines access to LAMMPS. mBUILD [20] and ParmEd (http://parmed.github.io/ParmEd) provide different APIs allowing Python programmers to prepare (typically all-atom) simulations using OpenMM [21], a powerful molecular dynamics engine.

Moltemplate is an intuitive, compact, user-readable file format (the LT “LAMMPS-Template” format) for storing coarse-grained molecular models that can be simulated in LAMMPS [22]. The moltemplate software converts these moltemplate files into files that may be run directly in LAMMPS with minimal editing. The moltemplate file format was conceived with several goals in mind: to be a general as possible, to streamline coarse-grained modeling for new and expert users, and to provide an easy way to store and document coarse-grained simulations to improve reproducibility. This file format is designed to be easily extensible, compatible with

other molecular building programs, and to simplify the design of custom coarse-grained representations of chemical and biological systems, from single biomolecules to entire cells, such as the bacterial chromosome and liposome included in Figure 1. A simple example of a moltemplate file is shown in Figure 2. Users can combine a variety of point-like, rigid, and continuum-field/atomistic hybrid representations. Building on the generality of the LT format, moltemplate has also been widely used for building complex atomistic simulations.

Note that coarse-grained models are only as good as the force fields being employed. Extensive work is required to choose particle placement, force-field parameters, and (if applicable) transition rates for internal state changes [6,8–10]. Moltemplate does not solve these critically important problems; rather, moltemplate is a tool for storing and using coarse-grained models once they are developed. This report presents the basic concept of moltemplate, the LT file format, and several applications to demonstrate the scope of simulations that are possible.

METHODS

Overview of moltemplate methods. Once users have created moltemplate files describing a system they want to simulate, the files must be converted into files that LAMMPS understands. Moltemplate does this using a collection of Python scripts for generating text files, generating coordinates, and assigning force field parameters. Moltemplate also includes LAMMPS-specific tools for parsing and converting LAMMPS files and converting arbitrary curves into polymers in LAMMPS format. These Python scripts can be run independently, however the majority of them communicate with each other through a BASH script (moltemplate.sh), which provides the main user interface.

Moltemplate is built around a template-based strategy that streamlines ease of use and extensibility of the method. In order to run a molecular simulation, a user needs to: 1) choose

the initial state of a system (including the positions of the particles that comprise the system and any other degrees of freedom), and 2) describe the interactions (forces or energies) between these particles. For systems that are driven out of thermal equilibrium, users must also: 3) control how these forces evolve over time. The LT file language is designed to store these object definitions, force fields, and process descriptions. The template-based strategy allows users to define subassemblies in small, manageable LT files, and then duplicate and combine them hierarchically to generate larger systems of interest.

Moltemplate duplicates blocks of text in LT files, substituting user-defined counters as new copies are created. Typically, these text blocks fall into four or five types, defining: 1) particles ("atoms"), 2) bonds between particles, 3) higher-order interactions between particles such as angles and dihedrals, 4) the parameters needed to define or modify these interactions during a simulation, and (in some cases), 5) rules explaining the circumstances when they are to be applied. Text blocks are written directly in the file formats used by LAMMPS. Within these text blocks, counter variables are defined that will be incremented as the text blocks are duplicated when molecules are added to the system. This generates output files with a full enumeration of the atom, bond, interaction and force field information. A simple example is presented in Figure 2a, with four "write" statements that create files with the first four types of information. Variables with "$" represent individual atoms or bonds and will be incremented each time the molecule it belongs to is duplicated, whereas variables with "@" represent types and will not be incremented. Moltemplate includes a rich language for controlling the position and orientation of individual molecules (eg. Figure 2c) and customizing them (Supplementary Figure 1). Similar commands can be used to generate polymers and other assemblies of arbitrary shape and connectivity. More complex geometries can be generated using external tools like PACKMOL or our polymer generator "genpoly_lt.py" and imported into moltemplate (eg. using "moltemplate.sh system.lt -xyz coords.xyz"). A summary of moltemplate commands is included

in Supplementary Table 1 and the documentation (http://moltemplate.org/doc). Detailed examples for all figures in this paper are included at http://doi.org/10.5281/zenodo.4392267. More examples are available online (http://moltemplate.org/examples.html), including coarse-grained polymer melts, proteins, lipid membranes, and examples with custom force fields.

Moltemplate provides a file format (documented in the moltemplate manual) that advanced moltemplate users can use to store force field parameters, as well as rules that describe when these forces should be applied. These force-field rules can automatically generate angle, dihedral, or improper interactions between atoms whose type and bond connectivity matches a user-supplied template. This file format is flexible enough to describe the many kinds of forces that LAMMPS supports. A wide variety of popular force fields have been converted into moltemplate format (including OPLSAA, AMBER, COMPASS, DREIDING, and MARTINI). However, moltemplate does *not* infer force-field-specific atom types or calculate partial charges.

Coarse-grained representations often present an additional challenge: they may include custom particle attributes and/or move under the influence of unconventional forces, such as the many-body "mW" water particle used in Figure 2b [23], as well as forces that drive the system out of equilibrium. A diverse and growing list of methods exist for running non-equilibrium simulations in LAMMPS, such as time-varying external forces, motors used to twist DNA in Figure 1ab (explained in the supplemental data), the collective motion of self-propelled colloidal particles (https://lammps.sandia.gov/doc/fix_propel_self.html), and chemical reactions [24]**.** The data required to characterize these non-standard forces is often encoded in domain-specific file formats that are not supported in traditional molecule builder software. In fact, the majority of the hundreds of force field styles and features that are available in LAMMPS have been created by LAMMPS *users*, and each of these features is controlled by text commands invented by different users. Any program that attempts to generate text files for *all* of these different formats

will need to use some form of template substitution. This is what moltemplate does: it has wide applicability because (for most files) it is intentionally ignorant about the format or structure of the text files that it creates. This allows it to make nearly any kind of text file that LAMMPS can read. By keeping the design of moltemplate simple and general, our goal is that moltemplate will also retain utility for the diverse kinds of files that future coarse-grained modelers will need.

Moltemplate is primarily used with LAMMPS, but it can be customized to prepare files for other molecular dynamics engines by customizing the ~2000 lines of code that process the format and syntax of “Data” sections of moltemplate files. For example, we have also implemented moltemplate tools for Espresso (http://moltemplate.org/espresso).

RESULTS

Sample applications. Moltemplate has been used in a wide range of applications in materials science and biology. To introduce new users to moltemplate and LAMMPS, progressively-complex examples are available on the moltemplate site (https://moltemplate.org), ranging from simple all-atom simulations of small molecules in periodic boxes to a coarse-grained simulation of a liposome with protein inclusions (Figure 1f). To demonstrate what is possible with moltemplate, files for running several advanced examples have been deposited at Zenodo (http://doi.org/10.5281/zenodo.4392267), along with the moltemplate code and a distribution of LAMMPS that includes several modifications needed by the examples. The examples are described briefly below, and in more detail in the Supplementary Text File.

Simulating DNA superhelicity. DNA is a particularly challenging topic for study, given the need to capture the topology of the double-helical arrangement of strands, the specific interaction between strands, and the complex stiffness and non-uniform bending properties of the helix. Moltemplate provides tools to create DNA models of arbitrary detail and sophistication

including OxDNA2 [25,26]. Figure 1a shows a Kratky-Porod-like polymer model of DNA [27,28] containing a twist motor, with three beads representing 42bp and with dummy atoms to represent the local superhelical twist of the chain.

The example included with this manuscript explores supercoiling of an entire bacterial chromosome (Figure 1b). This simulation demonstrates that supercoiling caused by enzymes like DNA gyrase is enough to generate compact, spatially segregated circular bottlebrush-like structures seen in prokaryotic chromosomes such as *Caulobacter crescentus* [29]. Early in the process of cell division, the new origin of replication is transported from one end of the cell by the ParABS system and anchored to the opposite pole and the small amount of DNA that has been replicated is stretched across the length of the cell (Supplementary Figure 5). The stretched DNA gradually relaxes as replication continues and more DNA is supplied to fill the space [30]. To approximate this, we simulate a 4Mbp ring of circular DNA that is initially stretched until it is straight. 400 twist motors are inserted evenly along the length of the DNA. (For details how add control twist motors using moltemplate, see the Supplemental text file.) During the simulation they maintain torsional tension throughout the polymer. Then the DNA is slowly relaxed, roughly mimicking the process of DNA relaxation after replication in *C. crescentus*. In this example, the branched plectonemic supercoils (up to ~15kb in length) form naturally under tension as the DNA relaxes.

Particles with custom attributes. Mesoscale modeling requires aggressive coarse-graining. As the size of each coarse-grained particle grows to encompass more atoms, the state of the system is no longer adequately described by those particles' X,Y,Z coordinates. Additional degrees of freedom must be added to each particle to compensate for the information that was discarded [8,10]. Recent "ultra-coarse-grained" models contain particles with internal states that change over time [10,11] and LAMMPS can simulate chemical reactions that alter particles'

types over time (see Future Directions). LAMMPS also allows particles to have directional attributes as well as other custom attributes created by the user. In the MOLC model [31], large organic molecules are represented in terms of ellipsoidal beads connected by directional bonds. Moltemplate format is used for storing the force-field parameters, ellipsoid shape and orientation, and bond topology of each coarse-grained molecule in a compact and human-readable way, and for constructing assemblies from these molecules for simulation in LAMMPS. The supplemental files include MOLC models of simple organic molecules and organic semiconductors, prepared using moltemplate.

DISCUSSION

User community and usability. Moltemplate is, by design, not a black box, so users (or 3rd-party programs) can customize the (human-readable) LT files with a text editor. The many worked examples distributed with moltemplate get users started quickly, and the simple LT syntax allows new users to add specific interactions between molecules and other details to customize subsequent LAMMPS simulations for their particular application.

The success of this approach is exemplified by the current large, diverse moltemplate user community. Currently, moltemplate software tools have been downloaded >70000 times, from >12000 unique IP addresses from the moltemplate.org web site (this excludes downloads using git and pip). Moltemplate also has a growing user community in materials science and biology. For example, moltemplate has been used to model the assembly of potential drug-delivery vehicles made from ssDNA polymers fused to dialkyl tails [32] and has been used to explore the structural transitions of genomic RNA in HIV virions, which includes defined secondary structure in the RNA chain and interaction with crosslinking proteins [33]. Living systems are also not in thermal equilibrium, posing a challenge for simulation: often, methods must be devised to model irreversible or directional processes. For example, moltemplate has been used in non-

equilibrium simulations to explore how ring-like motors walk along DNA and remodel the conformation of eukaryotic chromosomes at the megabase scale [34].

Moltemplate is increasingly being used as a *glue* or *backend* to connect domain-specific programs to LAMMPS. For example, two commercial services use moltemplate as a backend, ATB [35] and MaterialX (https://materialx.co.uk/). We have used moltemplate as an essential step in a modeling pipeline for building detailed models of entire cells with CellPACK [36], a suite of software for the generation of structural models of entire cellular organelles and bacterial cells (Figure 3). We have created a tool (https://github.com/jewettaij/cellpack2moltemplate) that enables CellPACK to generate input files for moltemplate so that the user need not be familiar with moltemplate file format to run LAMMPS simulations based on CellPACK models. In collaboration with the Covert laboratory, we are generating models of *Mycoplasma genitalium* cells based on discrete time points from WholeCell simulations [37] which can be simulated in LAMMPS. The models contain several hundred types of molecular entities with a variety of shapes and sizes, including long fibers and membrane-spanning proteins. Models are generated using CellPACKgpu [38], with DNA, messenger RNA, and associated proteins placed using LatticeNucleoids [39].

Extensibility. Modern MD programs are written in a modular style so that users can add new features and force fields when needed. For example, we added new features to LAMMPS to implement the DNA twist motor and the directional bonded ellipsoids used in the MOLC examples. The majority of LAMMPS features have similarly been contributed by users to simulate new types of representations, and LAMMPS currently includes hundreds of user-defined force-fields and features. This poses a challenge, since there is no universal file format for characterizing how these particles should move in a coarse-grained simulation. Future coarse-grained model builders are likely to create their own interactions, and it is not possible to

anticipate what their file syntax will be. The agnostic design of moltemplate fills this need, generating the text files needed for diverse current and future methods. Moltemplate currently prepares files that work with the vast number of force field styles that LAMMPS currently supports as well as diverse exotic molecular models. More importantly, moltemplate will likely be able to generate the kinds of files that users invent in the future.

Reproducibility. It has been estimated that 75%-90% of biomedical research is not reproducible [40]. However, simulation science is comparatively verifiable and reproducible. The state of the field of coarse-grained modeling is still quite experimental. In the most popular and well-studied applications, such as the MARTINI model, this ideal of reproducibility is achieved, however, for new methods, developers often are required to build code from scratch to explore hypotheses, and the resultant code is often domain-specific and not written for general use or extensibility. The LT file format used by moltemplate provides a way for scientists to design and share a general class of particle-based coarse-grained models with their peers for verification. Moltemplate includes approximately 50 examples that can be used as templates and modified to create new coarse-grained models. In this way, moltemplate accelerates the pace that scientists can innovate by taking someone's model, modifying it, and creating a new model to address a different question, while at the same time, reproducibly documenting the new methods.

Future Directions. LAMMPS currently has the ability to create and destroy bonds during a simulation and modify particle types [24], although the file format is complex and difficult to master. We are currently developing a user-friendly method to control these simulations using only a few lines of moltemplate code (https://github.com/jewettaij/lammps_mca_examples). This will make it possible to simulate “active matter” processes in the cell such as replication, transcription and translation, enzymatic reactions, cell division machinery, cytoskeletal

dynamics, trafficking, cell signaling, and many others. A Python API is also being developed allowing future moltemplate users to build molecules using Python syntax which is independent of LAMMPS whenever LAMMPS-specific features are not being used.

Code availability

Moltemplate (available at http://moltemplate.org) is free open-source software. It is distributed under the MIT and PSF licenses. Moltemplate uses modern tools for public collaborative development and has many contributors. Users contribute suggestions, bug-reports, code, force-fields, examples, and documentation using GitHub. TravisCI is used for continuous-integration.

Acknowledgements

Moltemplate was supported by NIH grants T32-AI007354-29, GM120604 and GM122588, NSF-MCB-1158577 and NSF-MCB-1716956, and HFSP RGP0014/2014. We thank Valeria Molinero, Yandong Zhang, Nathaniel Charest, William M. Clemons Jr., David Keffer, Marcus Martin, Paul Saxe and Robert Compton for support and useful discussions.

Author Contributions

AJ conceived, developed the original software, and carried out simulations. DS, JL, SMS, MAB, and OMR, contributed code improvements. OMR and MR contributed the MOLC example. SMB contributed the MARTINI example. MM, LA, and DSG created the mycoplasma models. AJ and DSG drafted the manuscript. DSG, GJJ, JES, TK, and RTD, planned and supervised the work.

Competing Interests

The authors declare no competing interests.

References

1. Senn HM, Thiel W (2009) QM/MM methods for biomolecular systems. *Angew Chem Int Ed* 48: 1198–1229.
2. Karplus M, McCammon JA (2002) Molecular dynamics simulations of biomolecules. *Nat Struct Biol* 9: 646–652.
3. Lee EH, Hsin J, Sotomayor M, et al. (2009) Discovery through the computational microscope. *Structure* 17: 1295–1306.
4. Dror RO, Dirks RM, Grossman JP, et al. (2012) Biomolecular simulation: A computational microscope for molecular biology. *Annu Rev Biophys* 41: 429–452.
5. Marrink SJ, Risselada HJ, Yefimov S, et al. (2007) The MARTINI force field: Coarse grained model for biomolecular simulations. *J Phys Chem B* 111: 7812–7824.
6. Saunders MG, Voth GA (2013) Coarse-graining methods for computational biology. *Annu Rev Biophys* 42: 73–93.
7. Hafner AE, Krausser J, Šarić A (2019) Minimal coarse-grained models for molecular self-organisation in biology. *Curr Opin Struct Biol* 58: 43–52.
8. Pak AJ, Voth GA (2018) Advances in coarse-grained modeling of macromolecular complexes. *Curr Opin Struct Biol* 52: 119–126.
9. Ingólfsson HI, Lopez CA, Uusitalo JJ, et al. (2014) The power of coarse graining in biomolecular simulations: The power of coarse graining in biomolecular simulations. *Wiley Interdiscip Rev Comput Mol Sci* 4: 225–248.
10. Dama JF, Sinitskiy AV, McCullagh M, et al. (2013) The Theory of Ultra-Coarse-Graining. 1. General Principles. *J Chem Theory Comput* 9: 2466–2480.
11. Katkar HH, Davtyan A, Durumeric AEP, et al. (2018) Insights into the Cooperative Nature of ATP Hydrolysis in Actin Filaments. *Biophys J* 115: 1589–1602.
12. Rühle V, Junghans C, Lukyanov A, et al. (2009) Versatile Object-Oriented Toolkit for Coarse-Graining Applications. *J Chem Theory Comput* 5: 3211–3223.

13. Nguyen LT, Gumbart JC, Beeby M, et al. (2015) Coarse-grained simulations of bacterial cell wall growth reveal that local coordination alone can be sufficient to maintain rod shape. *Proc Natl Acad Sci* 112: E3689–E3698.
14. Wang J, Wang W, Kollman PA, et al. (2006) Automatic atom type and bond type perception in molecular mechanical calculations. *J Mol Graph Model* 25: 247–260.
15. Jo S, Kim T, Iyer VG, et al. (2008) CHARMM-GUI: A web-based graphical user interface for CHARMM. *J Comput Chem* 29: 1859–1865.
16. Hollingsworth SA, Dror RO (2018) Molecular dynamics simulation for all. *Neuron* 99: 1129–1143.
17. Graham JA, Essex JW, Khalid S (2017) PyCGTOOL: Automated generation of coarse-grained molecular dynamics models from atomistic trajectories. *J Chem Inf Model* 57: 650–656.
18. Damre M, Marchetto A, Giorgetti A (2019) MERMAID: dedicated web server to prepare and run coarse-grained membrane protein dynamics. *Nucleic Acids Res* 47: W456–W461.
19. Qi Y, Ingólfsson HI, Cheng X, et al. (2015) CHARMM-GUI Martini Maker for coarse-grained simulations with the Martini force field. *J Chem Theory Comput* 11: 4486–4494.
20. Klein C, Sallai J, Jones TJ, et al. (2016) A hierarchical, component based approach to screening properties of soft matter, In: Snurr RQ, Adjiman CS, Kofke DA (Eds.), *Foundations of Molecular Modeling and Simulation*, Singapore, Springer Singapore, 79–92.
21. Eastman P, Swails J, Chodera JD, et al. (2017) OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Comput Biol* 13: e1005659.
22. Plimpton S (1995) Fast parallel algorithms for short-range molecular dynamics. *J Comput Phys* 117: 1–19.
23. Molinero V, Moore EB (2009) Water modeled as an intermediate element between carbon and silicon. *J Phys Chem B* 113: 4008–4016.

24. Gissinger JR, Jensen BD, Wise KE (2017) Modeling chemical reactions in classical molecular dynamics simulations. *Polymer* 128: 211–217.
25. Snodin BEK, Randisi F, Mosayebi M, et al. (2015) Introducing improved structural properties and salt dependence into a coarse-grained model of DNA. *J Chem Phys* 142: 234901.
26. Henrich O, Gutiérrez Fosado YA, Curk T, et al. (2018) Coarse-grained simulation of DNA using LAMMPS: An implementation of the oxDNA model and its applications. *Eur Phys J E* 41: 57.
27. Racko D, Benedetti F, Dorier J, et al. (2015) Generation of supercoils in nicked and gapped DNA drives DNA unknotting and postreplicative decatenation. *Nucleic Acids Res* 43: 7229–7236.
28. Krajina BA, Spakowitz AJ (2016) Large-scale conformational transitions in supercoiled DNA revealed by coarse-grained simulation. *Biophys J* 111: 1339–1349.
29. Wang X, Llopis PM, Rudner DZ (2013) Organization and segregation of bacterial chromosomes. *Nat Rev Genet* 14: 191–203.
30. Hong S-H, Toro E, Mortensen KI, et al. (2013) Caulobacter chromosome in vivo configuration matches model predictions for a supercoiled polymer in a cell-like confinement. *Proc Natl Acad Sci* 110: 1674–1679.
31. Ricci M, Roscioni OM, Querciagrossa L, et al. (2019) MOLC. A reversible coarse grained approach using anisotropic beads for the modelling of organic functional materials. *Phys Chem Chem Phys* 21: 26195–26211.
32. Kuang H, Gartner III TE, Dorneles de Mello M, et al. (2019) ssDNA-amphiphile architecture used to control dimensions of DNA nanotubes. *Nanoscale* 11: 19850–19861.
33. Goodsell DS, Jewett A, Olson AJ, et al. (2019) Integrative modeling of the HIV-1 ribonucleoprotein complex. *PLOS Comput Biol* 15: e1007150.

34. Sanborn AL, Rao SSP, Huang S-C, et al. (2015) Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes. *Proc Natl Acad Sci* 112: E6456–E6465.
35. Malde AK, Zuo L, Breeze M, et al. (2011) An Automated Force Field Topology Builder (ATB) and Repository: Version 1.0. *J Chem Theory Comput* 7: 4026–4037.
36. Johnson GT, Autin L, Al-Alusi M, et al. (2015) cellPACK: a virtual mesoscope to model and visualize structural systems biology. *Nat Methods* 12: 85–91.
37. Karr JR, Sanghvi JC, Macklin DN, et al. (2012) A whole-cell computational model predicts phenotype from genotype. *Cell* 150: 389–401.
38. Klein T, Autin L, Kozlikova B, et al. (2018) Instant Construction and Visualization of Crowded Biological Environments. *IEEE Trans Vis Comput Graph* 24: 862–872.
39. Goodsell DS, Autin L, Olson AJ (2018) Lattice models of bacterial nucleoids. *J Phys Chem B* 122: 5441–5447.
40. Niven DJ, McCormick TJ, Straus SE, et al. (2018) Reproducibility of clinical research in critical care: a scoping review. *BMC Med* 16: 26.

FIGURES

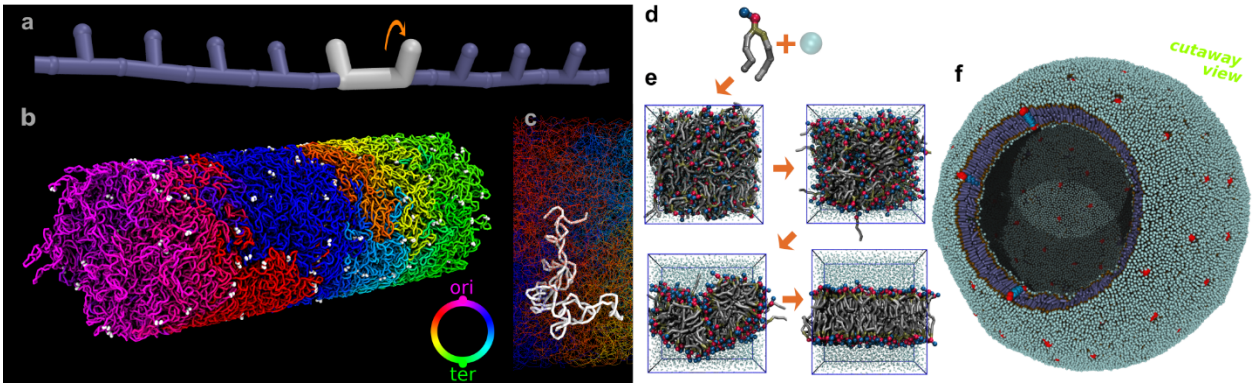


Figure 1. Coarse-grained physics-based models of chromosomes and organelles.

a) Coarse-grained DNA represented with three particles every 42 bp, including a dummy particle to represent the local superhelical state. A rotating motor (white) applies a constant torque to 4 particles. 219 lines of text files were required to implement this example including moltemplate files, tabulated potentials, minimization and run protocols. **b)** Predicted conformation of the entire genome of *Caulobacter crescentus* (4Mbp) in the absence of DNA-binding proteins, created by relaxing, twisting, and compressing a circular polymer that was originally stretched while confined in a tube of radius 320nm. Bottle-brush-like supercoils form as a result of maintaining the polymer at constant torsional tension. This example was implemented using 397 lines of text. **c)** Detail of a large, highly-branched plectonemic supercoil (10kbp). **d)-e)** Simulating the formation of a lipid bilayer using the *MARTINI* force field. This example contains 300 lipids, 6000 waters, and requires 220 lines of text. **f)** A liposome with protein inclusions containing 120 proteins, 65000 lipids, and implemented with 544 lines of text (including PACKMOL files). PACKMOL was used to randomize the molecular positions, and moltemplate was used to assemble the LAMMPS simulation files. Files for these examples can be downloaded at http://doi.org/10.5281/zenodo.4392267. Systems visualized using VMD and TopoTools.

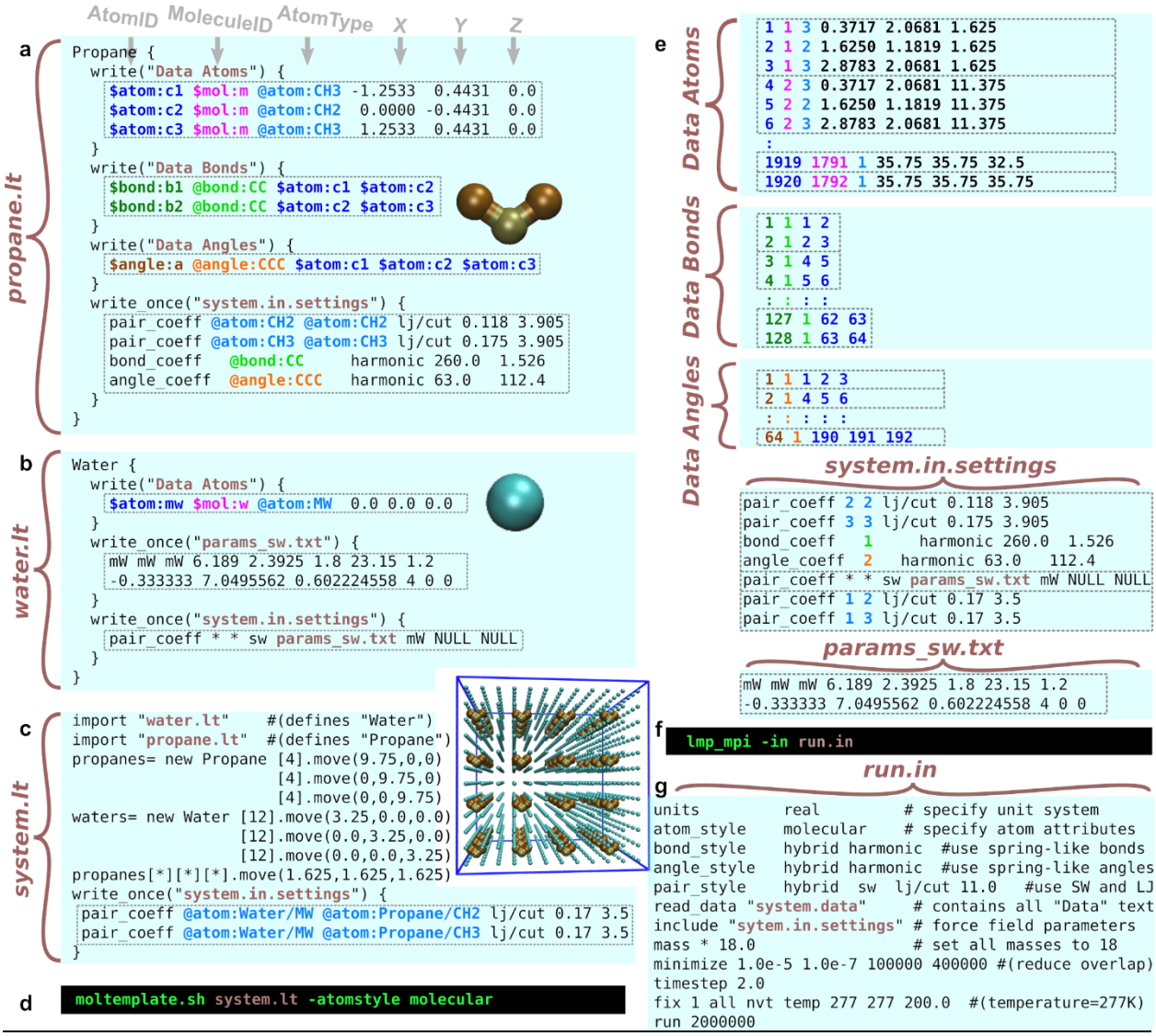


Figure 2. Files needed prepare and run a LAMMPS simulation with moltemplate

a) The “propane.lt” file contains the definition of a coarse-grained “Propane” molecule containing 3 particles, 2 bonds, and one angular spring. **b)** A similar file defining a coarse-grained “Water” molecule includes a text block (“params_sw.txt”) describing parameters for its more complicated *(many-body)* force-field. **c)** These molecule objects can be used as building blocks to create text files describing more complex systems using the “new” command (for example a water-hydrocarbon mixture containing 64 Propanes and 1728 WatMWs, *inset*). Text enclosed in each “*write(FILENAME)”* statement will be appended to the generated file (eg. “Data Bonds”) each time a copy of the molecule is created, and the counter variables ($atom:, $bond:, $angle:) will

be replaced by integers and incremented. However, type variables beginning with @ are not incremented. Interactions between water and propane are also specified. Generated files are shown in e), with rectangles enclosing the portion of text generated by each molecule copy. **d)** Command used to run moltemplate. The optional "*-atomstyle*" argument customizes particle attributes. **e)** Files generated by moltemplate. Note: Coordinates are modified by the move() commands in part c). "Data" files are concatenated together by moltemplate and renamed "system.data". **f)** Command used to run LAMMPS **g)** The "run.in" file contains links to these generated files as well as LAMMPS-specific run-time settings.

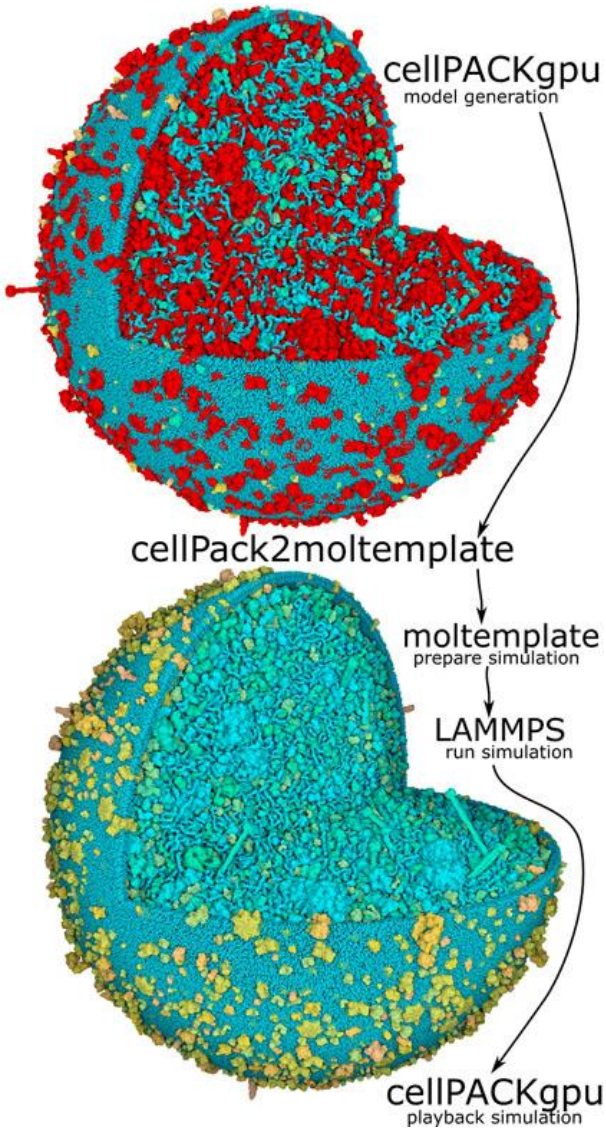


Figure 3. Integrated pipeline for building models of entire bacterial cells.

3D structures for the ~500 types of proteins in a mycoplasma proteome are curated in the online tool Mesoscope and used to create interactive draft models in cellPACKgpu. Moltemplate, with the utility program cellPack2moltemplate, then converts molecular location and orientation information into a LAMMPS input file, to perform a coarse-grained simulation that eliminates steric clashes and generates more realistic 3D models. Final models are interactively explored with cellPACKgpu. Molecules with steric clashes are shown in red in both model images.