User Guide for GB Polymer Model in LAMMPS

David Friday, Alex Cohen, Nick Jackson
friday1@illinois.edu, aecohen@mit.edu, jacksonn@illinois.edu

November 1, 2022

1 Introduction

Welcome to the user guide for the GB polymer model in LAMMPS. This polymer model can be used to model and simulate diverse anisotropic polymers with varying degrees of coarse-graining. The attached files include codes for generating LAMMPS data files and input scripts, submitting data files, and analyzing and visualizing the results from the simulations. The code is intended to be automated and easy-to-use. Thus, the user needs to specify only a small number parameters before running the codes. We believe that this model provides essential insights into anisotropic polymer structural and dynamical properties with enough specificity as to differentiate between different polymer chemistries. We hope this model proves useful in your research!

2 How to Anisotropic CG Code with LAMMPS

Before generating the LAMMPS executable, we need to edit the LAMMPS source code to contain the functionality for the dihedrals. Copy and paste the bond_eldihedral.cpp and bond_eldihedral.h files into the **ASPHERE** directory (which is within the **src** directory) in the LAMMPS source code. **Note: the dihedral vector is associate with the second diameter of the ellipsoids.** In order to generate a LAMMPS executable with the appropriate packages installed, you will need to run the cmake command with the following specifications:

cmake -D PKG_REPLICA=on -D PKG_USER-MISC=on -D PKG_ASPHERE=on -D PKG_MOLECULE=on -D PKG_BODY=on -D PKG_RIGID=on -D BUILD_MPI=yes -D BUILD_OMP=yes

3 How to Prepare Data files

The files provided allow you to generate LAMMPS data files with a minimal number of user input parameters. These parameters include the structure of a single monomer, the degree of polymerization, the number of polymers in the

simulation box, the polymer spacing, and the box dimensions. While reading the following explanations, it may be helpful to consult an example .mon file with a schematic of its corresponding polymer in Figures 1 - 2.

The first step for preparing a data file is to create a .mon file that includes the structure of a single monomer. The .mon file has four sections: Particles, Angles, Dihedrals/Bonds, and Rigid. Each section must begin with a single line that contains the name of the section followed by a colon, with no spaces. Specifically, the particles section must begin with a line containing only "Particles:", the angles section must begin with a line containing only "Angles:", the dihedrals/bonds section must begin with a line containing only "Dihedrals/Bonds:", and the rigid section must begin with a line containing only "Rigid:". The sections must also be specified in the order described above. The structure of information required in each section is outlined below. We have also provided a sample .mon file for a coarse-grained polymer and an image of the polymer with each particle numbered and certain parameters visually explained. It may be helpful to follow along with this image and the .mon file while trying to understand the information below.

3.1 Particles

The particles section contains information about the spatial arrangement and intrinsic features of the coarse-grained particles in the monomer. Each line in the particles section must correspond to the following format, where each component is separated by a single space:

Particle Number, Particle Type, X Position, Y Position, Z Position, Charge, Ellipsoid Flag, Density, X Diameter, Y Diameter, Z Diameter

The X,Y,Z positions are relative to an arbitrary starting position. The ellipsoid flag variable specifies whether the particle is an ellipsoid (in which can it should contain "1") or a point particle (in which case it should contain "0"). If the particle is an ellipsoid, the Density value is a true density. If the particle is a point particle, the Density value corresponds to the mass of the point particle. In addition, point particles must contain zeros for the three diameter values. Usually, all particles are ellipsoids except for the bond sites, which are point particles with very small masses relative to the masses of the ellipsoids.

3.2 Angles

The angles section contains information about the angle potentials that are specified in every monomer. Each line in the angles section must correspond to the following format, where each component is separated by a single space:

Angle Number, Angle Type, Particle Number of first particle in angle, Particle Number of middle particle in angle, Particle Number of third particle in angle

We also note that in many cases, there will be angles defined across monomers. For example, a simple coarse-grained anisotropic polymer consisting of disks connected by harmonic bonds will have angles defined between the disks down

```
Particles:
1 2 0.0 0.0 0.0 0 0 0.00001 0 0 0
2 1 0.5 0.0 0.0 0 1 6.667 1.0 0.4 0.72
3 2 1.0 0.0 0.0 0 0 0.00001 0 0 0
4 2 1.2 0.0 0.0 0 0 0.00001 0 0 0
5 1 1.7 0.0 0.0 0 1 6.667 1.0 0.4 0.72
6 2 2.2 0.0 0.0 0 0 0.00001 0 0 0
Angles:
1 1 2 3 4
2 1 3 4 5
-1 1 5 6 7
-1 1 6 7 8
Dihedrals/Bonds:
1 1 1 2
2 1 2 3
3 2 3 4
4 3 2 5
5 1 4 5
6 1 5 6
-1 2 6 7
-1 3 5 8
Rigid:
1 1
2 1
3 1
4 2
5 2
6 2
```

Figure 1: Example monomer file containing all necessary sections and formatting.

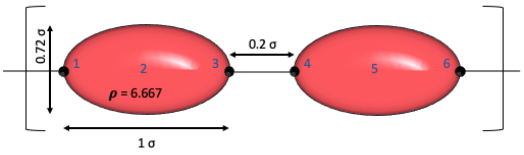

Figure 2: Schematic of monomer corresponding to the monomer file and input script included in this package.

the backbone of the polymer chain. In order to specify one of these angles that crosses the monomer boundary, a "-1" must be placed where the Angle Number usually resides. In addition, the particles that are included in the angle that are in the "next monomer" (and are therefore not specified in the particles section) must be given the number that the will correspond to in the actual polymer (this number will equal the particle number of the corresponding particle in the specified monomer plus the total number of specified particles in the .mon file).

3.3 Dihedrals/Bonds

The dihedrals/bonds section contains information about the dihedral and bond potentials that are specified in every monomer. Dihedrals and bonds are lumped together because the dihedrals between two ellipsoids have been implemented using the bonds feature in LAMMPS. Each line in the dihedrals/bonds section must correspond to the following format, where each component is separated by a single space:

Dihedral/Bond Number, Dihedral/Bond Type, Particle Number of first particle in the dihedral/bond, Particle Number of second particle in the dihedral/bond

Similar to the angles, there will be dihedrals and bonds that exist across monomers in many cases. For example, every polymer will have a bond defined between two monomers. These dihedrals and bonds that span across two monomers must contain a "-1" in the Dihedral/Bond Number spot. The particle numbers are then specified in the same way as the particle numbers in the angles section are specified.

3.4 Rigid

The rigid section contains information about the rigid bodies that are specified in every monomer. Each line in the rigid section must correspond to the following format, where each component is separated by a single space:

Particle Number, Rigid Body Number

If the particle is not a part of any rigid body, 0 can be placed in the spot for Rigid Body Number (although, in reality, any number should work).

Note: For simple polymers that contain no internal angles and dihedrals, you may need to include information about two monomers in your .mon file. The data the user then provides in the script to produce the actual LAMMPS data file, which is outlined below, must be adjusted accordingly.

3.5 Data file production script

The data file production script, named datafile_production.py, contains the code to generate the actual LAMMPS data file from the monomer file. There are 9 variables located at the top of the datafile_production.py that the user must specify. These variables are:

1. NUM_MOLECULES = total number of polymers in the simulation

2. NUM_MONOMERS = total number of monomers per polymer (degree of polymerization)
3. monomerOffset = distance between a particle in one monomer to the corresponding particle in the next monomer
4. moleculeOffset = distance to separate different polymers by
5. mon_file_name = name of monomer information file
6. data_file_name = name of the output data file
7. box_x = length of box in x direction
8. box_y = length of box in y direction
9. box_z = length of box in z direction

In order to create the data file, the monomer information file and datafile_production.py must be located in the same directory. The resulting data file will also be placed in this same directory.

4 How to Prepare input files

Input files define the interactions between the different types of particles as well as the simulation details. I will not go into the details of how to set it up, as extensive instructions can be found on the LAMMPS website. I have included a sample input file along with this user guide.

5 How to Analyze Output

Analysis will depend on what you are specifically interested in studying. In my case, I was interested in simple polymer structural properties, such as radius of gyration, order parameters along the stacking and bond axis, and RDFs. The analysis scripts that you will write to calculate properties of interest will depend on the way you have formatted the LAMMPS output files.

For reference, I have included six python scripts that accomplish some of the basic analysis listed above. The analysis scripts require the data to be formatted according to the LAMMPS output files in the submit_generator.py script. The six python scripts are described below:

1. replica_rg_compilation.py. Generates a file containing the values of the radius of gyration at each time step for each temperature.
2. replica_traj_compilation.py. Compiles the trajectory data into new files where each file corresponds to a specific temperature. (This is needed because LAMMPS will output the data such that each file corresponds to a processor, but the swaps change the temperature assigned to each processor – see LAMMPS website for more information.)

3. replica_op_compilation_bond.py. Creates a file for each temperature which contains the order parameter along the bond axis at each time step.

4. replica_op_compilation_stacking.py. Creates a file for each temperature which contains the order parameter along the stacking axis at each time step.

5. replica_op_onefile_bond.py. Compiles all of the files created by replica_op_compilation_bond.py into one file.

6. replica_op_onefile_stacking.py. Compiles all of the files created by replica_op_compilation_stacking.py into one file.

6 How to Visualize Output in OVITO

First, create a new .mon file which is the exact same as before except now only include the bonds that you want to visualize (and renumber the bonds). Next, run the datafile_viz.py script, using the same parameters as used in the datafile_production.py script. Then, run the conjugated_visualization.py script, where the ell variable contains the file name for the trajectory file which contains the real particles and the all_atoms variable contains the file name for the trajectory file which contains all particles. Load the data file (output of datafile_viz.py) into ovito. Then click **Add modification...** in the top right and click **Load trajectory** under the **Modification heading**. Then, load the trajectory file (output of conjugated_visualization.py) in the **Trajectory Source: External file** section on the right. To make the ellipsoids have the correct orientation, click **Edit column mapping...** in the bottom right. Set the **Particle property** of c_op[1], c_op[2], c_op[3], and c_op[4] to **Orientation**. Then, set the **Component** of c_op[1], c_op[2], c_op[3], and c_op[4] to **X**, **Y**, **Z**, **W**, respectively (Note: this may change if you change the LAMMPS output).

7 My Workflow

To give you a sense of how I ran all of the simulations to generate all of the data, I will briefly discuss my workflow. First, I would draw out a diagram of what I wanted the monomer to look like, similar to Figure 2. Next, I would translate this image into a .mon file similar to Figure 1. I would then run the datafile_production.py script to generate the LAMMPS data file. I would then create a LAMMPS input script for running a simulation of a single polymer chain at a single temperature (see in.initialize). This input script would be used to run the chain at a high temperature, generating restart files that I can use as starting trajectories for my replica exchange simulations. Once this simulation is complete, I would modify the submit_generator.py script to contain the correct parameters for my replica exchange runs. I would then run submit_generator.py to collect data.

8 LAMMPS Notes

Do not include a minimize command in the LAMMPS input script. It is incompatible with the rigid command, and the ghost particles will move. Instead, use a large langevin damping parameter for a short period of time to let the system slowly equilibrate.