Manual for running the BD Codes

January 18, 2026

Instructions for Running the Codes Stored in the Folder: `patchy_particle_model`

Patchy particle models are coarse-grained representations of complex molecules in which interactions are directionally selective rather than isotropic. Each particle consists of a central core decorated with a finite number of attractive interaction sites (*patches*) located at specific positions on its surface. These patches introduce anisotropy into the interaction potential, allowing particles to preferentially bond in certain orientations.

Such models are widely used to mimic systems where directional bonding plays a crucial role, including self-assembling colloids, protein-like particles, hydrogen-bonded fluids, and supramolecular polymers. By controlling the number, arrangement, and strength of patches, one can tune the resulting structures, ranging from linear chains and branched networks to crystalline or gel-like phases.

In this tutorial, the goal is to perform molecular dynamics simulations of a patchy particle system using a constant-temperature thermostat (NVT ensemble). By varying the temperature, students will study how thermal fluctuations compete with directional attractions, leading to distinct phase behaviors. In particular, the simulations aim to identify temperature regimes where particles form transient or persistent chains, exhibit liquid-like aggregation, or remain in a dispersed, gas-like state.

Through this exercise, students will gain insight into how anisotropic interactions govern self-assembly and phase behavior, and how simplified coarse-grained models can capture essential physical mechanisms underlying complex soft-matter systems.

Molecular Template Construction

1. Create a molecular template file (`.mol`) defining a single dumbbell-shaped molecule. The molecule should have:
   - 1 central core particle (type 1),
   - 2 patch particle (type 2) located on the poles of the particle,
   - a fixed bond length of 0.5 between the core and patch.
2. The file should clearly specify:
   - The total number of atoms,
   - Cartesian coordinates of all atoms,
   - Atom types.

3. Paste the complete contents of the `.mol` file in your answer sheet for verification. For example, a template may look like:

```
5 atoms

Coords
1  1.0    1.0    1.0
2  1.2886 1.2886 1.2886
3  0.7113 0.7113 1.2886
4  0.7113 1.2886 0.7113
5  1.2886 0.7113 0.7113

Types
1 1
2 2
3 2
4 2
5 2
```

This template ensures that the bond distance is fixed, and the patch particle is appropriately positioned relative to the core.

Simulation Setup

1. **Defining essential variables and simulation units**

```
units lj
atom_style  hybrid sphere molecular
boundary p p p
pair_style hybrid lj/cut 2.0 cosine/squared 0.12
```

The simulation uses reduced Lennard-Jones units. Periodic boundary conditions are applied in all directions to model a bulk system. The hybrid pair style allows core-core interactions via LJ and patch-patch interactions via cosine-squared potential.

2. **Creating the simulation box**

```
region box block 0 30.0 0 30.0 0 30.0
create_box 2 box
```

A cubic box of dimensions $30 \times 30 \times 30$ is created. Both core (type 1) and patch (type 2) particles will occupy this volume.

3. **Loading the molecular template and creating molecules**

```
molecule patchy_part patchy_molecule.mol
create_atoms 0 random 1000 87910 NULL mol
 ...patchy_part 454756 overlap 1.5 maxtry 50
```

The molecular template defines the dumbbell-shaped particle with a fixed bond length of 0.5. Molecules are randomly placed while minimizing overlaps via the ‘overlap‘ and ‘maxtry‘ parameters.

4. **Defining particle properties and interactions**

```
pair_coeff 1 1 lj/cut 0.01 1.3 2.0
pair_coeff 1 2 none
pair_coeff 2 2 cosine/squared 8 0.3 0.35

set type 1 mass 1.0
set type 2 mass 0.000001

set type 1 diameter 1.0
set type 2 diameter 0.0
```

Core-core particles interact via Lennard-Jones potential; patch-patch interactions are modeled using a Gaussian-like cosine-squared potential. Patch particles are nearly massless and small to enforce directional bonding.

5. **Grouping particles and neighbor exclusions**

```
group core type 1
group patch type 2
group rigid_molecule type 1 2

neigh_modify exclude molecule/intra rigid_molecule
 ...every 1 delay 0 check no
```

Particles are grouped for targeted fixes. Intramolecular neighbor interactions are excluded to avoid redundant calculations within the rigid molecules.

6. **Applying thermostat and rigid-body integration**

```
fix thermo_stat core langevin 1.0 1.0 0.1 428984 omega yes
fix rigid_thermo rigid_molecule rigid/small molecule
```

A Langevin thermostat is applied to the cores to maintain temperature. Rigid-body integration keeps the core-patch bond fixed during dynamics.

7. **Dumping trajectories for visualization**

```
dump 1 all custom 10 simulation_data.lammpstrj id type x y z mol
```

Particle positions are recorded every 10 steps for post-processing and visualization (e.g., in OVITO).

8. **Computing kinetic energy and temperature of core particles**

```
compute kinetic_core core ke
fix kinetic_output core ave/time 100 1 100 c_kinetic_core
 ...file kinetic.dat mode scalar

compute temp_core core temp/sphere
fix temp_output core ave/time 100 1 100 c_temp_core
 ...file temperature.dat mode scalar
```

These computes track thermodynamic properties of core particles, allowing monitoring of equilibration and system stability.

9. **Time step and thermodynamic output settings**

```
timestep 0.005
thermo 100
thermo_style custom step temp ke pe press c_kinetic_core c_temp_core
```

A small timestep ensures numerical stability for rigid molecules. Thermodynamic data are printed every 100 steps.

10. **Running the simulation and analyzing phase behavior**

```
run 100000
```

After the simulation:

- Visualize trajectories using OVITO.
- Color core and patch particles distinctly.
- Identify at least one temperature where the system exhibits liquid-like, chain-forming behavior.
- Capture high-quality screenshots and record the corresponding temperature and pressure values.
