lammps_helper Documentation

Release 0.1

Kevin Whitham

Sep 04, 2020

Contents:

CHAPTER 1

Running LAMMPS with lammps_helper

lammps_helper provides two things to help run LAMMPS simulations from python. One to help create LAMMPS input files and one to run them. I suggest you use an IPython environment such as Jupyter Lab.

1.1 Creating LAMMPS input files:

At the beginning of your Jupyter Lab notebook, use this code:

```
import lammps_helper as lh

ip = get_ipython()
ip.register_magics(lh.lammps_magics)
```

Then you can write LAMMPS files in your notebook and export them rather than writing them separately. Using the *%%writetemplate* cell magic, any python variables in your LAMMPS code will be replaced by their values during export. For example:

```
temperature = 300
output_base_name = f'my_lammps_output_{temperature}K'
simulation_steps = 20000
list_element_names = 'Pb I N C H'
```

```
%%writetemplate in.my_lammps_input_file

# This is a LAMMPS input file

units       real
dimension   3
boundary    p p p
atom_style  full

pair_style      lj/cut 12.500
```

(continues on next page)

(continued from previous page)

```
bond_style harmonic
angle_style harmonic
dihedral_style charmm

read_data data.my_lammps_data_file

thermo_style custom step temp etotal spcpu cpuremain

# equillibrate at temperature
fix fix_npt_equilibrate all npt temp {temperature} {temperature} 100 iso 1.0 1.0 500
dump dump_trajectory_equilibrate all dcd 100 {output_base_name}_equilibration.dcd
dump_modify dump_trajectory_equilibrate unwrap yes
run {simulation_steps}

# dump the system to check geometry
write_dump all xyz {output_base_name}_after_equilibrate.xyz modify element {list_
↪element_names}

unfix fix_npt_equilibrate
undump dump_trajectory_equilibrate

# write restart file after equilibration
write_restart {output_base_name}_after_equilibrate.restart
```

Will create a file named *in.my_lammps_input_file* that will run a simulation at 300 K for 20000 timesteps and the data files created by LAMMPS will share a common naming scheme that includes the temperature.

1.2 Running LAMMPS

To run your input file in LAMMPS, do this

```
lh.run_lammps('in.my_lammps_input_file', f'log_{output_base_name}_equilibrate.lammps')
```

This simply issues the lmp_serial command. You may add variables here as well. See `run_lammps()` for more info. You will get output like this

```
Running calculation...
Writing to log_my_lammps_output_300K_equilibrate.lammps.
Calculation started at 08-06-20 16:10:45
Calculation complete at 08-06-20 16:10:45 in 0:00:00.076904.
```

CHAPTER 2

Dipoles

2.1 Creating dipole data

To get dipole moment information from LAMMPS, add something like this to your input file:

```
# create a group of atom types in the molecules you care about
group group_organic type < 5

# create a compute to assign a chunk ID for each molecule
compute molecule_chunks group_organic chunk/atom molecule

# create a compute for the center of mass location of each molecule
compute compute_com group_organic com/chunk molecule_chunks

# create a compute for the dipole of each methylammonium molecule
compute compute_dipole group_organic dipole/chunk molecule_chunks

fix fix_dipole   all    ave/time 30      1        30     c_compute_dipole[*] file␣
↪dipoles_{temperature}K.out mode vector ave one title1 "Methylammonium Dipoles
↪{temperature} K"
fix fix_com      all    ave/time 30      1        30     c_compute_com[*]    file␣
↪molecule_location_{temperature}K.out mode vector ave one title1 "Methylammonium␣
↪Center of Mass {temperature} K"
```

2.2 Reading dipole data

After the simulation, use `get_dipole_data()` to read the data from LAMMPS:

```
dipole_file = f'dipoles_{temperature}K.out'
location_file = f'molecule_location_{temperature}K.out'
```

(continues on next page)

(continued from previous page)

```
dipole_data = np.empty(shape=(0,0))

dipole_data, data_rows = get_dipole_data(dipole_data, dipole_file, location_file,␣
↪temperature)
```

If you allocate the array *dipole_data* before calling `get_dipole_data()` it may run faster, for example:

```
import numpy as np
data_rows = int(np.ceil(simulation_timesteps / simulation_sampling_interval)) * num_
↪molecules
dipole_data = np.empty((data_rows, 12))
```

2.3 Visualizing dipole data

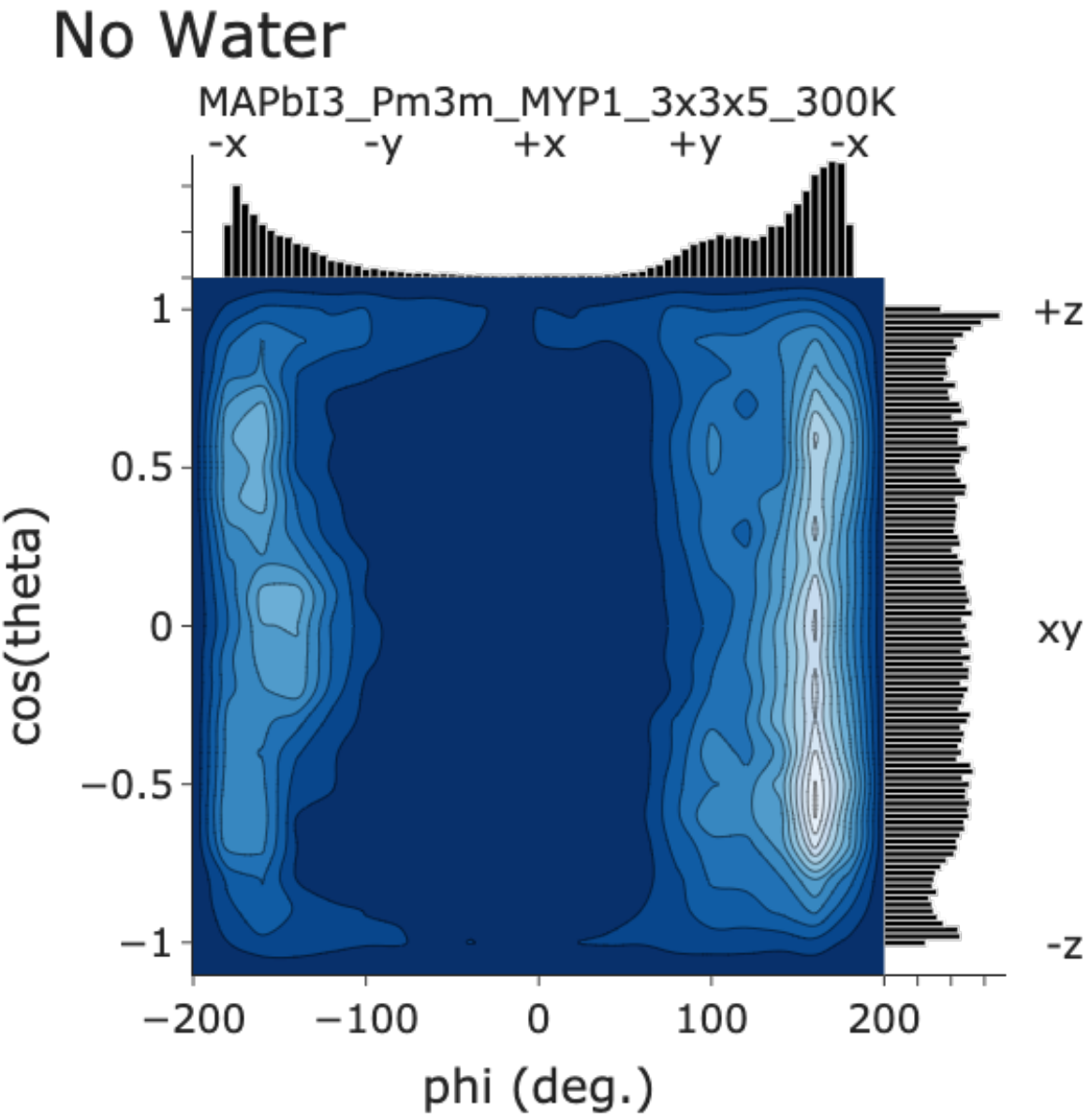
lammps_helper provides a few functions to help visualize dipole orientations:

- `make_dipole_contour_plot()`
- `plot_mean_dipole_orientation()`
- `plot_mean_dipole_angles()`

The first gives a 2D histogram, the second gives a 3D vector plot of the average molecule orientations, the third gives a volume plot of average dipole angles cos(theta) and phi.

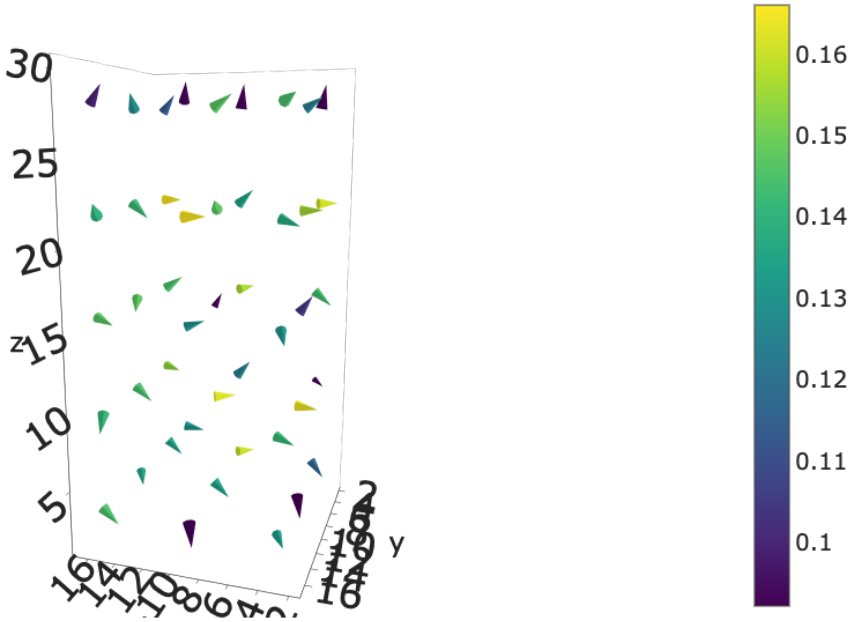
To view a histogram of dipole orientations over the course of the simulation:

```
fig = lh.make_dipole_contour_plot(dipole_data, title = 'No Water', subtitle = output_
↪base_name)
fig.show()
```



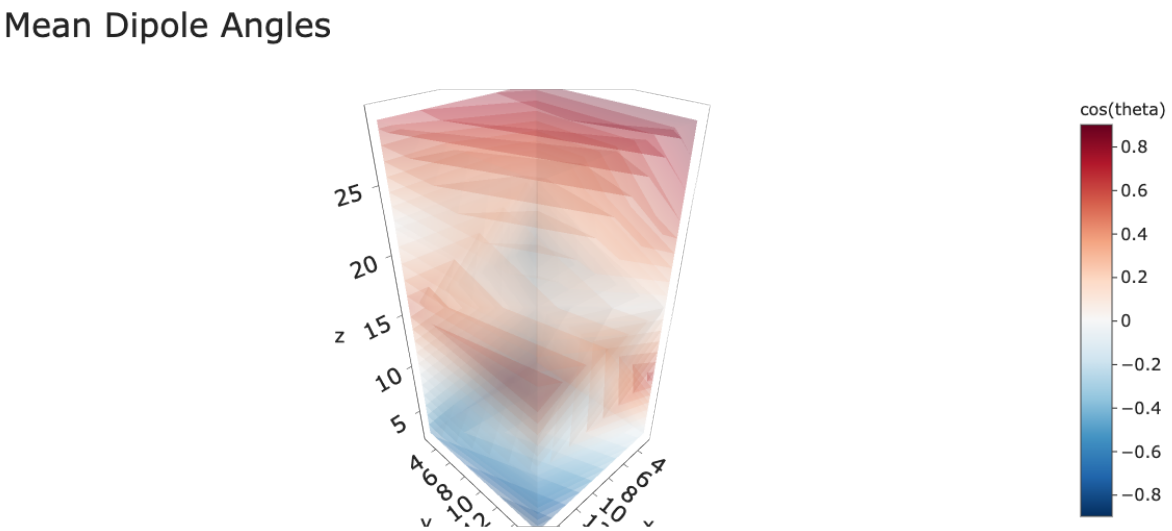
To view a 3D plot of the average orientation of each molecule:

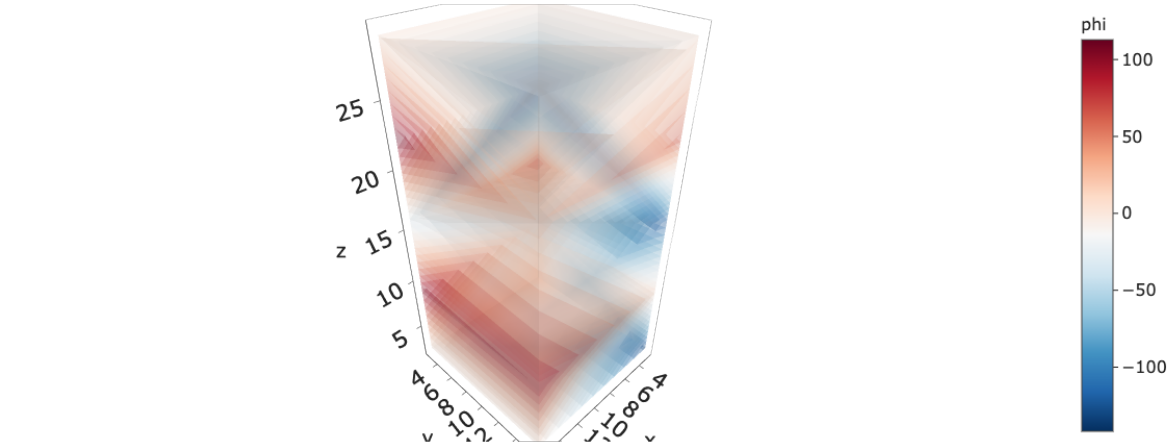
```
fig = plot_mean_dipole_orientation(dipole_data)
fig.show()
```



To view a 3D plot of the dipole angles:

```
fig_cos, fig_phi = lh.plot_mean_dipole_angles(dipole_data)
fig_cos.show()
fig_phi.show()
```




Mean Dipole Angles
25
20
z 15
10
5
phi
100
50
0
−50
−100

CHAPTER 3

Topology

3.1 Importing structure

To create bond topology you first need a LAMMPS data file with the *Masses* and *Atoms* sections. This can be generated from any structure file (cif, xyz, pdb, vesta, etc.) by the command-line tool *atomsk*:

```
atomsk structure.cif lmp
```

It can also be generated other ways such as *topotools* or *pymatgen*:

```
import pymatgen.io.lammps.data as pgld
from pymatgen import Structure

pg_structure = Structure.from_file('structure.cif')
ld_object = pgld.LammpsData.from_structure(pg_structure)
ld_object.write_file('structure.lmp')
```

The *Masses* section must have a comment after each mass with the name of the element. Element names are added automatically by *atomsk* but not by *pymatgen*. For example:

```
Masses

1   207.200000000 # Pb
2    14.006700000 # N
3    12.010700000 # C
4   126.904470000 # I
```

The *Atoms* section lists all atom coordinates. It can optionally have a column with the atomic charges.

3.2 Creating bond topology

First specify pairs of atoms to bond and the maximum distance for bonding. The following will create bonds between nitrogen and carbon atoms up to 1.5 Angstroms and between carbon atoms up to 1.52 Angstroms:

```
import pandas as pd

bond_pairs = pd.DataFrame(data = dict(element1 = ['N',  'C'],
                                      element2 = ['C',  'C'],
                                      cutoff   = [1.5, 1.52]))
```

To add bond, angle and dihedral information to the LAMMPS file:

```
import lammps_helper as lh
bonds =  lh.add_bond_data('lammps_data.lmp', bond_pairs)
angles, dihedrals = lh.add_angle_dihedral_data('lammps_data.lmp', bonds)
```

Keep in mind that the topology must be very simple. This code probably cannot handle cyclic molecules. If your structure has molecules with rings, I suggest you use moltemplate to generate the structure or lammps-interface to convert the structure from a .cif file.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search