Coarse-grained molecular modelling using molecular dynamics (MD)

February 6, 2024

Luke K. Davis[1,2,*]

[1] Department of Mathematics, University College London, Gordon Street, London, WC1H 0AY, UK
[2] Institute for the Physics of Living Systems, UCL, Gower Street, London, WC1E 6BT, UK
[*] correspondence: luke.davis@ucl.ac.uk

Abstract

Reading/reference material, instructions to build LAMMPS in python3, and example simulation code.

Contents

1 Reading and reference material

Molecular biology reference: [1]
Soft matter: [2]
Coarse-grained approaches in biophysical modelling: [3, 4]
Deformable particles: [5]
Feynman's "Simulating Physics with Computers" is available here
Molecular Dynamics: [6, 7]
D. C. Rapaport's "The Art of Molecular Dynamics Simulation" [8] is freely available here
LAMMPS: [9, 10].
LAMMPS Documentation: https://docs.lammps.org/Manual.html

2 Building and installing LAMMPS for python3 (Windows)

Follow the steps in [LAMMPS WSL](#) to install a linux virtual machine and to install the prerequisite packages.

You might need to update your WSL executable.

The last section (in the above link) that you should complete is the "Install prerequisite packages". DO NOT complete the "Dowload LAMMPS" section (and those that folloew). Once you have finished installing the prerequisite packages follow the instructions in section 5.2, and the following subsections, in this document.

3 Building and installing LAMMPS for python3 (MacOS- using Conda)

First create a LAMMPS conda environment:

```
1    conda config   add   channels conda-forge
2    conda create -n lammps-env
3    conda activate lammps-env
4    conda install lammps
```

Then add packages such as numpy, matplotlib, and scipy through the Anaconda navigator.

4 Building and installing LAMMPS for python3 (MacOS- just terminal)

The following instructions only apply after you have reached the 'Install LAMMPS" step in section 5.3. Before you starting the make commands do the following:

Execute "mdfind -name Python.h" (without quotation marks!) in the terminal. It will show a few options, select one line that ends in /Python.h and copy that line.

If mdfind does not initially work, follow the instructions it gives.

Next, in the /lammps/lammps-develop-python/lammps-develop/src/MAKE folder there is a file called Makefile.serial.

Open this file for editing in a text editor.

Change the following line:

CCFLAGS = -g -O3 -std=c++11

to

CCFLAGS = -g -O3 -std=c++11 -I¡path-to-Python.h¿ -lpython3.X

where ¡path-to-Python.h¿ is the line you copied from before (after having run mdfind -name Python.h) and X is the python 3 version e.g. -lpython3.10 is what I would use.

5 Building and installing LAMMPS for python3 (tested only on Linux/U-NIX)

5.1 Upgrade the apt package manager

Run the following in a terminal:

```
1    sudo apt update
2    sudo apt upgrade -y
```

Note: sudo X means run the X commands as super user. You need to know your password/super user password.

Next run the following to install relevant dependencies for LAMMPS:

```
1    sudo apt install -y cmake build-essential ccache gfortran openmpi-bin libopenmpi-
     dev libfftw3-dev libjpeg-dev libpng-dev python3-dev python3-pip python3-virtualenv
      libblas-dev liblapack-dev libhdf5-serial-dev hdf5-tools
```

5.2 Python3 (numpy, matplotlib, and scipy etc.

Check you have python3 and note down its version as something like 3.X (also useful in case bugs):

```
1    python3 -V
```

One thing I needed to do to get things working is to install the "venv" utility:

```
1    sudo apt install python3.X-venv
```

Note you should replace X in the above with the numbers after the first decimal point of the python version: 3.X.

Install numpy, matplotlib, and scipy...

```
1    python3.X -m pip install numpy matplotlib scipy
```

Now check they are installed correctly. You can do this by simply running a live python kernel as python3.X then sequentially importing the packages. If they are installed correctly you should get silent output. On my machine my terminal looked like this, indicating successful installation of the python packages:

```
1    user@ld-stone64:~/Work/Cell0027/lammps/lammps-develop-python/lammps-develop/src$
     python3.10
2    Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
3    Type "help", "copyright", "credits" or "license" for more information.
4    >>> import numpy as np
5    >>> import matplotlib
6    >>> import scipy
7    >>>
```

5.3 Download and install LAMMPS

Next:

```
1    wget https://github.com/lammps/lammps/archive/refs/heads/develop.zip
```

Then:

```
1    mkdir lammps
2    mv develop.zip lammps/
3    cd lammps/
4    unzip develop.zip -d ./lammps-develop-python/
```

Let's change to the src/ (source-code) directory:

```
1    cd lammps-develop-python/lammps-develop/src/
```

Install LAMMPS, with the usually useful packages:

```
1    machine="serial"
2    echo " == Making LAMMPS-$machine for $nodename == "
3    echo "Machine filename: $machine"
4    make clean-${machine}
5    make no-all
6    make yes-EXTRA-FIX
7    make yes-EXTRA-PAIR
8    make yes-MANIFOLD
9    make yes-MOLECULE
10   make yes-RIGID
11   make yes-MANYBODY
12   make yes-BROWNIAN
13   make yes-PYTHON
14   make -j 4 mode=shared ${machine}
15
16   make install-python
```

Now check if LAMMPS in python can be called upon in the right python environment. Like how we tested that numpy etc. was installed, we do the same for the LAMMPS package:

```
1  user@ld-stone64:~/Work/Cell0027/lammps/lammps-develop-python/lammps-develop/src$
   python3.10
2  Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
3  Type "help", "copyright", "credits" or "license" for more information.
4  >>> from lammps import lammps, PyLammps
5  >>>
```

6   Example (basic) LAMMPS simulation code

```python
1  ## This uses solely a Python (LAMMPS API) approach
2  ## system: WCA [repulsive] Brownian Particles
3  ## Author: Dr. Luke Davis
4  ## Year: 2022
5  ## E: luke.davis@ucl.ac.uk
6
7  from __future__ import print_function
8  import numpy as np
9  import sys
10 import random
11 import math
12 import ctypes
13 import os
14
15 wd = "./"
16 xyzdata_dir = f"{wd}xyzdata/"
17
18 try:
19     os.mkdir(xyzdata_dir)
20 except FileExistsError:
21     # directory already exists so ignore
22     pass
23
24 ## SIMULATION PARAMETERS
25 dimension = 2 # spatial dimension
26 bl = 80 # box length
27 hbl = 0.5*bl
28 pack_frac = 0.01 # requested packing fraction
29
30
31 T = 1.0 # reduced temperature (all units are reduced according to LAMMPS units LJ)
32 dt = 0.004 #timestep
33 Nthermo = 10 # display thermodynamic information every this number of steps
34 Nxyz = 1000 # dump xyz file every this number of steps
35 Nruns = 10000 # number of total timesteps
36 damp = 1.0 # inverse mobility
37 instance = 0 # label to distinguish different random runs
38 #active_types = [2]
39
40 # INTERACTION POTENTIAL PARAMETERS
41 epsilon = 10.0
42 sigma = 1.0
43 rad = sigma*0.5
44 HS_sigma = round(sigma*(2./( 1. + math.sqrt(T/epsilon)) )**(1./6.),3) # approximate
      HS diameter
45 HS_rad = 0.5*HS_sigma
46 cutoff = 2.**(1./6.)*sigma # this default value gives the WCA potential i.e. only
      repulsion (rc = 2^(1/6)*sigam ~ 1.122 when sigma = 1)
47 comm_cutoff = cutoff*1.2
48 vol_atom = 0.
49 if dimension == 2:
50     vol_atom = math.pi*HS_rad**2
51 Natoms = math.floor((pack_frac*(bl**dimension))/vol_atom) # number of atoms
52 dens = Natoms/(bl**dimension) # density
```

```python
53  unique_sim_label = f"RABPS_LAMMPS_N{Natoms}_bl{bl}_HSsig{HS_sigma}_T{T}_damp{damp}_dt
        {dt}_Nruns{Nruns}_instance{instance}"
54
55  # seeds for random things in LAMMPS script
56  seed0 = random.randint(10000,1000000)
57  seed1 = random.randint(10000,1000000)
58  seed2 = random.randint(10000,1000000)
59  seed3 = random.randint(10000,1000000)
60
61
62  # BEGIN LAMMPS SIMULATION
63  #from lammps import lammps , PyLammps
64  ## from mpi4py import MPI
65  from lammps import lammps
66  lmp = lammps()
67  #lmp.file(infile)
68  #L =PyLammps(ptr=lmp)
69
70  ##comm = MPI.COMM_WORLD
71  ##rank = comm.Get_rank()
72  ##nprocs = comm.Get_size()
73
74
75  lmp.command(f"units lj")
76  if dimension < 3:
77      lmp.command(f"dimension {dimension}")
78  lmp.command(f"atom_style atomic")
79  lmp.command(f"atom_modify map yes")
80  lmp.command(f"boundary p p p")
81  lmp.command(f"region box block -{hbl} {hbl} -{hbl} {hbl} -{rad} {rad} side in")
82  lmp.command(f"create_box 1 box")
83
84  # initialize configuration of non-overlapping hard disks
85  #lmp.command(f"create_atoms 1 random {Natoms} {seed0} box overlap {cutoff} maxtry
        1000")
86
87  # create lattice of particles (then only place Natoms of those)
88
89  ncnt = 0
90  z = 0.
91  y = -hbl + cutoff*0.5
92  while y <= hbl - cutoff*0.5 and ncnt < Natoms:
93      x = -hbl + cutoff*0.5
94      while x <= hbl - cutoff*0.5 and ncnt < Natoms:
95          x += cutoff
96          lmp.command(f"create_atoms 1 single {x} {y} {z}")
97          ncnt+=1
98          #lattice_coords.append([x,y,z])
99      y += cutoff
100
101
102
103  lmp.command(f"mass 1 1.")
104
105  # interactions
106  lmp.command(f"pair_style lj/smooth/linear {cutoff}")
107  lmp.command(f"pair_coeff * * {epsilon} {sigma}")
108
109  # outputs
110  lmp.command(f"dump 1 all xyz {Nxyz} {xyzdata_dir}{unique_sim_label}.xyz")
111  lmp.command(f"thermo {Nthermo}")
112  lmp.command(f"thermo_style custom step temp pe etotal press ")
113
114  #parallel stuff
115  #lmp.command(f"comm_style tiled")
116  #lmp.command(f"comm_modify cutoff {comm_cutoff}")
```

```
117
118  lmp.command(f"neighbor 0.3 bin")
119  lmp.command("neigh_modify every 1 delay 2 check yes")
120
121  # fixes
122  lmp.command(f"fix f0 all nve")
123  lmp.command(f"fix f1 all langevin {T} {T} {damp} {seed2}")
124  lmp.command(f"fix f2 all enforce2d")
125
126  # running params
127  lmp.command(f"timestep {dt}")
128
129  Natoms =lmp.extract_global("nlocal")
130  #if rank == 0:
131
132  print(f"\n\nPython Natoms = {Natoms}\n\n")
133
134  # implement dynamics
135  n = 0
136  while(n<Nruns-1):
137      #L.run(1)
138      #print(f"{L.runs[n].thermo.Step[0]} {L.runs[n].thermo.Temp[0]} ")
139      lmp.command(f"run 1 pre no post no")
140      n+=1
141  lmp.command(f"run 1 pre no post yes")
142
143  #print("Proc %d out of %d procs has" % (rank,nprocs),lmp)
144  #MPI.Finalize()
```

Listing 1: Minimal working example of a LAMMPS simulation using the python interface. The Code simulations repulsive disks.

References

1. Alberts, B. *et al. Molecular Biology of the Cell* 4th (Garland, 2002).

2. De Gennes, P. Soft Matter (Nobel Lecture). *Angewandte Chemie International Edition in English* **31,** 842–845 (1992).

3. Pak, A. J. *et al.* Advances in coarse-grained modeling of macromolecular complexes. *Current Opinion in Structural Biology* **52,** 119–126 (2018).

4. Hafner, A. E. *et al.* Minimal coarse-grained models for molecular self-organisation in biology. *Current Opinion in Structural Biology* **58,** 43–52 (2019).

5. Manning, M. L. Essay: Collections of Deformable Particles Present Exciting Challenges for Soft Matter and Biological Physics. *Physical Review Letters* **130** (2023).

6. Hollingsworth, S. A. *et al.* Molecular Dynamics Simulation for All. *Neuron* **99,** 1129–1143 (2018).

7. Frenkel, D. *et al. Understanding Molecular Simulation: From Algorithms to Applications* Second (Academic Press, San Diego, 2002).

8. Rapaport, D. C. *The Art of Molecular Dynamics Simulation* 2nd ed. (Cambridge University Press, 2004).

9. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* **117,** 1 (1995).

10. Thompson, A. P. *et al.* LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications* **271,** 108171 (2022).