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 5.1, in the linux sub-system of course, and the following subsections.

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

First create a LAMMPS conda environment:

```
    conda config  add  channels conda-forge
    conda create -n lammps-env
    conda activate lammps-env
    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/UNIX)

5.1 Upgrade the apt package manager

Run the following in a terminal:

```
    sudo apt update
    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:

```
    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):

```
python3 -V
```

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

```
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...

```
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:

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

5.3 Download and install LAMMPS

Next:

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

Then:

```
mkdir lammps
mv develop.zip lammps/
cd lammps/
unzip develop.zip -d ./lammps-develop-python/
```

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

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

Install LAMMPS, with the usually useful packages:

```
machine="serial"
echo " == Making LAMMPS-$machine for $nodename == "
echo "Machine filename: $machine"
make clean-${machine}
make no-all
make yes-EXTRA-FIX
make yes-EXTRA-PAIR
make yes-MANIFOLD
make yes-MOLECULE
make yes-RIGID
make yes-MANYBODY
make yes-BROWNIAN
make yes-PYTHON
make -j 4 mode=shared ${machine}

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:

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

6 Example (basic) LAMMPS simulation code

```python
## This uses solely a Python (LAMMPS API) approach
## system: WCA [repulsive] Brownian Particles
## Author: Dr. Luke Davis
## Year: 2022
## E: luke.davis@ucl.ac.uk

from __future__ import print_function
import numpy as np
import sys
import random
import math
import ctypes
import os

wd = "./"
xyzdata_dir = f"{wd}xyzdata/"

try:
    os.mkdir(xyzdata_dir)
except FileExistsError:
    # directory already exists so ignore
    pass

## SIMULATION PARAMETERS
dimension = 2 # spatial dimension
bl = 80 # box length
hbl = 0.5*bl
pack_frac = 0.01 # requested packing fraction


T = 1.0 # reduced temperature (all units are reduced according to LAMMPS units LJ)
dt = 0.004 #timestep
Nthermo = 10 # display thermodynamic information every this number of steps
Nxyz = 1000 # dump xyz file every this number of steps
Nruns = 10000 # number of total timesteps
damp = 1.0 # inverse mobility
instance = 0 # label to distinguish different random runs
#active_types = [2]

# INTERACTION POTENTIAL PARAMETERS
epsilon = 10.0
sigma = 1.0
rad = sigma*0.5
HS_sigma = round(sigma*(2./( 1. + math.sqrt(T/epsilon)) )**(1./6.),3) # approximate
   HS diameter
HS_rad = 0.5*HS_sigma
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)
comm_cutoff = cutoff*1.2
vol_atom = 0.
if dimension == 2:
    vol_atom = math.pi*HS_rad**2
Natoms = math.floor((pack_frac*(bl**dimension))/vol_atom) # number of atoms
dens = Natoms/(bl**dimension) # density
```

```python
unique_sim_label = f"RABPS_LAMMPS_N{Natoms}_bl{bl}_HSsig{HS_sigma}_T{T}_damp{damp}_dt
    {dt}_Nruns{Nruns}_instance{instance}"

# seeds for random things in LAMMPS script
seed0 = random.randint(10000,1000000)
seed1 = random.randint(10000,1000000)
seed2 = random.randint(10000,1000000)
seed3 = random.randint(10000,1000000)


# BEGIN LAMMPS SIMULATION
#from lammps import lammps, PyLammps
## from mpi4py import MPI
from lammps import lammps
lmp = lammps()
#lmp.file(infile)
#L =PyLammps(ptr=lmp)

##comm = MPI.COMM_WORLD
##rank = comm.Get_rank()
##nprocs = comm.Get_size()


lmp.command(f"units lj")
if dimension < 3:
    lmp.command(f"dimension {dimension}")
lmp.command(f"atom_style atomic")
lmp.command(f"atom_modify map yes")
lmp.command(f"boundary p p p")
lmp.command(f"region box block -{hbl} {hbl} -{hbl} {hbl} -{rad} {rad} side in")
lmp.command(f"create_box 1 box")

# initialize configuration of non-overlapping hard disks
#lmp.command(f"create_atoms 1 random {Natoms} {seed0} box overlap {cutoff} maxtry
    1000")

# create lattice of particles (then only place Natoms of those)

ncnt = 0
z = 0.
y = -hbl + cutoff*0.5
while y <= hbl - cutoff*0.5 and ncnt < Natoms:
    x = -hbl + cutoff*0.5
    while x <= hbl - cutoff*0.5 and ncnt < Natoms:
        x += cutoff
        lmp.command(f"create_atoms 1 single {x} {y} {z}")
        ncnt+=1
        #lattice_coords.append([x,y,z])
    y += cutoff



lmp.command(f"mass 1 1.")

# interactions
lmp.command(f"pair_style lj/smooth/linear {cutoff}")
lmp.command(f"pair_coeff * * {epsilon} {sigma}")

# outputs
lmp.command(f"dump 1 all xyz {Nxyz} {xyzdata_dir}{unique_sim_label}.xyz")
lmp.command(f"thermo {Nthermo}")
lmp.command(f"thermo_style custom step temp pe etotal press ")

#parallel stuff
#lmp.command(f"comm_style tiled")
#lmp.command(f"comm_modify cutoff {comm_cutoff}")
```

```

lmp.command(f"neighbor 0.3 bin")
lmp.command("neigh_modify every 1 delay 2 check yes")

# fixes
lmp.command(f"fix f0 all nve")
lmp.command(f"fix f1 all langevin {T} {T} {damp} {seed2}")
lmp.command(f"fix f2 all enforce2d")

# running params
lmp.command(f"timestep {dt}")

Natoms =lmp.extract_global("nlocal")
#if rank == 0:

print(f"\n\nPython Natoms = {Natoms}\n\n")

# implement dynamics
n = 0
while(n<Nruns-1):
    #L.run(1)
    #print(f"{L.runs[n].thermo.Step[0]} {L.runs[n].thermo.Temp[0]} ")
    lmp.command(f"run 1 pre no post no")
    n+=1
lmp.command(f"run 1 pre no post yes")

#print("Proc %d out of %d procs has" % (rank,nprocs),lmp)
#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).