Lab 4

Handout: Classical Molecular Dynamics of Silver Iodide *

Simone Cigagna[1] and Xing Wang[2]
[1] simone.cigagna@epfl.ch [2] xing.wang@psi.ch

May 2025

1 Molecular dynamics codes and references

In this Lab, we will be using the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) package to run classical molecular dynamics (MD) simulations. You should already be familiar with LAMMPS from the first Lab, but we will provide a brief review nonetheless.

We also suggest two books that are standard references for learning MD simulations:

- *Frenkel & Smit, Understanding Molecular Simulations, Academic Press (2002).* In particular, Chapter 4 gives a detailed description of the basics of MD, we recommend you read this part. A good description of the various ensembles is provided in Chapter 6. A nice explanation of the methods to treat electrostatic interactions is reported in the first part of Chapter 12, whereas the algorithms to speed up the calculation of non-bonded interactions are reported in Appendix F.
  Science Direct link

- *Allen & Tildesley, Computer Simulations of Liquids, Oxford (1987).* This text is more formal than the previous one, but it also contains some important derivations and formulas. Green-Kubo relations and other quantities important in MD runs are well described in Chapter 2. The expression of the errors in the statistical averages is quite clear and can be found in Chapter 6.4.
  Google Books link

2 Quick summary of molecular dynamics

In the MD technique, the time evolution of a set of N interacting atoms is followed by integrating the set of classical (Newton's) equations of motion

$$\vec{F}_i = m_i \vec{a}_i, \tag{1}$$

*This handout is a modified version of the notes written by former lecturers of this Lab.

where i labels the i^{th} atom in the system, m_i is the mass of the atom, $a_i = d^2\vec{r}_i/dt^2$ its acceleration, and $\vec{F}_i$ the force that acts on it, due to its interaction with the other atoms. In principle, from any set of initial positions and velocities, the positions $\{\vec{r}_i\}$, velocities $\{\vec{v}_i\}$ at each MD time step will be determined by knowing the forces and thus the acceleration. The force on each atom i, $\vec{F}_i$, is the negative of the derivative of the potential energy with respect to the position of the atom

$$\vec{F}_i = -\nabla_{\vec{r}_i} U \tag{2}$$

where, $U = U(\vec{r}_1, ..., \vec{r}_N)$ is the potential energy of the system. For simple systems (for example the harmonic oscillator), the equation of motion can be integrated analytically, but for more complex systems, we need to integrate numerically using a finite timestep, which can eventually cause the computed trajectory to deviate from the true trajectory.

MD can therefore be seen as a technique to follow the *time evolution* of a system: the trajectory of N atoms (i.e., the set of the $2N$ vectors $\{\vec{r}_i\}$, $\{\vec{v}_i\}$) is determined from the initial set of positions and velocities by numerically integrating the equations of motion (once given a form of the potential U).

However, one can also use MD as a *statistical mechanics method*, which is the more common use-case and also what this Lab is about. This means that we are not interested in the trajectory itself, but rather in a set of configurations distributed according to a statistical ensemble (i.e. to some distribution function). For example, if we place N atoms in a box with constant volume V and keep their total energy E (kinetic + potential) constant during the simulation, it automatically follows that each time step of the system trajectory will be a point in the *microcanonical* phase space (N, V, E) at those constant values N, V, E. Statistical mechanics calculates thermodynamic variables as averages of the corresponding (microscopic) dynamical variables over the phase space (i.e. the $6N$ coordinates and momenta) multiplied by a distribution function appropriate to that ensemble. Such an average, indicated by the angular brackets $\langle\rangle$, is indeed called *ensemble average*. Below we provide the ensemble averages in the microcanonical (NVE) and canonical (NVT) ensembles of a quantity A:

$$\langle A \rangle_{N,V,E} = \frac{\int A(\vec{r},\vec{p})\delta(\mathcal{H}(\vec{r},\vec{p}) - E)d\vec{r}d\vec{p}}{\int \delta(\mathcal{H}(\vec{r},\vec{p}) - E)d\vec{r}d\vec{p}} \tag{3}$$

$$\langle A \rangle_{N,V,T} = \frac{\int A(\vec{r},\vec{p})e^{-\beta\mathcal{H}(\vec{r},\vec{p})}d\vec{r}d\vec{p}}{\int e^{-\beta\mathcal{H}(\vec{r},\vec{p})}d\vec{r}d\vec{p}} \tag{4}$$

where $\mathcal{H}$ is the Hamiltonian used and $A(\vec{r},\vec{p})$ is the value of A at the point $\vec{r},\vec{p}$ in phase space. If we let the system evolve in time, after a reasonable time evolution, it will explore different regions of the phase space according to its ensemble distribution function. This means that, after a long enough evolution time, the temporal average along the trajectory will be equal to the ensemble average over the phase space, also known as the *ergodic hypothesis*. We can thus use our trajectory as a sequence of "samples" to be averaged in time to obtain the macroscopic thermodynamic quantity of interest:

$$\bar{A} = \lim_{\tau\to\infty} \frac{1}{T} \int_0^\tau A(\vec{r}(t)\vec{p}(t))dt \to \langle A \rangle \tag{5}$$

From now on we will use the angular brackets $\langle\rangle$ simply to mean a time integral over our trajectory, as implied by the ergodic hypothesis.

According to these considerations, any MD code will look like the following pseudo-algorithm which includes five main steps:

```
program md
    call initialize
    loop
        call compute_energy_and_forces
        call integrate_equation_of_motion
        call sample_averages
        if (converged_averages .or. exceed_time) exit
    end loop
end program md
```

In short, we need to: 1) Initialize positions and velocities, 2) calculate the energies and forces, 3) integrate the equations of motion, 4) sample averages and 5) check the convergence of the average quantities or wall time.

In the next sections we will briefly explain these steps to get you familiar with classical MD. We advice you to consult the provided references and, in case you have doubts, ask the tutors.

2.1 Initialization of positions and velocities

MD codes that deal with solids and liquids employ periodic boundary conditions (PBC). But, it is not mandatory, if you work with molecules you can turn the PBCs off. When using PBCs, the cell vectors and starting positions are explicitly provided in the input file. As an alternative, an ordered crystal structure can be chosen as starting point. When starting a new run, velocities can be given in the input (for example, in case of restarting from a previous run) or can be randomly generated according to the Maxwell-Boltzmann or another kind of distribution.

Both the initial positions and the initial velocities can be quite far from the equilibrium at the beginning of the simulation. Thus, except when restarting a previously equilibrated simulation, an equilibration run should be performed to bring the system to the desired thermodynamic state. The length of the equilibration run depends on how far the starting state of the system is from the required one and is very system dependent. For example, to get liquid water at 350 K will require a much shorter equilibration time by starting an equilibration run from a configuration of liquid water obtained at 300 K, than it would by starting from ideal crystalline ice. Effective achievement of the final thermodynamic state can be difficult to detect, especially when dealing with slow, non-ergodic systems, such as liquids at low temperatures or amorphous solids. In these cases, a careful estimation of the error in the averages of the thermodynamic quantities of interest (e.g., temperature, volume, pressure, etc.) should be performed.

2.2 Empirical energy and forces

Calculation of energy and forces depends on the choice of the interaction potential. In Lab 4 exercises, we will study silver iodide (AgI), which exhibits a *superionic phase* where the silver ions are mobile, while the iodine ions maintain a rigid crystal structure. To accurately describe this peculiar phase, we will employ a promising[1] Morse-Coulomb hybrid pair potential with parameters derived from DFT simulations.

2.3 Integration of the equations of motion

Newton's equation of motion can be integrated numerically by means of several different integration algorithms. Preservation of the phase space volume and time reversibility are important features of some of the most commonly used integrators, such as the velocity Verlet and the leap-frog algorithm. Nonetheless, in order to speed up the calculations some less accurate schemes can be adopted. The Verlet algorithm is very well described in the Allen & Tildesley book, it consists of simple Taylor series expansions around $t + \Delta t$ and $t - \Delta t$ summed up, giving a local error on the atomic positions of the order of Δt^4 per integration step.

In order to simulate a system in the canonical (NVT) ensemble, a thermostat is added to the equations of motion. Some of the most commonly used thermostats include stochastic velocity rescaling, Berendsen and Nose-Hoover thermostats. Again, the interested reader can find this in the book Allen & Tildesley. It should be noted that some of these thermostats require the tuning of their frequency, which is related to how strongly the system would be coupled to the thermostat. If the coupling is too strong, the system will thermalize fast, but quantities will be affected by the thermostat. If the coupling is very weak, the system will not reach the desired temperature.

2.4 Sample averaged quantities

MD is a tool for computing thermodynamic quantities from time averages under the assumption of ergodicity (see above). As the system visits different regions of the configuration space, most of the quantities that are computed in an MD run fluctuate. The average of these quantities over the entire simulation will provide an estimate of the thermodynamic quantity of interest. Even though the ergodic theorem is valid only for infinitely long MD simulations, a simulation can be stopped when the mean has converged. Suppose we are analyzing simulation results that contains a total of t_{run} time steps, or configurations. The run average of some property A is:

$$\langle A \rangle_{run} = \frac{1}{t_{run}} \sum_{t=1}^{t_{run}} A(t) \tag{6}$$

and its variance is:

$$\sigma_A^2 = \frac{1}{t_{run}} \sum_{t=1}^{t_{run}} \left(A(t) - \langle A \rangle_{run} \right)^2 . \tag{7}$$

[1] Niu, Hongwei, Yuhang Jing, Yi Sun, and Narayana R. Aluru. "Ab Initio Based Interionic Potential for Silver Iodide." Solid State Ionics. Elsevier BV, November 2018..

Looking at the fluctuations of a quantity A is very important in some cases. Firstly, fluctuations inform you how much you can trust your results:

- we will see in the Lab how the fluctuations (in this case standard deviation) in the temperature in a microcanonical run scale with system size (i.e. the number of atoms). In such a case, large fluctuations are an indication that you are very far from the thermodynamic limit;
- other kinds of fluctuations have a direct physical interpretation: the specific heat capacity can be directly computed from variance in the total energy in the canonical ensemble.

For an extensive explanation on error estimates in MD simulations, refer to Allen & Tildesley Chapter 6. But please note that this is not needed for this assignment.

3 LAMMPS getting started

3.1 Before you begin any calculation

Compared to the *ab-initio* calculations in Lab 2 and Lab 3, the classical MD simulations in this Lab are faster and require less memory. Therefore we don't need to use any high-performance supercomputing resources. As in previous Labs, we provide a `LAB4.zip` file which contains an example of how to run the MD simulation with LAMMPS. Unzip the file and copy its contents to the virtual machine.

3.2 LAMMPS input data parameters

Now you will perform a test run of LAMMPS. Assuming you have LAB4 in your shared folder, you first create the directory `Test` and then copy the input file that we provided to that directory.

```
max@mse468:~/LAB4$ mkdir Test
max@mse468:~/LAB4$ cd Test
max@mse468:~/LAB4/Test$ cp ../examples/AgI.in .
max@mse468:~/LAB4/Test$ cp ../examples/restart_AgI.in .
```

The file `AgI.in` is an example of input file that contains the information needed to compute a MD run at constant number of particles, volume and energy (NVE). You can view the input file with the command:

```
max@mse468:~/LAB4/Test$ less AgI.in
```

it will look like this:

```
# LAMMPS input script for a molecular dynamics simulation of alpha-AgI using combination of Cou
# By Lab 4 team (Spring 2025)

# Variables
# Only make changes here!
variable lattice_parameter equal 5.37 # Angstrom
variable supercell equal 2
variable timestep equal 0.005
variable stride equal 100
variable temperature equal 300.0
variable t_damp equal ${timestep}*100.0
variable sim_length equal 5000


# Initialise simulation
# Variables for AgI; after https://aluru.web.engr.illinois.edu/Journals/SSI18.pdf
# species 1 = (I)
# species 2 = (Ag)
# Conventional cell creation
dimension       3
units           metal
boundary        p p p
atom_style      charge
lattice         custom   1.0     &
        a1      ${lattice_parameter}       0.0     0.0     &
        a2      0.0      ${lattice_parameter}      0.0     &
        a3      0.0      0.0     ${lattice_parameter}      &
        basis   0.00  0.00  0.00              &
        basis   0.50  0.50  0.50              # Lattice for I system
region          box block 0 1 0 1 0 1 units lattice
create_box      2 box # 2 atom-type number (I have two species), box is the name
create_atoms    1 box # type 1 is I
lattice         custom   1.0     &
        a1      ${lattice_parameter}       0.0     0.0     &
        a2      0.0      ${lattice_parameter}      0.0     &
        a3      0.0      0.0     ${lattice_parameter}      &
        basis   0.25  0.00  0.50              &
        basis   0.75  0.50  0.00              # Lattice for Ag system
create_atoms    2 box # type 2 is Ag

# Masses and charges
mass 1 126.90447      # mass I
mass 2 107.8682       # mass Ag
set type 1 charge -0.3181
set type 2 charge  0.3181

# Hybrid potential: Morse + Coulomb
```

```
kspace_style ewald 1.0e-6
pair_style hybrid/overlay coul/long 10.0 morse/smooth/linear 10.0
pair_coeff * * coul/long
#          type type          D      alpha   r0
pair_coeff   1    2   morse/smooth/linear  0.5500 1.6000 2.6000
pair_coeff   1    1   morse/smooth/linear  0.1600 0.6840 5.7000

replicate  ${supercell} ${supercell} ${supercell}

# Define simulation parameters
timestep ${timestep}

# Set initial temperature and velocity
velocity all create ${temperature} 87287 loop geom

# Following will be returned in the output file
thermo 5000
thermo_style custom cella cellb cellc

# Run equilibration for the first 10000 timesteps in canonical ensemble
fix 1 all nvt temp ${temperature} ${temperature} ${t_damp}
run 5000

# Calculate mean square displacement
group I type 1
group Ag type 2
compute mymsdI I msd com yes
compute mymsdAg Ag msd com yes
variable msd_normI equal c_mymsdI[4]
variable msd_normAg equal c_mymsdAg[4]

# Following will be returned in the output file
thermo ${stride}
thermo_style custom time temp pe ke press v_msd_normI v_msd_normAg

# Dumping positions and velocities
dump dp1 all custom ${stride} positions.lammpstrj id type xu yu zu
dump dp2 all custom ${stride} velocities.lammpstrj id type vx vy vz
dump_modify dp1 format line "%d %d %g %g %g"
dump_modify dp2 format line "%d %d %g %g %g"

# Run production simulation
unfix 1
fix 2 all nve
run ${sim_length}

# Dumping recovery file
```

```
95 write_data recovery.data
```

NOTE: Lines starting with # are ignored by LAMMPS and are used as comments to explain the input file.

Below you will find a brief explanation for each of the input parameters. For an exhaustive explanation of the physical meanings of these parameters please refer to the html version of the LAMMPS manual, or by clicking on the links that we provide for each of them.

- lines 6-12
  To specify all the variables used in the simulation. This is the only place where you should make changes to the inputs like timestep, temperature, etc. Mind the syntax for performing mathematical operations on variables in line 11.
  See `variable`.

- line 20
  To set the dimensionality of the simulation, so 3 for a 3D simulation.
  See `dimension`.

- line 21
  This specifies all the units of the input quantities and also the units in which quantities are written in the output. In our case we use the style `metal`.
  See `units`.

- line 22
  Enable periodic boundary conditions
  See `boundary`.

- line 23
  To setup the attributes the atoms can achieve in a simulation. For example the atomic attributes would depend on the underlying potential model used in a simulation, a metallic system would have different attributes compared to biological molecules.
  See `atom_style`.

- lines 24-39
  The first command defines the lattice of the system. The following two setup the simulation `box` that contains all the atoms generated by `create_atoms` command. We create two lattices, one for each species.
  See `lattice`, `region`, `create_box` and `create_atoms`

- lines 42-45
  To set the mass and charges of the atoms. The units are those previously specified by the `units` command.
  See `mass` and `set`

- lines 48-53
  To define the type of interaction potential and the coefficients of the potential.
  See `pair_style`, `pair_coeff`, `kspace_style`

- line 55
  To replicate the unit cell in the three dimensions. The number of unit cells in each direction is defined by the variable `supercell` defined in line 8.
  See `replicate`.

- line 58
  To setup the timestep of the simulation. Note that we use a variable defined in line 9 here for better readability and editing of the input script.
  See `timestep`.

- line 61
  To initialize the velocities sampled from Maxwell-Boltzmann distribution based on a given temperature and a random seed. Please keep the same seed to ensure reproducibility of your simulations.
  See `velocity`.

- line 64-65
  To write specific quantities evaluated during a simulation. In this case the cell vectors are outputted every 5000 timesteps.
  See `thermo` and `thermo_style`.

- lines 68-69
  To setup the kind of simulation you want to run. In this case we are running an MD simulation in the canonical ensemble to equilibriate the system at the desired temperature. `run` command simply runs the simulation for 5000 timesteps.
  See `fix` and `run`.

- line 72-77
  To compute MSD and store it in the variables `msd_norm_I` and `msd_norm_Ag`.
  See `compute` and `group`.

- line 80-81
  Same as lines 63-64, this time we output time, temperature, potential energy, kinetic energy, pressure, and msd every 100 `timesteps`.

- lines 84-87
  Dumping positions and velocities in LAMMPS trajectory format.
  See `dump` and `dump_modify`.

- lines 90-92
  Delete the fix command, in this case, we delete the canonical ensemble (equilibration run) and then run a microcanonical ensemble as production.
  See `unfix`.

- line 95
  Dumps recovery file, if simulation is interrupted. Useful in cases of extremely long and expensive simulations.
  See `write_data`.

3.3 Running the classical molecular dynamics

Now there two ways to execute LAMMPS:

```
max@mse468:~/LAB4/Test$ lmp_mpi < AgI.in > AgI.out  &
```

where the optional `&` lets you keep control of the terminal while the job is running. This will be usually a slower and an inefficient way of running LAMMPS on only a single processor. To run LAMMPS on multiple processors, you can use `mpirun`. So run on two processors execute following command:

```
max@mse468:~/LAB4/Test$ mpirun -np 2 lmp_mpi -in AgI.in > AgI.out
```

We recommend running LAMMPS in parallel to significantly reduce simulation time, especially for larger systems or longer simulations. For more information on how to run LAMMPS refer to the documentation (see running).

Once the program has been executed, a few files are written. The `AgI.out` file contains all the details of the input files and of the run itself. This file is quite similar to `log.lammps` that contains more information regarding the run. Another file called `recovery.data` is used to restart a simulation when the job crashes or when you want to improve the statistics of your calculations. Two more files are generated containing positions and velocities in LAMMPS trajectory format. To restart the calculation, first we save the output files of the first run:

```
max@mse468:~/LAB4/Test$ cp AgI.out old_AgI.out
max@mse468:~/LAB4/Test$ cp positions.lammpstrj old_positions.lammpstrj
max@mse468:~/LAB4/Test$ cp velocities.lammpstrj old_velocities.lammpstrj
```

Then, use the `restart_AgI.in` as input file[2]:

```
max@mse468:~/LAB4/Test$ mpirun -np 2 lmp_mpi -in restart_AgI.in > AgI.out
```

You can modify the `AgI.in` file to change supercell size, timestep, stride i.e. after how many timesteps quantities should be written in the output (we suggest keeping it as is), temperature, damping factor of the thermostat (we suggest keeping it as is), an the duration of the MD run.

3.4 Using bash scripts to run LAMMPS multiple times

In order to check the numerical convergence of your calculations with respect to MD time-step, supercell size etc, you will have to run LAMMPS multiple times with different values for those parameters. Of course you may create a new input file for every set of parameters, but using `bash` scripts can help you to speed up and automate your work. In this Lab we provide you (in the `/LAB4/scripts` directory) with a useful `bash` script, called `script.sh`, that automatically loops over the size of the cell, temperature and timestep values. You can also write a `bash` script by yourself, or modify this one according to your own taste or even use Python to automate this process.

The script looks like this:

[2]this file is a simpler version `AgI.in`, check out the documentation for the `read_data` command.

```
#!/bin/bash -f

# Script written to run LAMMPS multiple times on one node.
# The script can loop over 3 different input parameters:
### the time step, specified in list_time_steps
### The temperature, specified in list_temperatures
### The supercell size, specified in list_supercell_size

# There are several ways to set your list in BASH.
### For explicit definition of e.g. your time_steps, you can do
####### list_supercell_size="2 3 4 5 6 7 8 9 10" #######

### You can also use the seq command to create a sequence as in:
####### list_temperatures=‘seq 1000 100 3000‘ #######
### This creates a sequence of values between 1000 and 3000, every 100

list_time_steps="0.0001 0.0005 0.001 0.002 0.003 0.005 0.007 0.01"
list_temperatures="300"
list_supercell_size="2"

##################### !IMPORTANT! #####################
#   To run in canonical ensemble for problem 2 comment out line 100 and uncomment line 101
#   Remember to change the 2 to 1 if your virtual machine is running on 1 CPU in line 104
##################### !IMPORTANT! #####################

# This is the executable path for LAMMPS
exec=‘which lmp_mpi‘

# This is the executable path for the parser
# Change it if you moved the ‘parser.py‘ to some other location
parser="python3 /home/max/Desktop/SHARED/LAB4/scripts/parser.py"

# These are constants that the script does not loop over.
# You need to change them during the exercise.
# Equilibration and production time in ps, lattice parameter in Angstrom and
# how often (in ps) everything will be outputted
sim_time="6"
equilibration_time="5"
lattice_parameter="5.37"
stride_time="0.1"

# Start loops
for supercell in $list_supercell_size; do
    for temperature in $list_temperatures; do
        for time_step in $list_time_steps; do
            sim_steps=$(echo "scale=0; $sim_time / $time_step" | bc)
            stride=$(echo "scale=0; $stride_time / $time_step" | bc)
```

```
            equilibration_steps=$(echo "scale=0; $equilibration_time / $time_step" | bc)
            base_name="${supercell}_${temperature}_${time_step}"
            cat > md_${base_name}.in << EOF
# LAMMPS input script for a molecular dynamics simulation of Fe using an EAM potential

# Initialise simulation
clear
# Variables for AgI; after https://aluru.web.engr.illinois.edu/Journals/SSI18.pdf
# species 1 = (I)
# species 2 = (Ag)
# Conventional cell creation
dimension       3
units           metal
boundary        p p p
atom_style      charge
lattice         custom   1.0     &
        a1      ${lattice_parameter}       0.0     0.0     &
        a2      0.0      ${lattice_parameter}      0.0     &
        a3      0.0      0.0     ${lattice_parameter}      &
        basis   0.00  0.00  0.00              &
        basis   0.50  0.50  0.50              # Lattice for I system
region          box block 0 1 0 1 0 1 units lattice
create_box      2 box # 2 atom-type number (I have two species), box is the name
create_atoms    1 box # type 1 is I
lattice         custom   1.0     &
        a1      ${lattice_parameter}       0.0     0.0     &
        a2      0.0      ${lattice_parameter}      0.0     &
        a3      0.0      0.0     ${lattice_parameter}      &
        basis   0.25  0.00  0.50              &
        basis   0.75  0.50  0.00              # Lattice for Ag system
create_atoms    2 box # type 2 is Ag
# Masses and charges
mass 1 126.90447      # mass I
mass 2 107.8682       # mass Ag
set type 1 charge -0.3181
set type 2 charge  0.3181
# Ewald sums
kspace_style ewald 1.0e-6
# Hybrid potential: Morse + Coulomb

pair_style hybrid/overlay coul/long 10.0 morse/smooth/linear 10.0
pair_coeff * * coul/long
#          type type          D      alpha   r0
pair_coeff   1    2   morse/smooth/linear  0.5500 1.6000 2.6000
pair_coeff   1    1   morse/smooth/linear  0.1600 0.6840 5.7000

replicate  ${supercell} ${supercell} ${supercell}
```

```
95
96
97  # Define simulation parameters
98  timestep ${time_step}
99
100 # Set initial temperature and velocity
101 velocity all create ${temperature} 87287 loop geom
102
103 # Following would be returned in the output file
104 thermo ${stride}
105 thermo_style custom cella cellb cellc
106 # Run equilibration for the first 5000 timesteps in canonical ensemble
107 fix 1 all nvt temp ${temperature} ${temperature} $(echo "$time_step * 100.0" | bc)
108 run ${equilibration_steps}
109
110 # Calculate mean square displacement
111 group I type 1
112 group Ag type 2
113 compute mymsdI I msd com yes
114 compute mymsdAg Ag msd com yes
115 variable msd_normI equal c_mymsdI[4]
116 variable msd_normAg equal c_mymsdAg[4]
117
118 # Following would be returned in the output file
119 thermo ${stride}
120 thermo_style custom time temp pe ke press v_msd_normI v_msd_normAg
121
122 # Dumping positions and velocities
123 dump dp1 all custom ${stride} positions_${base_name}.lammpstrj id type xu yu zu
124 dump dp2 all custom ${stride} velocities_${base_name}.lammpstrj id type vx vy vz
125 dump_modify dp2 format line "%d %d %g %g %g"
126
127 # Run production simulation
128 unfix 1
129 fix 2 all nve
130 # fix 2 all nvt temp ${temperature} ${temperature} $(echo "$time_step * 100.0" | bc)
131 run ${sim_steps}
132 EOF
133             mpirun -np 2 $exec -in md_${base_name}.in > md_${base_name}.out
134             $parser md_${base_name}.out positions_${base_name}.lammpstrj velocities_${base_name
135         done
136     done
137 done
```

As you can see, the script is well commented. And, from line 51 to line 129 you will have already recognized the input file that we described above, with the addition that some parameters are inserted based on the variable value in the loop. The only new command here is `clear` which is used to set all settings to their default values (see clear). Here you

have the possibility of changing and looping over the values of timestep, temperature and cell size by setting the respective "lists".

First copy the script to your test directory by:

```
max@mse468:~/LAB4/Test$ cp ../scripts/script.sh .
```

You can now run the script by using:

```
max@mse468:~/LAB4/Test$ bash script.sh
```

or

```
max@mse468:~/LAB4/Test$ nohup ./script.sh &
```

The nohup part allows the program to keep running even if you close the terminal in the computer. At the end, you will find all the output files in your working directory. Type the command `ls -lh` to get a list of the produced files. In the problems of this Lab, you can modify the script appropriately in order to run the different calculations that you might need.

The LAMMPS program will generate two `.lammpstrj` trajectory files that contain the configurations generated during the MD simulation. We provide a parser (`parser.py`) in the `scripts` directory[3] that takes the output file and trajectory of LAMMPS and provides:

- A text file that for every sampled trajectory point lists the time, temperature, potential energy, kinetic energy, pressure and msd. You specify the text file name with the `-e` option. This file can be easily visualized with gnuplot.
- A JSON file that contains positions, velocities etc, that will be an input to the other analysis scripts we provide. The filename is the last positional input.

As a first example of analysis script, `python3 scripts/converter_to_xcrysden.py output.json` takes the JSON output you provide and writes a file that can be visualized with xcrysden, a program you have used before. Note that all possible input options to a python script can be viewed when passing `-h`, as in `python3 myscript.py -h`.

4 Thermodynamics of bulk silver iodide (exercise)

In this section, we study the thermodynamics of bulk silver iodide employing an empirical force field to describe the interatomic potential.

In order to solve the problems in Lab 4, you will have to run several MD calculations at different temperatures (within 293 K and 800 K). In the first part of the exercise you will be asked to determine the optimal parameters in order to ensure that the structural and dynamical properties that you will compute in the second part of the exercise are converged.

[3] In order to speed up the setup and the analysis of the simulations, we provide some useful python scripts that you can find in the scripts folder. All the scripts are intended to simplify your life during the set up of the simulations required by the different exercises. You are not forced to use them, you can analyze each calculation as you wish.

4.1 Convergence of the parameters

When starting a simulation, it is fundamental to have well converged parameters. These usually include the timestep of the simulation, the supercell size, and the production run length, which must be large enough to allow the sampling of uncorrelated structures.

A good choice for the equilibration time is at least 5000 steps. You can also try 10000 or more, as LAMMPS is quite fast it shouldn't incur any noticeable cost.

In general, you should choose a `timestep` large enough in order to save CPU time, but small enough to minimize integration errors, that can lead to an unstable simulation with non-physical behavior. The main limitation imposed by the system here is the highest-frequency motion that must be considered: a vibrational period must be split into around 100 segments for molecular systems to satisfy the Verlet assumption that the velocities and accelerations are constant over the timestep used. A very safe timestep in this system is around 10^{-15} seconds (= 0.001 ps = 1 fs). The standard way to test the convergence of your `timestep` is to measure the energy conservation in the NVE ensemble, because in this case total energy (kinetic + potential) is expected to be constant. If the `timestep` is too big, the total energy may drift or the atoms may crash into each other. A practical hint is to use the highest temperature and the smallest supercell that you are interested in. As the temperature increases, the vibrational frequencies are higher, and the vibrational periods are shorter, thus if your `timestep` is converged for the higher temperatures it will also be converged for the lower temperatures. Notice that the conserved quantity, even if there is no drift, is not truly constant, as there are small fluctuations that are due to the numerical integration. Thus, the amplitude of these standard deviation as a function of timestep can be used to study how the integration error scales. This convergence test should be done for a production time of 6 ps (with extra 5 ps of equilibration). Remember to analyze only the production part of your simulation: the equilibration part in LAMMPS has a velocity-rescaling algorithm to speed up the equilibration which induces strong oscillations in the total energy. By default, the scripts provided will automatically neglect the equilibration part for all post processing.

Once you have a good timestep (which you should use from now on), you are asked to check how much the temperature fluctuates around the average in an NVE simulation as a function of the supercell size. As seen above, the latter can be easily tuned in the calculation by changing the parameter "`supercell`". As a rule of thumb, the standard deviation of the temperature should be below 3% of the target temperature, i.e., the one used in the equilibration run and set up in the input file. You should check both the highest and lowest temperatures, and use the smallest supercell that converged for both temperatures. To see how the fluctuations decrease with increased system size, perform a linear fit on $\log(\sigma_T)$ against $\log(N)$, where N is the number of atoms in the system. Also here, 5 ps of equilibration and 6 ps of production are fine.

NOTE: Always use a production time, denoted as `sim_time` in the input script, greater than 5 ps, otherwise you will need to make necessary changes to the `vaf.py` and `rdf.py` scripts.

As a last step, you should figure out how long the `production` time should be. As everything else, this quantity is also something you would ideally converge for every quantity of interest that you get get from your simulation, but this can be very cumbersome. A faster

estimation of the production time can be obtained if we know the effective decorrelation time of our system, that is to say the time it takes for a trajectory to lose memory. That can be estimated by looking at the velocity auto-correlation function (VAF):

$$\mathrm{VAF}(t) = \frac{1}{3N} \sum_{I}^{N} \sum_{\alpha=1}^{3} \langle V_{I\alpha}(t) V_{I\alpha}(0) \rangle, \tag{8}$$

where $V(t)_{I\alpha}$ stands for the velocity of particle I at time t in direction α and $\langle\rangle$ denotes a time average over the trajectory. From some time t_c (called the decorrelation time) onwards, this quantity fluctuates randomly around 0. Only simulation times above t_c give us new information about the system. If you simulate $50 \times t_c$, you will have 50 independent estimates of your quantity to average, and that is usually a sufficient choice. The VAF can be computed using the script `vaf.py` provided in the `scripts` directory.

```
max@mse468:$ python3 scripts/vdf.py output_xxx.json
```

Problem 1 - Convergence tests [35 points]

A) Determine a suitable timestep for molecular dynamics simulations at the highest simulated temperature (800 K) by analyzing the conservation of the total energy (the constant of motion, defined as the sum of the potential and kinetic energies). You can read the energies (E for potential energy and K for kinetic energy) from the output file (e.g., `E_of_t_6_800_0.001.dat`).

Since the total energy is not perfectly constant due to numerical integration errors, quantify these fluctuations by calculating the standard deviation of the total energy over time for different timesteps. Plot the standard deviation of the total energy as a function of the timestep. [5 points]

Based on your simulations, select a timestep where the standard deviation of the total energy fluctuations is below 1×10^{-5} eV/atom. [5 points]

Note: The standard deviation should be reported *per atom*. The unit cell contains 4 atoms; so e.g. in a $2 \times 2 \times 2$ supercell there are 32 atoms.

B) Converge the system supercell size such that the standard deviation of temperature is below 3% of the target temperatures (293 K and 800 K). [5 points]

How does the standard deviation decrease with increased system size (number of atoms)? [5 points]

C) Estimate the ideal production length from the velocity auto-correlation function VAF(t) at room temperature (293 K). [5 points]

Explain why this estimate should be made at the lowest temperature you plan to simulate. [5 points]

Derive how the value of VAF($t = 0$) is related to $\langle E_{kin} \rangle$, and check this for one of your trajectories. [5 points]

4.2 Structural and dynamical properties of silver iodide

In this section we perform MD simulations at constant volume and temperature (NVT) to compute structural and dynamical properties of silver iodide. To run in the NVT ensemble, please refer to the `ensemble nvt` part of the above commented LAMMPS input file.

Throughout this problem, you will be required to consider a significant number of different temperatures and extract several structural and dynamic information from the output of MD calculations.

4.2.1 Radial distribution function

The radial distribution function is one of the major observables to determine the structure of a system. In a pair potential, it uniquely determines all the thermodynamic properties.

The radial distribution function (RDF) measures the density of atoms (normalized by its average value ρ) as a function of the distance d from a reference one. It is calculated by summing the number of atoms found at a given distance in all directions. For a multi component system, the RDF can be computed for each species individually. In a solid, the radial distribution function will consist of sharp peaks, whereas as a material melts these peaks will broaden and some will disappear, as you can see in Figure 1.

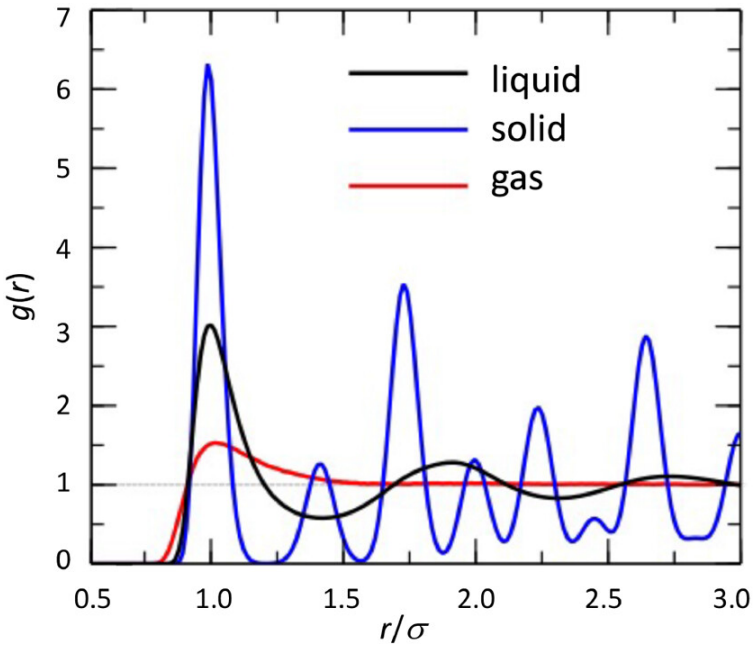

Figure 1: Examples of radial distribution functions for different phases of matter (source: 100 Years of the Lennard-Jones Potential).

You can use the provided script to calculate the radial distribution function:

```
max@mse468:$ python3 scripts/rdf.py output.json
```

You have several options to choose from, which you can access with the `-h` option. Such options are `bin-size`, the `stepsize` that you can use to speed up the calculation by not considering every step, etc. For example:

```
max@mse468:$ python3 scripts/rdf.py output.json --init-time 5.0 --n-bins
100 --stepsize-t 10 -o RDF.dat
```

This command writes the histogram in an output file (`RDF.dat` by default). As in other scripts, the initialization time is the interval of time to discard from the start of the trajectory when performing statistics. Note that the python script `rdf.py`, by default, uses only data from the production part of the LAMMPS run, the equilibration part (see equilibration flag in the `.in` file) is discarded by default. So, with `--init-time` $= 0$, all data from the production part will be considered.

The script computes the RDF up to the maximum radial distance in the cell, which is half the cell size in a cubic cell. As customary, the RDF is normalized to the mean density in the system. The script also integrates the RDF, and you can use this integral to estimate the number of nearest neighbors. The distance r, RDF(r), and $\int \text{RDF}(r)dr$ are written in the first, second, and third columns in the output file produced by the script.

The number of bins to use is a number you have to tune: too many bins will give you a histogram that is not a smooth function, whereas with too few bins you lose resolution. When you have figured out a good bin size, plot the radial distribution functions for the different simulated temperatures.

4.2.2 To diffuse or not to diffuse

The mean square displacement (MSD) is useful to determine if atoms can move freely or are bound to their position. With the MSD one can obtain the self-diffusion coefficient (D), which is related to how fast the atoms are moving in the system, and can also be obtained experimentally, thus it is a good benchmark for the simulations. The MSD is defined as:

$$\text{MSD}(t) = \frac{1}{3N} \sum_{I}^{N} \sum_{\alpha=1}^{3} \langle |R_{I\alpha}(t) - R_{I\alpha}(0)|^2 \rangle \tag{9}$$

where $R(t)_{I\alpha}$ stands for the coordinate α of particle I at time t and $\langle\rangle$ denotes a time average over your trajectory. From the MSD we obtain the self-diffusion coefficient, using the Einstein relation, as follows:

$$\text{D} = \frac{1}{2d} \lim_{t\to\infty} \frac{d}{dt} \text{MSD}(t), \tag{10}$$

where d is the dimensionality of the system, in our case 3, as we are dealing with a three-dimensional system. One should note that the limit to infinite time is being taken, whereas the simulations are finite. Thus, a careful evaluation of convergence as a function of time should be performed. Note also that, since we are dealing with a material that is a superionic conductor, i.e., a system made by a diffusive subsystem and a non-diffusive one, it is crucial here to distinguish between the diffusive behaviour of Ag and I. To calculate the self-diffusion coefficients, the MSD for both Ag and I are already outputted by LAMMPS. The diffusion coefficient can then be estimated from the slope of the MSD curve.

4.2.3 Arrhenius equation

The Arrhenius equation describes the temperature dependence of the rate constant of a reaction, such as the diffusion coefficient. Indeed it is natural to expect an increase of the diffusion coefficient with increasing temperature, as the atoms have more energy and can move more easily. Empirically, it can be shown that

$$D = D_0 \exp\left(-\frac{E_a}{RT}\right) \tag{11}$$

where D_0 is a pre-exponential factor, E_a is the (molar) activation energy, R is the gas constant[4] and T is the temperature. In the literature, it is standard practice to show instead the linear relation between $\log(D)$ and $1/T$. In this exercise you will be asked to follow this convention and estimate D_0 and E_a through a linear fit.

Problem 2 - Structural and dynamical properties [65 points]

IMPORTANT: Make sure to change your ensemble to NVT in the input script before attempting this problem.

A) Plot the radial distribution function (RDF) at room temperature (293 K) for both species [5 points].
Then, show the same plot at a very low temperature (e.g. 10 K) and explain the differences [5 points].
NOTE: express all lengths in angstrom [Å].

B) For both Ag and I ions, plot the MSD(t) as a function of time for six distinct temperatures within the range 293 K to 800 K.
What can you infer about the mobility of the two species as the temperature changes? [10 points].
NOTE: Give the MSD in units of [Å^2] and times in picoseconds [ps].

C) Estimate the diffusion coefficient for Ag at room temperature [10 points].
HINT: focus on a region of the MSD(t) curve that is roughly linear and perform a fit.
NOTE: to be consistent with the literature, the diffusion coefficient should be expressed in units of [cm^2/s].

D) Repeat the procedure from the previous point to calculate the diffusion coefficient for Ag at all six temperatures you simulated for point B [15 points].
Subsequently, plot the natural logarithm of the diffusion coefficients versus the inverse of the temperature. Fit the data to verify Arrhenius law and provide an estimate for the activation energy E_a and the pre-exponential factor D_0 [20 points].
NOTE: report the activation energy in units of [cal mol^{-1}].

[4] This is related to the Boltzmann constant k_b through the Avogadro number N_A, $R = k_b N_A$.

NOTE: Make sure to document all relevant simulation parameters. Discuss your results, and explain how you arrived at your conclusions. Put meaningful labels and units on plots.