

Practical work on Molecular Dynamics: Phase Transition

Master program DMS

February 13, 2018

1 Introduction

In this practical work we will simulate phase transitions in argon (Ar), “terrestrially the most abundant and industrially the most frequently used of the noble gases”¹ (since it is noble, the gas is mono-atomic). The phase diagram for Ar is given in Fig. 1.

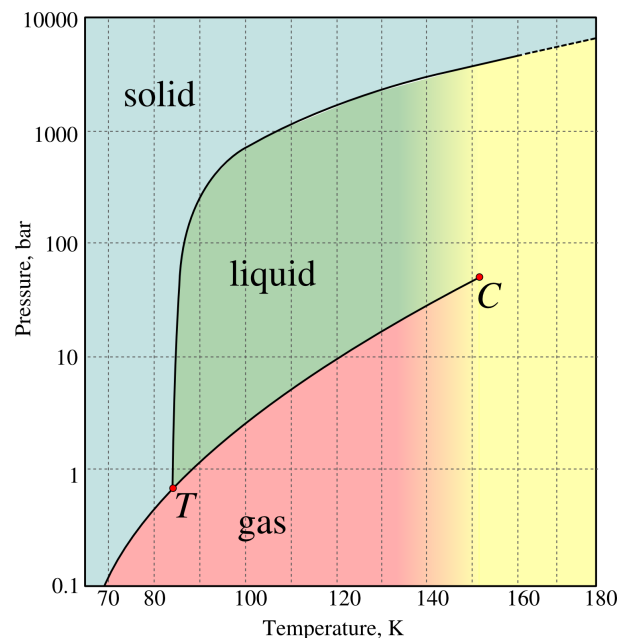


Figure 1: Phase diagram of Argon

To study the phase transformation, we will use 2D molecular dynamics simulations (see input data format in Appendix) for a constant volume $V = L \times L$ with periodic boundaries and study macroscopical quantities such as temperature, pressure and energies of the system of particles. The objective is to investigate this model system and verify if molecular dynamics simulation provides us with qualitatively reasonable results.

2 Provided files

To download all necessary files, go to www.yastrebov.fr/TPMD2018.zip.

1. [md.sim.inp](#): input file for MD simulations (see Appendix A for details).

¹www.britannica.com

2. [Make_initial_config.py](#): a draft script to construct the initial distribution of particles.
3. [plot_configuration.gpl](#): a gnuplot script to plot a particular particle configuration.
4. [anim.py](#), [aplot.gpl](#): files needed to make animation, run [anim.py](#) with an integer number n as an argument, so that only n -th data will be plotted. For example [anim.py 10](#)
5. Folder [CODE](#). To make the MD code working on your PC, copy this folder somewhere, enter the folder [CODE](#), compile the code [Zmake](#) and copy the library *.so into the directory where you plan to run your MD simulations.

3 Initial configuration. First simulations.

Argon data: $\sigma = 3.4 \text{ \AA}$, $\varepsilon = 0.0343 \text{ eV}$, $m = 39.95 \text{ u}$. Convert to SI.

- Write a small script to distribute N particles randomly in the simulation box (see data format in Appendix A). Example of a python script ([TP.Scripts/Make_initial_config.py](#)) is given. A reasonable number of particles would be $N < 1000$ for a square box of side $a \leq 10 \text{ nm}$. If you think it would be too difficult to code this routine within a reasonable time, please use directly `**configuration gas` (see Appendix A).
- Use `**configuration *file your_file` without temperature control $\gamma = 0$ to run the first simulation `Zrun md_sim.inp`.
- Check what happens with macroscopic physical quantities: kinetic and potential energy, pressure, temperature (these data are stored in file "energy.md").
- Find the equilibrium distance between particles for 0K. Based on this result ensure that initial distribution does not result in huge pressure and temperatures.
- If you succeeded to create your generation file then use it; if you did not then download www.yastrebov.fr/ready_script.py and work with it. Prescribe initial velocities of particles in any way (see data format in Appendix A). Check how the probability distribution of particle velocities varies with time at constant temperature.

4 How to analyze the simulation results

- The simulation writes two files [energy.md](#) and [frames.md](#), the first contains all integrated quantities (potential and kinetic energies, temperature, and pressure) the second contains instantaneous configurations of the system (particle coordinates and velocities).
- To plot data you can use gnuplot in the following way: run gnuplot by typing `gnuplot` in terminal, next execute `plot 'energy.md' using 1:4'` which will plot you column 4 (temperature) as a function of column 1 (time).
- To plot the configuration at increment $i \times N_b$, where N_b is the frequency of data output, in `gnuplot` execute `plot 'frames.md' u 2:3 every ::i::i with points pointtype 7 pointsize 1.` You can also use the proposed script [plot_configuration.gpl](#), to execute it run `./plot_configuration.gpl`, if it does not execute, make it executable by typing `chmod +x plot_configuration.gpl`.
- Equivalently, you can use matplotlib scripts to plot the results: `./plot.py` to plot the potential energy as a function of temperature and `./plot_configuration.py N` to plot configuration at increment N . Here too, if it does not execute, make it executable by typing `chmod +x plot.py` and `chmod +x plot_configuration.py`.
- To construct an animation execute `./anim.py`.

5 Computation

- Start the simulation using your generated file or “gas” option indicated in the input file.
- Control the temperature using `**set_temperature,`
`**temperature_adjustment_coefficient γ , **adjust_temperature_every`
- Equilibrate the system at $T = 300K$ and next cool it down. Follow changes in macroscopical quantities. Remember that if the number of particles is small, you need to consider time averages. Identify the phase transition.
- Verify what happens with the particle arrangement.

6 Questions for the evaluation

Right and complete answer for each question adds you one point, thus your note for this part will be $\in [0, 5]$. Written answers should be submitted by 16h30.

0. Your name²:

1. How do the particle velocities are distributed in your simulations? How should they be distributed according to the theory?

2. For the gas: which thermodynamic characteristics would change if for a given number of particles the size of the box is increased by a factor of two? Explain.

3. What does determine the cooling rate, which you used in your simulations? Illustrate.

²Sorry, for this question you do not get any points ☹.

4. How to determine the heat capacity of your system in gas state and solid state?

5. Does this heat capacity is measured under constant volume or constant pressure?

A Appendix. Input file structure

- ****configuration *file filename**
Name of the input file containing initial configuration of atoms (positions and velocities).
The format of this file is the following:

```

**particles
dim N
id1 x1 y1 vx1 vy1 color1
id2 x2 y2 vx2 vy2 color2
...

```

Where **dim** is the system dimension, **N** is the number of particles, **id** is the particle id (int type), **x, y** are its Cartesian coordinates (double type), **vx, vy** are its velocities (double type), **color** is a color associated with this particular particle (int type).

If the time seems to be too short for you to code a routine to properly distribute atoms, please use ****configuration gas** and define the appropriate ****box**.

- ****box_size (a,b)**
Size of the simulation box (vector type). For example: ****box_size (1e-9, 1e-9)**³ creates a simulation box of size $1nm \times 1nm$ with the left lower corner coordinates (0,0) and the right top coordinates (1e-9, 1e-9). Pay attention that coordinates of particles provided in ****configuration *file** are compatible with the simulation box.
- ****num_timesteps Nt**
Number of simulation time steps (int type).
- ****cut_off c**
Cut-off distance used in MD simulations, the double number is a multiplier c for the equilibrium distance computed for given parameters of Lennard-Jones 6-12 potential, so that $r_{\text{cutoff}} = c\sigma^{1/6}$. The standard number used in simulations $c = 2.5$.
- ****dim d**
Problem dimension, here $d = 2$.
- ****set_temperature t₀ T₀ t₁ T₁ ... t_n T_n**
A table which prescribes the needed temperature in the simulation. Time points t_i and associated temperature points T_i are provided. A linear interpolation is used in between, i.e.

$$\forall t \in [t_i, t_{i+1}]: T(t) = T_i + \frac{t - t_i}{t_{i+1} - t_i} (T_{i+1} - T_i)$$

Note that if simulation time $t < t_0$, then temperature T_0 is used, equivalently if $t > t_n$, then temperature T_n will be used. By construction the number of double numbers to be provided should be even.

Example: ****set_temperature 0. 300. 1.e-12 300. 10.e-12 100.**

- ****temperature_adjustment_coefficient γ** Provides the coefficient $0 \leq \gamma \leq 1$ used to scale the temperature using the following rule:

$$\beta = \sqrt{1 + \gamma \left(\frac{T_t}{T} - 1 \right)},$$

where β is the coefficient used to scale the velocity of particles, T_t is the target temperature, T is the current system temperature.

- ****adjust_temperature_every na**
Provides the number of integration time steps between consecutive temperature scaling, i.e. if $na = 100$, the temperature is scaled on every 100-th time step.

³Do not forget brackets.

- **dt dt**
Provides the integration time step. It should be big enough to go as fast as possible and small enough to be accurate and converge at all. If dt is chosen too high, an error message will appear.
- **integrator StormerVerlet**
Integration method: Velocity Störmer-Verlet is used.
- **md.output increment Nb TYPE**
Outputs energy (file “energy.md”) and particle configuration (file “frames.md”). Integer value Nb controls the frequency of output, i.e. $Nb = 100$ means that energy and configuration is saved every 100 time steps. $TYPE$ can be either `particle_coord_velocity` or `energy`, in the former case, both configuration and macroscopic quantities are saved, in the latter case only macroscopic quantities are saved. The data is saved in adimensional units:

$$x' = x/\tilde{x}, \quad v' = v/\tilde{v}, \quad E' = E/\tilde{E}, \quad t' = t/\tilde{t}$$

The normalization parameters for length \tilde{x} , time \tilde{t} , velocity \tilde{v} and energy \tilde{E} are provide in file “energy.md”.

The data format for the particle configuration is the following:

```
...
# time ti inc/every: inci
color1 x1 y1 vx1 vy1
color2 x2 y2 vx2 vy2
...
colorn xn yn vxn vyn
empty line
# time ti+1 inc/every: inci+1
color1 x1 y1 vx1 vy1
color2 x2 y2 vx2 vy2
...
colorn xn yn vxn vyn
empty line
...
```

- **potential LJ *powers 6. 12.**
Lennard-Jones 6-12 potential is used.
- **MD.material LJ 1**
 - *mass m
 - *epsilon ϵ
 - *sigma σ****return**
Input data to determine particle characteristics and parameters of the Lennard-Jones potential: mass m , distance σ and energy ϵ parameters are given in SI units.