Modeles et numerical methodes in mechanics and physico-chemistry

Practical work on Molecular Dynamics: Phase Transition

Master program DMS

February 2, 2016

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 monoatomic). The phase diagram for Ar is given in Fig. 1.

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 peridodic 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/TPMD.zip`

1. `md_sim.inp`: input file for MD simulations.
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 arguments "frames.md" and an integer number n, so that only n-th data will be plotted. For example `anim.py frames.py 10`

[1] www.britannica.com

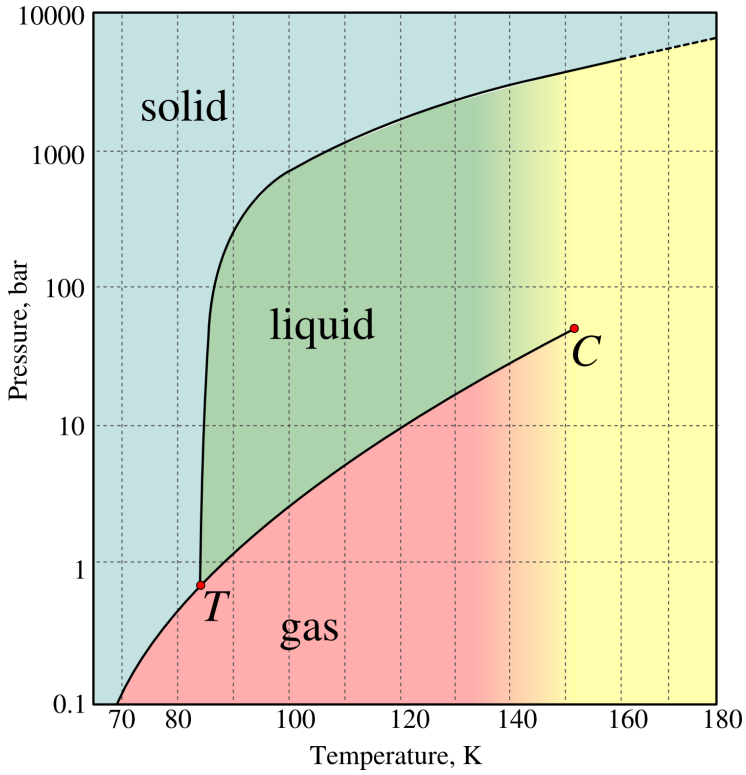

Figure 1: Phase diagram of Argon

5. Folder `CODE`. To make the MD code working on your PC, copy this folder somewhere, enter the folder, compile the code `Zmake` and copy the library *.so into the directory where you will run your MD simulations.

3 Initial configuration

Argon data: $\sigma = 3.4$Å, $\varepsilon = 0.0343eV$, $m = 39.95u$. Convert to SI.

- Write a small script to distribute N particles randomly in the simulation box (see data format in Appendix). Example of a python script is given. A reasonable number of particles would be $N < 1000$ for the square box of side $a \leq 10$ nm.
- Run the simulation using `**configuration *file your_file` without temperature control $\gamma = 0$
- Check what happens with macroscopic physical quantities: kinetic and potential energy, pressure, temperature (these data are plot 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.
- Implement in the same script Box-Muller algorithm to generate random velocities. Box-Muller algorithm consists in generating two random number $0 < a, b < 1$ from a uniform distribution, next compute $\alpha_x = \sqrt{-2\ln(a)}\cos(2\pi b$ and $\alpha_y = \sqrt{-2\ln(a)}\sin(2\pi b$, then random velocities can be assigned to particles as

$$v_x = \sqrt{k_B T/m}\alpha_x, \quad v_y = \sqrt{k_B T/m}\alpha_y.$$

  To test, construct probability density of resulting distribution for system of many particles.
- Run the simulation using `**configuration *file your_file` without temperature control $\gamma = 0$. Verify the consistency of the prescribed temperature.

4 Computation

- Start the simulation using your generated 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 than if the number of particles is small, you need to consider time averages. Find the phase transition.
- Verify what happends with the particle arrangement.

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`
  $\texttt{id}_1$ $\texttt{x}_1$ $\texttt{y}_1$ $\texttt{vx}_1$ $\texttt{vy}_1$ $\texttt{color}_1$
  $\texttt{id}_2$ $\texttt{x}_2$ $\texttt{y}_2$ $\texttt{vx}_2$ $\texttt{vy}_2$ $\texttt{color}_2$
  `...`
  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).
- `**box_size (a,b)`
  Size of the simulation box (vector type). For example: `**box_size (1e-9, 1e-9)`[2] 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 $r_{\text{cutoff}} = c\sigma 2^{1/6}$. The standard number used in simulations $c = 2.5$.
- `**dim d`
  Problem dimension, here $d = 2$.
- `**set_temperature` $\texttt{t}_0$ $\texttt{T}_0$ $\texttt{t}_1$ $\texttt{T}_1$ `...`$\texttt{t}_n$ $\texttt{T}_n$
  A table which prescribes the needed temperature in the simulation. Time points t_i and associated temperature points T_i are provided. A linear

[2] Do not forget brackets.

interpolation is used in between, i.e.

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

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 consequitive temperature scaling, i.e. if $na = 100$, the temperature is scaled on every 100-th time step.

- `**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 t_i inc/every:  inc_i
color_1 x_1 y_1 vx_1 vy_1
color_2 x_2 y_2 vx_2 vy_2
```

```
...
color_n x_n y_n vx_n vy_n
    empty line
# time t_{i+1} inc/every:  inc_{i+1}
color_1 x_1 y_1 vx_1 vy_1
color_2 x_2 y_2 vx_2 vy_2
...
color_n x_n y_n vx_n vy_n
    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.