Project01 — Fys4460 — 2018

Project 1: Introductory Molecular Dynamics Modeling

Through this project you will learn to use molecular dynamics to address the behavior of single atoms and molecules with particular emphasis on understanding the statistical properties of he system.

Atomic modeling basics. How do we determine the motion of a system of atoms? Here, we will use a classical approximation to describe the motion of a system of atoms. In this approximation, we use Newton's laws of motion and a well-chosen description of the forces between atoms to find the motion of the atoms. The forces are described by the potential energies of the atoms given their positions. The potential energy functions can, in principle, be determined from quantum mechanical calculations because the forces and the potential energies depend on the states of the electrons — it is the electrons that form the bonds between atoms and are the origin of the forces between atoms. However, in the classical approximation we parametrize this problem: We construct simplified models for the interactions that provide the most important features of the interactions.

Interatomic potentials. The behavior of a system will depend on the potential energy and how it depends on the positions of all the particles. In general, we assume that the potential energy, U, depends on the positions, $\mathbf{r}_i$, of all the particles:

$$U = U\left(\{\mathbf{r}_i\}\right) \ . \tag{0.1}$$

This is a simplification. In general, the potential energy may depend on additional internal states of the interactions between the atoms. For example, we may know that two atoms interact through a covalent bond, and may therefore prescribe this covalent bond as part of the potential energy: The energy would depend both on the positions of the particles and on what particles are connected by these covalent bonds. Here, we will consider two types of interactions: bonded interactions where the interaction types between atoms are prescribed and does not change throughout the simulation, and non-bonded interactions, where the potential energy only depends on the positions of the particles. Non-bonded interactions open for modeling more complicated interactions between atoms, including the formation and breaking of strong bonds between atoms.

We will start by assuming that the potential energy can be written as a sequence of interactions of higher and higher order, starting from pair-wise interactions, then moving on two three-particle-interactions, four-particle-interactions and so on. We can then write the potential energy as

$$U\left(\{\mathbf{r}_i\}\right) = \sum_{ij} U_{ij}\left(\mathbf{r}_i, \mathbf{r}_j\right) + \sum_{ijk} U_{ijk}\left(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k\right) + \dots \ . \tag{0.2}$$

Here, we will first address a model that only includes two-particle-interactions, then move on to address a model with three-particle-interactions, and finally address several models for the interactions between water molecules.

Lennard-Jones potential. One of the simplest models for atom-atom interactions is a representation of the interactions between nobel-gas atoms, such as between two Argon atoms. For the interaction between two noble-gas atoms we have two main contributions:

- There is an attractive force due to a dipole-dipole interaction. The potential for this interactions is proportional to $(1/r)^6$, where r is the distance between the atoms. This interaction is called the *van der Waals* interaction and we call the corresponding force the *van der Waals* force.
- There is a repulsive force which is a quantum mechanical effect due to the possibility of overlapping electron orbitals as the two atoms are pushed together. We use a power-law of the form $(1/r)^n$ to represent this interaction. It is common to choose $n = 12$, which gives a good approximation for the behavior of Argon.

The combined model is called the **Lennard-Jones potential**:

$$U(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) \ . \tag{0.3}$$

Here, ϵ is a characteristic energy, which is specific for the atoms we are modeling, and σ is a characteristic length.

The Lennard-Jones potential and the corresponding force $F(r)$ is illustrated in Fig. 0.1. We see that the Lennard-Jones potential reaches

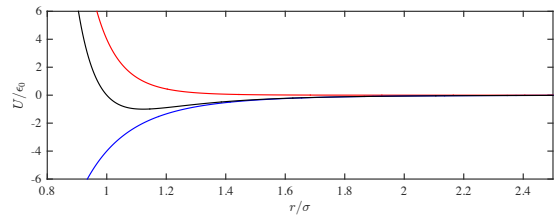

Fig. 0.1 Illustration of the Lennard-Jones potential.

its minimum when

$$F(r) = -\frac{d}{dr}U(r) = 0 \,, \tag{0.4}$$

which occurs at

$$F(r) = 24\,\epsilon_0 \left((\sigma/r)^6 - 2\,(\sigma/r)^{12} \right) = 0 \;\Rightarrow\; \frac{r}{\sigma} = 2^{1/6} \,. \tag{0.5}$$

We will use this potential to model the behavior of an Argon system. However, the Lennard-Jones potential is often used not only as a model for a noble gas, but as a fundamental model that reproduces behavior that is representative for systems with many particles. Indeed, Lennard-Jones models are often used as base building blocks in many interatomic potentials, such as for the interaction between water molecules and methane and many other systems where the interactions between molecules or between molecules and atoms is simplified (coarse grained) into a single, simple potential. Using the Lennard-Jones model you can model 10^2 to 10^6 atoms on your laptop and we can model 10^{10}-10^{11} atoms on large supercomputers.

Initial conditions. An atomic (molecular) dynamics simulation starts from an initial configuration of atoms and determines the trajectories of all the atoms. The initial condition for such a simulation consists of all the positions, $\mathbf{r}_i(t_0)$ and velocities $\mathbf{v}_i(t_0)$ at the initial time t_0. In order to model a realistic system, it is important to choose the initial configuration with some care. In particular, since most potentials such as the Lennard-Jones potential increase very rapidly as the interatomic distance r goes to zero, it is important not to place the atoms too close to each other. We therefore often place the atoms regularly in space, on a lattice, with initial random velocities.

We generate a lattice by first constructing a *unit cell* and then copying this unit cell to each position of a lattice to form a regular pattern of unit cells. (The unit cell may contain more than one atom). Here, we will use cubic unit cells. For a cubic unit cell of length b with only one atom

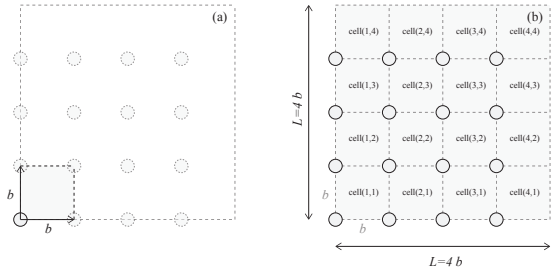

Fig. 0.2 **a** Illustration of a unit cell for a square lattice. **b** A system consisting of 4×4 unit cells, where each of the cells are marked and indexed for illustration.

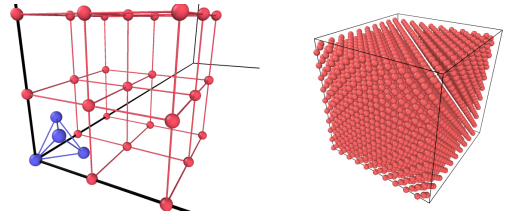

Fig. 0.3 *Left* Illustration of a unit cell for a face centered cubic lattice. Unit cell atoms illustrated in blue and the base position of other cells shown in red. *Right* A system consisting of $10 \times 10 \times 10$ unit cells.

in each unit cell, we can place the atom at $(0, 0, 0)$ in the unit cell and generate a cubic lattice with distances b between the atoms by using this cubic unit cell to build a lattice. This is illustrated for a two-dimensional system in Fig. 0.2. Such a lattice is called a *simple cubic lattice*.

However, for a Lennard-Jones system we know (from other theoretical, numerical and experimental studies) that the equilibrium crystal structure is not a simple cubic lattice, but a face centered cubic lattice. This is a cubic lattice, with additional atoms added on the center of each of the faces of the cubes. The unit cell for a face centered cubic lattice is illustrated in Fig. 0.3. We will use this as our basis for a simulation and then select a lattice size b so that we get a given density of atoms. The whole system will then consist of $L \times L \times L$ cells, each of size $b \times b \times b$ and with 4 atoms in each cell.

Boundary conditions. A typical molecular model of a liquid of Argon molecules is illustrated in Fig. 0.4a. In this case, we have illustrated a small system of approximately $10 \times 10 \times 10$ atom diameters in size. Below, you will learn how to set up and simulate such systems on your laptop. Unfortunately, you will not be able to model macroscopically large systems — neither on your laptop nor on the largest machine in the world. A liter of gas at room temperature typically contains about 10^{23} atoms, and this is simply beyond practical computational capabilities.

But it is possible with a small system to gain some insights into how very large, even infinite, systems behave? One of the problems with the $10 \times 10 \times 10$ system above is the external boundaries. But we can fool

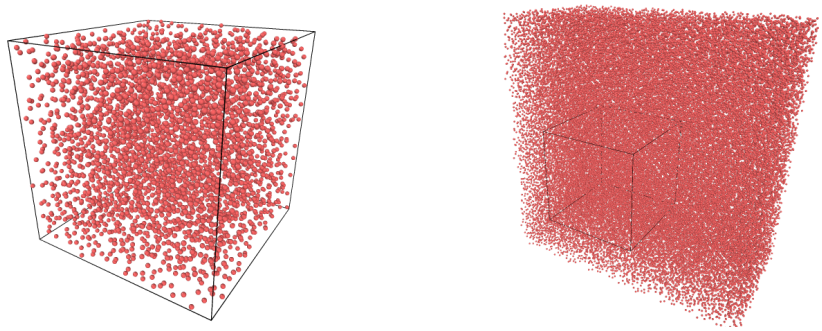

Fig. 0.4 **a** Visualization of a $10 \times 10 \times 10$ simulation of an Argon liquid. **b** Visualization of a 10×10 simulation of an Argon gas, showing the actual simulation area in the center box, and 8 copies of the center box illustrating the principles of the periodic boundary conditions.

the system into believing it is infinite by applying what we call periodic boundary conditions. If the atoms on the left of the system do not see emptiness to their left, but instead see the right hand side of the system, as if the system is wrapped around a cylinder, the system will look like it is inifinite. This is illustrated in Fig. 0.4b. This convention of periodic boundary conditions is usually applied in all simulations in order to avoid dealing with boundary conditions. (There may be some possibly problematic aspects of periodic boundaries, but we will not address or worry about these here).

Integrating the equations of motion. How do we determine the behavior of the system? We solve the equations of motion. For molecular dynamics simulations we usually use an algorithm called the Velocity-Verlet, which is approximately like the forward Euler method, but it is very well suited for conservative forces. The velocity is calculated at both time t and at intermediate times $t + \Delta t/2$, where Δt is the time-step, whereas the forces are only calculated at the full time-steps, t, $t + \Delta t$, $t + 2\Delta t$ etc. The most time-consuming part of the calculation is the calculation of the forces. We therefore want to limit the number of times we calculate the forces and still have as high precision as possible in our calculations. The Velocity Verlet algorithm ensures a good trade-off between precision in the integration algorithm and the number of times forces are calculated.

In the Velocity-Verlet algorithm the positions $\mathbf{r}_i(t)$ and velocities $\mathbf{v}_i(t)$ of all the atoms, $i = 1, \ldots, N$, are propagated forward in time according to the algorithm:

$$\mathbf{v}_i(t+\Delta t/2) = \mathbf{v}_(t) + \mathbf{F}_i(t)/m_i\Delta t/2 \tag{0.6}$$

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_(t) + \mathbf{v}_i(t+\Delta t/2) \tag{0.7}$$

$$\mathbf{F}_i(t+\Delta t) = -\nabla V(\mathbf{r}_i(t+\Delta t)) \tag{0.8}$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_(t+\Delta t/2) + \mathbf{F}_i(t+\Delta t)/m_i\Delta t/2\ ,, \tag{0.9}$$

This method has very good properties when it comes to energy conservation, and it does, of course, preserve momentum perfectly.

Non-dimensional equations of motion. However, all the quantities in a molecular dynamics simulations are very small. It is therefore usual to introduce measurement units that are adapted to the task. For the Lennard-Jones model we usually use the intrinsic length and energy scale of the model as the basic units of length and energy. This means that we measure lengths in units of σ and energies in units of ϵ_0. A vector $\mathbf{r}'_i$ in the simulation is therefore related to the real-world length $\mathbf{r}_i$ through

$$\mathbf{r}_i = \sigma\,\mathbf{r}'_i \ \Leftrightarrow\ \mathbf{r}'_i = \mathbf{r}_i/\sigma\ . \tag{0.10}$$

Similarly, we can introduce a Lennard-Jones time, $\tau = \sigma\sqrt{m/\epsilon}$, where m is the mass of the atoms, and the Lennard-Jones temperature $T_0 = \epsilon/k_B$. Using these notations, we can rewrite the equations of motion for the Lennard-Jones system using the non-dimensional position and time, $\mathbf{r}'_i = \mathbf{r}_i/\sigma$ and $t' = t/\tau$:

$$m\frac{d^2}{dt^2}\mathbf{r}_i = \sum_j 24\epsilon_0\left((\sigma/r_{ij})^6 - 2\,(\sigma/r_{ij})^{12}\right)\left(\mathbf{r}_{ij}/r_{ij}^2\right)\ , \tag{0.11}$$

to become

$$\frac{d^2}{d(t')^2}\mathbf{r}'_i = \sum_j 24\left(r_{ij}^{-6} - 2r_{ij}^{-12}\right)\left(\mathbf{r}'_{ij}/r_{ij}'^2\right)\ . \tag{0.12}$$

Notice that this equation is general. All the specifics of the system is now part of the characteristic length, time and energy scales σ, τ, and ϵ_0. For Argon $\sigma = 0.3405\mu$m, and $\epsilon = 1.0318\cdot 10^{-2}$eV, and for other atoms you need to find the corresponding parameter values.

Quantity	Equation	Conversion factor	Value for Argon
Length	$x' = x/L_0$	$L_0 = \sigma$	0.3405μm
Time	$t' = t/tau$	$\tau = \sigma\sqrt{m/\epsilon}$	$2.1569\cdot 10^3$ fs
Force	$F' = F/F_0$	$F_0 = m\sigma/\tau^2 = \epsilon/\sigma$	$3.0303\cdot 10^{-1}$
Energy	$E' = E/E_0$	$E_0 = \epsilon$	$1.0318\cdot 10^{-2}$ eV
Temperature	$T' = T/T_0$	$T_0 = \epsilon/k_B$	119.74 K

Running a molecular dynamics simulation. There are several efficient packages that solves the equations of motion for a molecular dynamics simulation. The packages allow us to model a wide variety of systems, atoms and molecules, and are efficienty implemented on various computing platforms, making use of modern hardware on your laptop or desktop or state-of-the-art supercomputing fascilities. We use a particular tool developed at Sandia National Laboratories called LAMMPS.

Installation of LAMMPS. If you want to be able to reproduce the simulations performed here you will need to install LAMMPS on your computer. This is very simple if you have a Mac or an Ubuntu system — for a windows system you will have to follow the installation instructions found at the web-site for LAMMPS. You can find all the documentation of LAMMS here.

Mac installation. On a mac you should be able to install LAMMS using Homebrew or Macports.

Using **Homebrew** (Homebrew) LAMMPS is installed with:

```
brew tap homebrew/science
brew install lammps
```

If you want to use the parallell implementation of LAMMPS you can install this version using

```
brew tap homebrew/science
brew install lammps --with-mpi
```

Using **MacPorts** (MacPorts) LAMMPS is installed with:

```
port install lammps
```

Ubuntu installation. You can install a recent version of LAMMPS with:

```
sudo apt-get install lammps
```

Starting a simluation in LAMMPS. If you have successfully installed LAMMPS, you are ready to start your first molecular dynamics simulations. The LAMMPS simulator reads its instructions on how to run a simulation from an input file with a specific syntax. Here, we will set up a three-dimensional simulation of a Lennard-Jones system using the file `in.myfirstmd`:

```
# 3d Lennard-Jones gas
units lj
dimension 3
boundary p p p
atom_style atomic

lattice fcc 0.01
region simbox block 0 10 0 10 0 10
create_box 1 simbox
create_atoms 1 box

mass 1 1.0
velocity all create 2.5 87287

pair_style lj/cut 3.0
pair_coeff 1 1 1.0 1.0 3.0

fix 1 all nve

dump 1 all custom 10 dump.lammpstrj id type x y z vx vy vz
thermo 100
run 5000
```

The simulation is run from the command line in a Terminal. Notice that the file `in.myfirstmd` must be in your current directory. I suggest creating a new directory for each simulation, copying the file `in.myfirstmd` into the directory and modifying the file to set up your simulation, before starting the simulation with:

```
lammps < in.myfirstmd
```

This simulation should only take a few seconds. It produces output in the Terminal and two files: `dump.lammpstrj`, which contains all the data from the simulation, and `log.lammps`, which contains a copy of what was output to your terminal. Fig. 0.5 illustrates the positions of the atoms initially and at the end of the simulation.

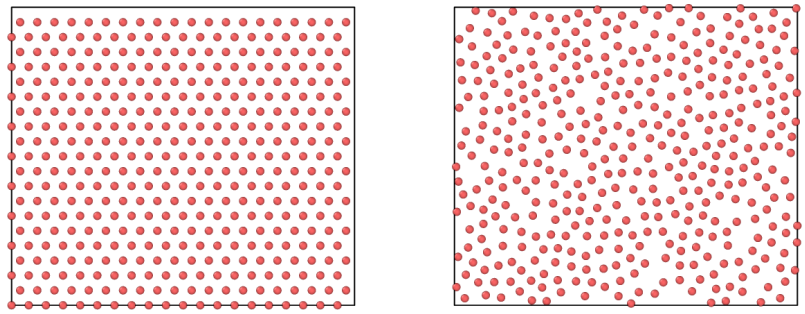

Fig. 0.5 (*Left*) Illustration of the initial hexagonal lattice generated in the simulation. (*Right*) Illustration of the final positions of the atoms after 5000 timesteps.

The input file `in.myfirstmd` consists of a series of commands to be interpreted by LAMMPS. Here, we look at what these do in detail. (You can skip this at first reading, and return when you wonder what the parameters actually do).

```
# 3d Lennard-Jones gas
```

This line starts with a `#` and is a comment that is ignored by the program.

```
units lj
dimension 3
boundary p p p
atom_style atomic
```

This block describes general features of the simulation:

The `units lj` command selects Lennard-Jones units, which were introduced above. This means that lengths are measured in units of σ, energies in units of ϵ_0, time in units of $\tau = \sigma\sqrt{m/\epsilon}$, and temperature in terms of $T_0 = \epsilon/k_B$. For Argon $\sigma = 0.3405\mu$m, and $\epsilon = 1.0318 \cdot 10^{-2}$eV. Other atomic models will have other parameters.

The `dimension` command specifies the dimensionality of the simulation: 2 or 3. Here we run a 3d simulation.

The `boundary` command specifies boundary conditions to be applied. Here we have periodic boundaries in the x-, y-, and z- directions.

The `atom_style` command specifies the complexity of the description of each atom/particle. Here, we will use the simplest description, `atomic`, which is used for noble gases and coarse-grained simulation models.

```
lattice fcc 0.01
region simbox block 0 10 0 10 0 10
create_box 1 simbox
create_atoms 1 box
```

This block sets up the dimensions of the $10 \times 10 \times 10$ simulation box and fills the box with atoms with a given packing fraction.

The `lattice` command generates a lattice of points. This does, surprisingly enough, not actually generate any atoms, it only generates a set of positions in space where atoms will be generated when we generate atoms. The type `fcc` specifies a three-dimensional lattice of face-centered-cubic shape. And the number `0.01` is called the

scale and is the reduced density, ρ', when we have chosen LJ units for the simulation. (Notice that the scale is interpreted differently if we do not use LJ units, see the LAMMPS documentation for more information).

The `region` command defines a region which is a `block` extending over $0 < x < 10$, $0 < y < 10$, $0 < z < 10$ We give this region the name `simbox`.

The `create_box` command now actually creates the simulation box based on the spatial region we called `simbox`. The simulation box will only contain 1 (one) type of atoms, hence the number `1`.

The `create_atoms` finally fills the simulation box we have defined using the lattice we have defined with atoms of type `1`.

```
mass 1 1.0
velocity all create 2.5 87287
```

This block defines the material properties of the atoms and defines their initial velocities.

The `mass` command defines that atoms of type `1` will have a mass of `1.0` relative to the mass of the Lennard-Jones model. This means that all atoms have mass 1 in the Lennard-Jones units. This means that the masses of all the atoms are the same as the mass m used in the non-dimensionalization of the Lennard-Jones model.

The `velocity` command generates random velocities (using a Gaussian distribution) so that the initial temperature for `all` atom types in the system is `2.5` in the dimensionless Lennard-Jones units. The last, strange integer number `87287` is the seed for the random number generator used to generate the random numbers. As long as you do not change the seed number you will always generate same initial distribution of velocities for this simulation.

```
pair_style lj/cut 3.0
pair_coeff 1 1 1.0 1.0 3.0
```

This block specifies the potential between the atoms.

The `pair_style` command specifies the we want to use a Lennard-Jones potential with a cut-off that is of length `3.0`. What does this mean? It means that since the Lennard-Jones potential falls so quickly to zero as the distance between the atoms increase, we will approximate interaction to be zero when the atoms are further than 3.0 away from each other (measured in Lennard-Jones units, that is

in units of σ). The simulator ensures that both the potential and the force is continuous across the transition. There are many other types of force fields that you may use — take a look at the documentation of LAMMPS for ideas and examples.

The `pair_coeff` command specifies the parameters of the Lennard-Jones model. The two first numbers, `1 1`, specifies that we describe the interaction of atoms of type 1 with atoms of type 1. And the parameters of the Lennard-Jones model are `1.0 1.0 3.0`. This means that The interaction between an atom of type 1 with an atom of type 1 has a σ-value corresponding 1.0 times the the general σ-value (hence the first number 1.0), and a ϵ_0-value corresponding to 1.0 times the overall ϵ-value (hence the second number 1.0). The cut-off for this interaction is 3.0 — the same value as we specified above.

```
fix 1 all nve
```

This one-line block specifies what type of simulation we are performing on the atoms. This is done by one or more `fix` commands that can be applied to regions of atoms. Here, the `fix`, which we call `1` (you can choose numbers or names for identity), is applied to `all` particles and specifies that the simulation is run at constant `nve`, that is, at constant number of particles (`n`), constant volume (`v`, meaning that the simulation box does not change during the simulation), and constant energy (`e`). You may be surprised by the constant energy part. Does the integration algorithm ensure that the energy is constant. Yes, it does. However, there can be cases where we want to add energy to a particular part of the system, and in that case the basic interation algorithm still conserves energy, but we add additional terms that may change the total energy of the system.

```
dump 1 all custom 10 dump.lammpstrj id type x y z vx vy vz
thermo 100
run 5000
```

This block specifies simulation control, inclusing output and the number of time-steps to simulate.

The `dump` command tells the simulator to output the state. The `1` is the name we give this dump — it could also have been given a name such as `mydump`. We specify that `all` atoms are to be output

using a `custom` output format, with output every `10` time-steps to the file `dump.lammpstrj`, and the 'id type x y z vx vy vz' list specifies what is output per atom.

The `thermo` command specifies that output to the Terminal and to the log file, `log.lammps`, is provided every `100` timesteps.

The `run` command starts the simulation and specifies that it will run for `5000` timesteps.

Visualizing the results of a simulation. It is good practice to look at the results of the simulation. Use for example Ovito to visualize the results.

Macroscopic observables. We can use the MD simulation to measure macroscopic quantities by averaging properties of the simulation system. The ergodic hypothesis states that the time a system has one particular value of an observable A is proportional to the phase space volume where A has this value. This applies to systems in equilibrium studied for a long period of time. As a result, the time average and ensemble average of a variable are equal. If we average over long enough periods of time, and our microscopic interaction model is correct, we can use the models to predict equilibrium properties of real systems. In addition, we can use the simulation to gain insight into the statistical properties of systems with many particles, and in particular into the fluctuations in such systems.

a) According to the central limit theorem, the velocity distribution of the particles will eventually evolve into a Maxwell-Boltzmann distribution independent of the initial conditions. Let us test this by starting the simulation with velocities that are uniformly distributed random numbers in the interval $[-v, v]$, for your own choice of v (You will actually provide the value of T and not v). Look at the description of the ´velocity´ command in Lammps to see how to specify a uniform distribution. Investigate the probability density for the velocities and for the speeds of the atoms at a given time by writing the velocities to a file and visualizing the results in a histogram. Study the time-development of the velocity distribution to estimate how long time it takes for the velocities to reach a Maxwell-Boltzmann distribution. Hint: You can find the histogram $h_i(t)$ for a given set of bins of velocities, v_i. You can characterize the time development of $h_i(t)$ by looking at the normalized inner product, $\sum_i h_i(t)h_i(t_n)/\sum_i h_i(t_n)h_i(t_n)$, where t_n is the final time-step in your simulation.

We are modelling a micro-canonical ensamble, since we study systems with constant volume and a constant number of particles, and, in theory, constant energy. In practice, though, our integration scheme does not conserve energy exactly.

b) Find the total energy (kinetic and potential) of the system, $E(t)$, and plot it as a function of time. How does the size of the fluctuations in energy depend on the time-step Δt? (Hint: Use the ´timestep´ command in Lammps to change the timestep in the simulation). Check a few values of Δt to ensure your choice provides reasonable results.

In general, it is not trivial to calculate the temperature for general potential forms. The simplest estimate assumes equilibrium between the translational and potential degrees of freedom. According to the equipartition principle, the average total kinetic energy is

$$\langle E_k \rangle = \frac{3}{2} N k_{\mathrm{B}} T \tag{0.13}$$

where N is the number of atoms and T is our estimate for the system temperature.

c) Use this method to measure the temperature as a function of time, $T(t)$. (Don't forget to equilibriate the system first). Find the average temperature of the system (after equilibration), and compare with the temperature you used to generate the initial velocity distribution. Characterize the fluctuations in temperature, and find how the fluctuations vary with system size.

There are several ways of measuring the pressure P of a many-atom system. The method we will use, and which is implemented in Lammps, is derived from the virial equation for the pressure. In a volume V with particle density $\rho = N/V$, the average pressure is

$$P = \rho k_{\mathrm{B}} T + \frac{1}{3V} \sum_{i<j} \mathbf{F}_{ij} \cdot \mathbf{r}_{ij} \ , \tag{0.14}$$

where the sum runs over all interacting particle pairs. Note that this expression depends on the ensemble – and is valid for the micro-canonical ensemble only. The vector products should be computed and summed up inside the force loops for efficiency.

d) Measure the pressure as a function of temperature in your system, plot $P(T)$, and discuss the result. How do they compare with your theoretical expectations?

e) Measure the pressure as a function of both temperature and density, visualize and discuss the results in comparison with relevant theoretical models. Can you predict the parameters in the model using these simulations?

The diffusion constant. We want to characterize transport in a fluid by measuring the self-diffusion of an atom: We give an atom i a label, and measure its position as a function of time, $\mathbf{r}_i(t)$. We find the diffusion constant from the mean square displacement of all atoms (we trace the motion of every atom):

$$\langle r^2(t) \rangle = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{r}(t) - \mathbf{r}_{\text{initial}}) \ . \tag{0.15}$$

From theoretical considerations of the diffusion process we can relate the diffusion constant in the liquid to the mean squear displacement through:

$$\langle r^2(t) \rangle = 6Dt \text{ when } t \to \infty \ . \tag{0.16}$$

This result is similar to the behavior of a random walker in three dimensions, which is indeed a good approximation to the motion of an atom in a fluid.

f) Plot the mean square displacement as a function of time and estimate the diffusion constant, D. Investigate the effect of temperature by finding $D(t)$ for some temperatures of your own choice in the liquid phase of Argon. Remember that you are measuring the total distance travelled by the atoms, which must be continous when an atom is displaced though the periodic boundaries!

Microscopic structure – radial distribution functions. The radial distribution function $g(r)$, also called a pair correlation function, is a tool for characterizing the microscopic structure of a fluid. It is interpreted as the radial probability for finding another atom a distance r from an arbitrary atom, or equivalently, the atomic density in a spherical shell of radius r around an atom. It is commonly normalized by dividing it with the average particle density so that $\lim_{r\to\infty} g(r) = 1$.

g) Estimate $g(r)$ for $r \in (0, \frac{L}{2}]$ in your Argon system. The easiest way is to divide the distance interval into bins, loop over all pairs of particles and count how many distances belong in each bin. Time-averaging the function gives a better description of the system's general behaviour. Plot $g(r)$ for temperatures where the system is in solid and liquid phases.

Does it appear as expected? How would the exact $g(r)$ look for a perfect crystal?

Thermostats. In order to simulate the canonical ensemble, interactions with an external heat bath must be taken into account. Many methods have been suggested in order to achieve this, all with their pros and cons. Requirements for a good thermostat are:

1. Keeping the system temperature around the heat bath temperature
2. Sampling the phase space corresponding to the canonical ensemble
3. Tunability
4. Preservation of dynamics

The method closest to fullfilling these requirements which is in widespread use is the Nosé-Hoover thermostat, which is somewhat complicated to implement, but is the standard in Lammps.

Some thermostats work by rescaling the velocities of all atoms by multiplying them with a factor γ. The Berendsen thermostat uses

$$\gamma = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T_{\text{bath}}}{T} - 1\right)}\,. \tag{0.17}$$

with τ as the relaxation time, tuning the coupling to the heat bath. Though it satisfies Fourier's law of heat transfer (the transfered heat between two bodies is proportional to their temperature difference) it does a poor job at sampling the canonical ensemble.

h) Discuss possible challenges with the Berendsen thermostat.

The Andersen thermostat simulates (hard) collisions between atoms inside the system and in the heat bath. Atoms which collide will gain a new normally distributed velocity with standard deviation $\sqrt{k_{\text{B}}T_{\text{bath}}/m}$. For all atoms, a random uniformly distributed number in the interval $[0, 1]$ is generated. If this number is less than $\frac{\Delta t}{\tau}$, the atom is assigned a new velocity. In this case, τ is treated as a collision time, and should have about the same value as the τ in the Berendsen thermostat. The Andersen thermostat is very useful when equilibrating systems, but disturbs the dynamics of e.g. lattice vibrations.

i) Use the Andersen thermostat, and compare $T(t)$ graphs for simulations using the Anderson and he Noose-Hoover thermostats. Be aware that our T is just an approximation to the real temperature. Describe the impact of the thermostat on the observed dynamics of the system of atoms – as seen in your visualizations of the dynamics.

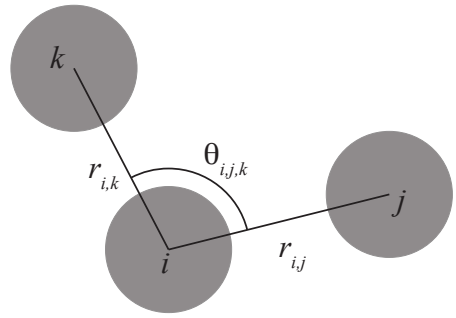

Fig. 0.6 Illustration of a thee-particle configuration.

Three-particle-potentials. Two-particle potential can be as simple as the Lennard-Jones potential, or include other terms that include other physical effects. However, there are fundamental challenges with a two-particle potential. For example, if we were to model a water molecule, the angle between the two hydrogen atoms would be free. In a two-particle potential, the directions of the bonds between atoms are not specificed.

The effect of covalent bonds can only be induced using a three-particle potential. Three-particle potentials depend on the relative positions of three atoms, such as atom i and its two neighbors, j and k.

The potential can be *bonded* or *unbonded*. For a *bonded* potential, we prescribe which atoms are connected by three-particle interations. For example, we can introduce a potential term that depends on the angle, θ_{ijk} between i and j and i and k, as illustrated in Fig. 0.6

For example, we may prescribe that atom i is connected to j and k with an energy that depends on the angle θ_{ijk}. The corresponding term in the potential energy would then be

$$V_{ijk} = \epsilon_{ijk} \left(\cos\left(\theta_{ijk}\right) - \cos\left(\theta_{ijk,0}\right)\right)^2 \ , \tag{0.18}$$

where ϵ_{ijk} is the energy for this interaction and $\theta_{ijk,0}$ is the equiliubrium angle between the three atoms. This interaction will always act between these three atoms. The interaction is therefore not only dependent on the positions of the atoms, but also on the *state* — a description of what atoms are interacting with each other. Typically, this will correspond to the atoms that are initially close together and therefore forming bonds.

However, we could also introduce an energy that does not depend on the state, but only the positions of the particles. Thus, the angular dependency would only be included if the atoms are close together, and would disappear as the atoms are moved further apart. This can be achieved by including a cut-off function that depends on the distances

r_{ij} and r_{ik}. Typically, we will introduce a cut-off for r_{ij} and a cut-off for r_{ik} and multiply the two.

An example of a potential with both two-particle and three-particle interactions is the Stillinger-Weber potential. In general, this potential is described by the following potential energy function:

$$E = \sum_i \sum_j V_2(r_{ij}) + \tag{0.19}$$

$$\sum_i \sum_j \sum_k V_3(r_{ij}, r_{ik}, \theta_{ijk}) \ , \tag{0.20}$$

$$V_2(r_{ij}) = A_{ij}\epsilon_{ij}\left[B_{ij}\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{p_{ij}} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{q_{ij}}\right] \tag{0.21}$$

$$\exp\left(\frac{\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right) \ , \tag{0.22}$$

$$V_3(r_{ij}, r_{ik}, \theta_{ijk}) = \lambda_{ijk}\epsilon_{ijk}\left[\cos\theta_{ijk} - \cos\theta_{ijk,0}\right]^2 \tag{0.23}$$

$$\exp\left(\frac{\gamma_{ij}\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right)\exp\left(\frac{\gamma_{ik}\sigma_{ik}}{r_{ij} - a_{ik}\sigma_{ik}}\right) \tag{0.24}$$

First, we notice that all the parameters may depend on the type of atom. For a case where all the atoms are the same, we expect all of these constants to be the same.

We notice that the two-particle part is similar to the Lennard-Jones form we studied previously, but with exponents p and q that can be varied. In addition, a cut-off function has been introduced. This cut-off is a function, $f(u)$ of the difference $u = r_{ij}/\sigma_{ij} - a_{ij}$:

$$f(u) = \exp(1/u) \ . \tag{0.25}$$

Similar cut-off functions are introduced for the angular dependence.

The Stillinger-Weber potential is a good model for silicon, Si, and can also be a good approximation for structures that involve Si. However, we would recommend the Vashishta potential for SiC or SiO_2.

Implementation in Lammps. The Stillinger-Weber potential is well integrated in Lammps as a `pair_style` called `sw`. The following Lammps script sets up a simulation of a Si system based on a diamond lattice structure. In this case we use the `metallic` units.

```
# run this on multiple partitions as
#   mpirun -np 3 lmp_g++ -partition 3x1 -in in.tad
```

```
units           metal

atom_style      atomic
atom_modify     map array
boundary        p p p
atom_modify        sort 0 0.0

# temperature
variable myTemp equal 1200.0

# diamond unit cell
variable myL equal 4
variable myscale equal 1.3

variable a equal 5.431*${myscale}
lattice         custom $a               &
                a1 1.0 0.0 0.0          &
                a2 0.0 1.0 0.0          &
                a3 0.0 0.0 1.0          &
                basis 0.0 0.0 0.0       &
                basis 0.0 0.5 0.5       &
                basis 0.5 0.0 0.5       &
                basis 0.5 0.5 0.0       &
                basis 0.25 0.25 0.25    &
                basis 0.25 0.75 0.75    &
                basis 0.75 0.25 0.75    &
                basis 0.75 0.75 0.25

region          myreg block     0 ${myL} &
                                0 ${myL} &
                                0 ${myL}
create_box      1 myreg
create_atoms    1 region myreg

mass            1       28.06

group Si type 1

velocity all create ${myTemp} 5287286 mom yes rot yes dist gaussian

pair_style      sw
pair_coeff * * Si.sw Si

neighbor        1.0 bin
neigh_modify    every 1 delay 10 check yes

timestep        1.0e-3
fix             1 all nve

# Run simulation
thermo          10
dump 1 all custom 10 dump.lammpstrj id type x y z vx vy vz
run             1000
```

We also need to specify a file that contains the parameters for all the atom types in the simulation. Here, this file is `Si.sw`:

```
# DATE: 2007-06-11 CONTRIBUTOR: Aidan Thompson, athomps@sandia.gov
# CITATION: Stillinger and Weber,  Phys Rev B, 31, 5262, (1985)
# Stillinger-Weber parameters for various elements and mixtures
# multiple entries can be added to this file,
```

```
# LAMMPS reads the ones it needs
# these entries are in LAMMPS "metal" units:
#   epsilon = eV; sigma = Angstroms
#   other quantities are unitless

# format of a single entry (one or more lines):
#   element 1, element 2, element 3,
#   epsilon, sigma, a, lambda, gamma, costheta0, A, B, p, q, tol

# Here are the original parameters in metal units, for Silicon from:
#
# Stillinger and Weber,  Phys. Rev. B, v. 31, p. 5262, (1985)
#

Si Si Si 2.1683  2.0951  1.80  21.0  1.20  -0.333333333333
         7.049556277  0.6022245584  4.0  0.0 0.0
```

j) Based on the `Si.sw` file, how does the terms in the Stillinger-Weber potential compare with the terms in the Lennard-Jones potential?

k) Run and visualize simulations of the Si system in solid, liquid and gas states.

l) Calculate the diffusion constant $D(T)$ for the Si system. Use the plot to determine the melting point of the Si system. How does your result match experimental values?

Water models. The behavior of water is important and complex. Important because of its role in biological and geological processes and chemical processing. Complex because of the many non-trivial details in the behavior of water: the equations of state, the structure of water and ice, the effect of weak hydrogen bonding, critical points and phase transitions, and heat capacity. Many models have been developed with focus on different physical effects with a large variation in complexity. Simple models capture some effects and allow large systems to be modelled, whereas more complex models capture many effects, but are limited to small systems.

Various aspects of water models are illustrated in Fig. 0.7. The models typically include both electrostatic effects from point charges and Lennard-Jones effects. The Lennard-Jones effects ensure that there are repulsive interactions at short distances so that the structures do not collapse under electrostatic interactions. The electrostatic interactions introduce a component that leads to alignment and directional interactions. Typically, the models have been developed to ensure a good fit to one or a few physical structures or parameters, such as the radial distribution function or critical parameters. However, it should be noted that good correspondence for the radial distribution function when compared with

experimental data is not a sensitive test, but mainly ensures a tetrogonal structure.

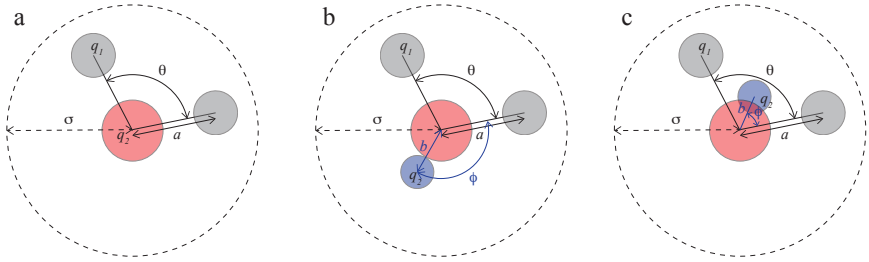

Fig. 0.7 Illustration of different types of planar water models. **a** corresponds to SPC SPC/E and flexible SPC, and **c** corresponds to TIP4P models.

SPC and SPC/E. The simplest water models are rigid models, where the relative positions of the atoms in the water molecule is constant. The interactions between molecules are in the form of Lennard-Jones and electrostatic forces, with a potential on the form:

$$V_{ij} = \sum_i \sum_j \frac{k_c q_i q_j}{r_{ij}} + 4\epsilon \left(\left(\frac{\sigma}{r_{OO,ij}} \right)^{12} - \left(\frac{\sigma}{r_{OO,ij}} \right)^{6} \right) , \qquad (0.26)$$

where the r_{ij} distances are between point charges in the two molecules and $r_{OO,ij}$ are between the oxygen atoms in the two molecules.

Many models use the actual angles θ between the hydrogen atoms, but the SPC model uses a slightly higher angle, as illustrated in the following table:

Model	σ (A)	ϵ (kJ mol^{-1})	a	b	q_1	q_2	θ	ϕ
SPC	3.166	0.650	1.0	-	+0.410	-0.820	109.47	-
SPC/E	3.166	0.650	1.0	-	+0.4238	-0.8476	109.47	-
TIP4P	3.15365	0.6480	0.9572	0.15	+0.52	-1.04	104.52	52.26
TIP4P/Ice	3.1668	0.8822	0.9572	0.1577	+0.5897	-1.1794	104.52	52.26

The Single Point Charge (SPC) model is illustrated as **a** in Fig. 0.7 is in a single plane.The Single Point Charge (SPC) models in its simplest form include Lennard-Jones interactions and the effect of single point charges placed at the positions of the oxygen and hydrogen atoms. Typical values for the charges and the distances between the atoms are shown in the following table.

The SPC/E model is a modified version of the SPC model that also includes an additional average polarization correction in the energy. The TIP3P model is also a 3-site model, which is slightly modified in the CHARMM force field.

Notice that the parameters may be slightly changed in various force field applications. In this case, you should use the modified parameters since these have been optimized for the types of interactions that are included.

Flexible SPE. The flexible SPC model is a bonded model where the behavior of the O-H and H-O-H bonds are explicitely included and modeled. The O-H stretching can also be made anharmonic, such as in the model by Toukan and Rahman, which provides a better description of the dynamical behavior of the system. This model is considered one of the most accurate three-site water models without polarization effects.

TIP4P. The TIP4P model is an example of an in-plane four-site model. This model is frequently used in many computational chemistry and biomolecular systems. There have been many reparameterizations of the TIP4P model for various special situations, for example the TIP4P/Ice model which is particularly suited for modeling ice systems.

ClayFF.

Implementing water models in Lammps. Here, we will address water systems that are modeled using the SPC/E model. In this case, the water molecules will move as a rigid body. This requires the `fix shake` command in Lammps, which is used to hold the O-H bonds and the H-O-H angles constant.

We will use the `moltemplate` program to set up water simulations. This is an effective tool for setting up and manipulating more complicated structures. In particular, it allows us to build structures from a single unit. Moltemplate includes not only coordinates, topology and force-field settings, but can also include additional information such as the `shake` constraints needed to model rigid molecules.

We will do this by following the example introduced in the moltemplate manual that sets up a box of SPCE water and runs a simulation on this system.

Defining a single water molecule. The following program introduces the SPCE model in moltemplate. We can then use this to define a box full of SPCE water molecules. The file `spce-simple.lt` is a moltemplate file that defines a single instance of the SPCE molecule.

```
# (NOTE: Text following # characters are comments)
#
# file "spce_simple.lt"
#
#   H1   H2
#     \ /
#      O
#
SPCE {
        # LAMMPS supports a large number of force-field styles. We must select
        # which ones we need. This information belongs in the "In Init" section.
        write_once("In Init") {
                units real                              # angstroms, kCal/mole, Dalto
                atom_style full                         # select column format for At
                pair_style lj/cut/coul/long 10.35       # params needed: epsilon sigm
                bond_style harmonic                     # parameters needed: k_bond,
                angle_style harmonic                    # parameters needed: k_theta,
                kspace_style ewald 0.0001               # long-range electrostatics s
                pair_modify tail yes
        }
        ## Atom properties and molecular topology go in the various "Data ..." sectio
        # We selected "atom_style full". That means we use this column format:
        # atomID molID atomType charge coordX coordY coordZ
        write("Data Atoms") {
                $atom:O $mol:. @atom:O -0.8476 0.0000000 0.000000 0.00000
                $atom:H1 $mol:. @atom:H 0.4238 0.8164904 0.5773590 0.00000
                $atom:H2 $mol:. @atom:H 0.4238 -0.8164904 0.5773590 0.00000
        }
        # All 3 atoms share same molID number which is unique for each water molecule
        # The "O" & "H1","H2" atoms in ALL molecules share same atom types: "O" & "H"
        write_once("Data Masses") {
                # atomType mass
                @atom:O 15.9994
                @atom:H 1.008
        }
        write("Data Bonds") {
                # bondID bondType atomID1 atomID2
                $bond:OH1 @bond:OH $atom:O $atom:H1
                $bond:OH2 @bond:OH $atom:O $atom:H2
        }
        write("Data Angles") {
                # angleID angleType atomID1 atomID2 atomID3
                $angle:HOH @angle:HOH $atom:H1 $atom:O $atom:H2
        }
        # --- Force-field parameters go in the "In Settings" section: ---
        write_once("In Settings") {
                # -- Non-bonded (Pair) interactions --
                # atomType1 atomType2 parameter-list (epsilon, sigma)

                pair_coeff @atom:O @atom:O 0.1553 3.5532
                pair_coeff @atom:H @atom:H 0.0 2.058

                # (mixing rules determine interactions between types @atom:O and @ato
                # -- Bonded interactions --
                # bondType parameter list (k_bond, r0)

                bond_coeff @bond:OH 554.1349 1.0

                # angleType parameter-list (k_theta, theta0)

                angle_coeff @angle:HOH 45.7696 109.47
```

```
                # Group definitions and constraints can also go in the "In Settings" sectio

                group spce type @atom:O @atom:H

                #fix fSHAKE spce shake 0.0001 10 100 b @bond:OH a @angle:HOH
                # (lammps quirk: Remember to "unfix fSHAKE" during minimization.)
        }
} # SPCE
```

However, to use this definition, we need to use moltemplate commands to generate water molecules. We write these commands in a script, `spce-water-system.lt`, which we then subsequently run to generate a set of files that are used when we run Lammps.

Let us first look at how we generate water molecules and place then in space. The command

```
wat1 = new SPCE
```

generates a single water molecule. We can generate another at a slightly displaced position by

```
wat2 = new SPCE.move(3.450, 0.0, 0.0)
wat3 = new SPCE.move(6.900, 0.0, 0.0)
```

which generates a new molecule at a position translated by 3.450 in the x-direction, and then at 0.6900 in the x-direction. We can simplify this an generate a $10 \times 10 \times 10$ box using the command

```
wat = new SPCE [10].move(0,0,3.45)
               [10].move(0,3.45,0)
               [10].move(3.45,0,0)
```

This generates a $10 \times 10 \times 10$ cubic lattice of SPCE water molecules. The full script, `spce-water-system.lt` reads as follows:

```
# -- file "spce-water-system.lt" --
import "spce-simple.lt"

wat  = new SPCE [10].move(0,0,3.45)
                [10].move(0,3.45,0)
                [10].move(3.45,0,0)

write_once("Data Boundary") {
   0.0  34.5  xlo xhi
   0.0  34.5  ylo yhi
   0.0  34.5  zlo zhi
}
```

You can generate the files for lammps by running the following command on the command line:

```
moltemplate.sh -atomstyle "full" spce-water-system.lt
```

This generates four files: `spce-water-system.in`, which is the file to be used by lammps; `spce-water-system.data` which is a file that is read by lammps, `spce_water_system.in.init` and `spce-water-system.in.setting`, which are included by the `spce-water-system.in` file. Notice that we also need to specify the system size, the boundaries of the simulation box, with the `write_once("Data Boundary")` command, which is followed by the system size.

The `spce-water-system.in` file will contain the following:

```
# ----------------- Init Section -----------------
include "spce-water-system.in.init"

# ----------------- Atom Definition Section -----------------
read_data "spce-water-system.data"

# ----------------- Settings Section -----------------
include "spce-water-system.in.settings"

# ----------------- Run Section -----------------
timestep        1.0
dump            1 all custom 10 traj_npt.lammpstrj id mol type x y z ix iy iz
fix             fxnpt all npt temp 300.0 300.0 100.0 iso 1.0 1.0 1000.0 drag 1.0
thermo          100
run             1000
```

Here, the part in the `Run Section` have been manually added afterwards to describe the particulars of the simulation. You can now run this Lammps-script using

```
lmp_serial < spce-water-system.in
```

and analyse the resulting simulation using Ovito.

m) Characterize the SPCE system using the tools you have developed previously. Find $g(r)$ and $D(T)$ for T over a reasonable range of temperatures for this system.

n) (Optional) Introduce an additional water model and redo the study for this model and compare with your results for SPCE. Comment on similarities and differences.

Tables of values.

Units. In all calculations for the Lennard-Jones system, we will use so-called MD units. These assume that all the particles in a simulation are identical, so the masses and LJ parameters can be factored out of the

equations. You will need to insert $A = \bar{A}A_0$ for every variable quantity A in the equations of motion. For example, for velocity, $v = \bar{v}\frac{L_0}{t_0}$. The time step Δt must also be treated this way. All non-numerical constants should disappear, except for in the velocity distribution standard deviation.

Quantity	Conversion factor	Value
Length	$L_0 = \sigma$	3.405 Å
Time	$t_0 = \sigma\sqrt{m/\epsilon}$	$2.1569 \cdot 10^3$ fs
Force	$F_0 = m\sigma/t_0^2 = \epsilon/\sigma$	$3.0303 \cdot 10^{-1}$ eV/Å
Energy	$E_0 = \epsilon$	$1.0318 \cdot 10^{-2}$ eV
Temperature	$T_0 = \epsilon/k_\mathrm{B}$	119.74 K

Table 0.1 Conversion factors A_0 from MD units for variable quantities.

In case you want to convert between your internal MD units and other units during input and output, the actual values of the conversion factors are listed in the table. These are calculated using the argon mass, lattice constant and LJ parameters: $m = 39.948$ amu, $a = 5.260$ Å (solid argon), $\sigma = 3.405$ Å, $\epsilon = 1.0318 \cdot 10^{-2}$ eV. Another common practice is putting $E_0 = 4\epsilon$, affecting the conversion factors F_0, T_0 and t_0.