Molecular Dynamics with C++

Lars Pastewka, Wolfram G. Nöhring, Lucas Frérot

April 20, 2022

Department of Microsystems Engineering
University of Freiburg

Contents

Chapter 1

Introduction

Context: We start by introducing the concept of the potential energy and the interatomic force. Those are the central ingredients to the molecular dynamics simulation method.

1.1 Structure of matter at the atomic scale

All matter is build out of quark and leptons, or perhaps even smaller particles, but for the sake of modeling the real material world the atom is the fundamental unit. Atoms can be described by nuclei and electrons or through "coarse-grained" models that ignore the fact that there are electrons. Both types of models are useful for describing materials, and the latter ones will be extensively used in this class.

Atoms in solids can arrange in different configurations that are called crystals when there is long-ranged order or glasses when there is not. (All solid matter typically has short-ranged order that is determined by the chemical bonds between atoms.) Atoms in solids are immobile and self-diffusion is limited. Conversely, liquids and gases are disordered (like glasses) but have mobile constituent atoms. Macroscopic object typically contain a lot of atoms – on the order of Avogadro's constant $N_A \approx 6 \times 10^{23}$. The atomic-scale simulation techniques discussed in this class can at the time of this writing (2020) treat on order of $\sim 10^6$ atoms, $10^8 - 10^9$ if you use the biggest computers available to us. Of course, this boundary is pushed towards larger systems as computer technology evolves.

Nowadays, we can even observe matter at atomic scales and "see" individual atoms. The collaborative research center 103 has produced an extremely

instructive video on the structure of specific type of alloys, dubbed "superalloy", that is used in e.g. turbine blades. Enjoy the ride from the blade to the atom. This class is about modeling matter at the smallest scales that you see in this video.

1.2 Interatomic forces and the potential energy

Atoms interact via forces. As Feynman put it in his famous lectures on physics, the fundamental truth about man's understanding of the physical world is "that all things are made of atoms – little particles that move around in perpetual motion, attracting each other when they are a little distance apart, but repelling upon being squeezed into one another". Indeed this is the essence of the molecular dynamics simulation method.

As the simplest example why atoms attract each other, let us consider the example of simple salt, e.g. Na-Cl that we all have sitting in our kitchen. Na-Cl in its solid form is an ionic crystal. Na atoms have approximately a charge of $q_{\mathrm{Na}} = +1|e|$, where e is the electron charge, and Cl atoms have a charge of approximately $q_{\mathrm{Na}} = -1|e|$. The Coulomb interaction between these atoms is a fundamental force of nature. Basic physical principles tell us, that the interaction energy between a Na and a Cl atom is given by

$$V_{\mathrm{Coulomb}}(r; q_{\mathrm{Na}}, q_{\mathrm{Cl}}) = \frac{1}{4\pi\varepsilon_0}\frac{q_{\mathrm{Na}}q_{\mathrm{Cl}}}{r}. \tag{1.1}$$

We also know that this energy is pair-wise additive, allowing us to write down the Coulomb interaction energy for Na-Cl consisting of N atoms,

$$E_{\mathrm{Coulomb}}(\{\vec{r}_i\}) = \sum_{i=1}^{N}\sum_{j=i+1}^{N} V_{\mathrm{Coulomb}}(r_{ij}; q_i, q_j) = \frac{1}{4\pi\varepsilon_0}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{q_i q_j}{r_{ij}} \tag{1.2}$$

where q_i is the charge on atom i and $r_{ij} = |\vec{r}_i - \vec{r}_j|$ the distance between atom i and j. Note that we have introduced — in passing — a central quantity of the molecular dynamics method, the atomic positions $\vec{r}_i$ and Eq. (1.2) indicates that the interaction energy depends on the positions of all atoms.

The Coulomb interaction has a singularity at $r \to 0$. The attractive force between opposite charges becomes infinitely large. The salt crystal does not collapse because atoms are, as Feynman puts it, "repelling upon being squeezed into one another". While the attraction between our Na and Cl atoms are described by a fundamental force of nature, it is more difficult to

understand the origin of this repulsion. Hand-wavingly, it goes back to the fact that electrons are Fermions and electrons from the electron shells of Na and Cl therefore cannot exist at the same point in space (and the same spin state). This is the Pauli exclusion principle, and the resulting repulsive force is called Pauli repulsion.

Different models for the Pauli repulsion exist. While the Coulomb interaction is a fundamental force of nature, these models are approximations to the true quantum physics that is the origin of the repulsive form. Two common forms are exponential repulsion,

$$E_{\text{rep,exp}}(\{\vec{r}_i\}) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} A e^{-r/\rho}, \tag{1.3}$$

or an algebraic repulsion of the form

$$E_{\text{rep,12}}(\{\vec{r}_i\}) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} A r^{-12}. \tag{1.4}$$

Note that A and ρ are *parameters*, that need to be somehow determined. This can be done with the help of either experimental data or *first-principles* calculations, that treat the electrons explicitly. These parameters depend on the atom types under consideration and, in contrast to the parameter that show up in the Coulomb interaction (the permittivity ε_0), they are not universal.

For our Na-Cl model, we combine Coulomb interaction with an exponential repulsion, to give the total energy

$$E_{\text{pot}}(\{\vec{r}_i\}) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(A_{ij} e^{-r_{ij}/\rho_{ij}} + \frac{1}{4\pi\varepsilon_0} \frac{q_i q_j}{r_{ij}} \right). \tag{1.5}$$

This energy is called the *potential energy* and is the central property of an atomic-scale model. With Eq. (1.5), we have also encountered our first atomic-scale model for a real material. Potentials that can be decomposed as Eq. (1.5) into pair-wise terms are called *pair potentials*. They are often written as

$$E_{\text{pot}}(\{\vec{r}_i\}) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} V(r_{ij}), \tag{1.6}$$

with

$$V(r_{ij}) = A_{ij} e^{-r_{ij}/\rho_{ij}} + \frac{1}{4\pi\varepsilon_0} \frac{q_i q_j}{r_{ij}} \tag{1.7}$$

for the above potential. The quantity $V(r_{ij})$ is called the pair interaction energy.

The most famous pair-potential is likely the Lennard-Jones potential. Its pair interaction energy is given by

$$V(r_{ij}) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right]. \tag{1.8}$$

The repulsive term $\propto r^{-12}$ is one of the models for Pauli repulsion discussed above. The attractive term $\propto r^{-6}$ arises from *London dispersion interactions*. Dispersion forces exist between all atoms, even uncharged molecules or noble gases. They are widely employed for the nonbonded portion of valence force-fields. Simple Lennard-Jones systems are often used to study generic phenomena found in real materials, e.g. the glass transition or plasticity of amorphous materials. However, there are limitations to pair potentials, and more sophisticated potential energy models have been developed over the past decades to address shortcomings of simple potentials. We will discuss a few of those in Chapter 3.

Note: A repulsive term of the form r^{-12} is advantageous from a simulation point of view, since it is faster to compute than an exponential. This has helped popularize the Lennard-Jones potential in the early days of molecular dynamics simulations.

Writing the potential energy of a system of particles allows the derivation of the forces acting on these particles:

$$\vec{f}_k = -\frac{\partial}{\partial \vec{r}_k} E_{\text{pot}}(\{\vec{r}_i\}). \tag{1.9}$$

These forces are the essential ingredient to *molecular dynamics*, as they determine the motion of the atoms, in accordance to Newton's second law.

The potential energy itself describes what is called the *potential energy landscape*. The potential energy landscape depends on $3N$ degrees of freedom (as compared to the landscape we experience while walking, which depends on 2 degrees of freedom); it is therefore an object that is complex to visualize. Simplifying some of its aspects is the core of *molecular statics*. For example, it is often important to identify the ground-state of a system; this is the most stable configuration of a material and has the lowest possible potential energy. There is usually some crystal that is lower in energy than the energy of a glass with the same stoichiometry. Yet, in many real-world engineering

applications, the materials are not in their crystalline ground-state: the most common material we encounter with this property may be window glass. In molecular statics, we therefore seek to enumerate those *local minima* of the potential energy landscape and the energy barriers between them.

Since the dynamics of a molecular system is determined by the forces, we only need to specify the potential energy up to a constant, which disappears in the derivative Eq. (1.9). We can therefore measure the potential energy with respect to any reference configuration. This reference configuration is often the atomized state of the material, where all the constituent atoms sit individually in vacuum and are not interacting with each other. If this reference situation is assigned the energy 0, then the potential energy is generally negative, because if it was positive the system would spontaneously atomize. (Remember, any physical system evolves to a state of lower energy.)

Chapter 2

Molecular dynamics

Context: Molecular *dynamics* follows the motion of individual atoms through a solution of Newton's equations of motion. We need integration algorithms to be able to solve this set of coupled differential equations on a computer.

2.1 Equations of motion

2.1.1 Newton's equations of motion

We have now (almost) all the ingredients to carry out a molecular dynamics simulation. From our or potential energy expression $E_{\text{pot}}(\{\vec{r}_i\})$ discussed in the previous chapter, we obtain the force

$$\vec{f}_i = -\partial E_{\text{pot}}/\partial \vec{r}_i \tag{2.1}$$

on each of the N atoms. Once we know the forces, we can obtain the accelerations $\vec{a}_i$ through Newton's third law,

$$\vec{f}_i = m_i \vec{a}_i. \tag{2.2}$$

We are therefore assuming that atom i can be described as a point of mass m_i! The mass can be obtained from the periodic table of elements. Note that the mass listed in the periodic table is usually the average over all isotopes weighted by their occurrence on earth, and this mass is used for most practical purposes. For some application, in particular to understand the different behavior of Hydrogen and Deuterium, it can be necessary to actually model the individual isotopes by using their respective mass.

We further have $\vec{a}_i = \dot{\vec{v}}_i$, where $\vec{v}_i$ is the velocity of atom i, and $\vec{v}_i = \dot{\vec{r}}_i$. The dot superscript indicates derivative with respect to time. The set of linear differential equations to solve is therefore

$$\dot{v}_i(t) = \vec{f}_i(t)/m_i \quad \text{and } \dot{\vec{r}}_i(t) = \vec{v}_i(t) \tag{2.3}$$

with the initial (boundary) conditions $\vec{r}_i(0) = \vec{r}_0$ and $\vec{v}_i(0) = \vec{v}_0$. Note that the boundary condition is an integral part of the differential Eq. (2.3). The state of the system is therefore fully and uniquely determined by the positions $\vec{r}_i$ and the velocities $\vec{v}_i$ of all atoms. This set of positions $\vec{r}_i$ and momenta $\vec{p}_i = \vec{v}_i/m_i$ defines a point in *phase-space* $\vec{\Gamma} = \{\vec{r}_i, \vec{p}_i\}$. The evolution of position and velocities given by Eq. (2.3) can therefore be thought of as a single point moving in the $6N$ dimensional phase-space. The concept of a phase-space will become important in the next chapter when we talk about statistical mechanics.

Code example: For a molecular dynamics code, it is useful to have a data structure that represents the state of the simulation and stores at least positions and velocities. This data structure could also store element names (or atomic numbers), masses and forces. An example that uses Eigen arrays as the basic array container is shown below. As a general rule, the data structure should be designed in a way that data that is processed consecutively is also stored in memory in a contiguous manner. This ensures predictable memory access patterns and efficient caching of the data necessary for computation. Instead of the `Atoms` container described below, we could be tempted to create a class `Atom` that contains the positions, velocities, etc. of a single atom and then use an array (e.g. `std::vector<Atom>`) of that class as the basic data structure. However, positions are then no longer consecutive in memory: they are interlaced with other atomic data. A function (e.g. computing forces) that does not need the velocities would still load them into the cache, causing more data transfers between cache and RAM, lowering performance. For high-performance numerical code, it is therefore *always* preferable to use structures of arrays rather than arrays of structure.

```
// Type aliases
using Positions_t = Eigen::Array3Xd;
using Velocities_t = Eigen::Array3Xd;
using Forces_t = Eigen::Array3Xd;

struct Atoms {
    Positions_t positions;
```

```
    Velocities_t velocities;
    Forces_t forces;

    Atoms(Positions_t &p)
        : positions{p},
          velocities{3, p.cols()},
          forces{3, p.cols()} {
        velocities.setZero();
        forces.setZero();
    }

    size_t nb_atoms() const {
        return positions.cols();
    }
};
```

2.1.2 Kinetic energy and energy conservation

In addition to the potential energy $E_{\text{pot}}(\{\vec{r}_i\})$, the dynamical state of a system is characterized by its kinetic energy,

$$E_{\text{kin}}(\{\vec{p}_i\}) = \sum_i \frac{1}{2}\frac{p_i^2}{m_i}. \tag{2.4}$$

Note: The *temperature* is simply a measure of the kinetic energy of the system, $\frac{3}{2}Nk_BT = E_{\text{kin}}$ where N is the number of atoms. In other words, E_{kin} measures the variance of the velocity distribution, which is Gaussian. We will learn more about this when discussing the basics of statistical mechanics.

The total energy

$$H(\{\vec{r}_i\},\{\vec{p}_i\}) = E_{\text{kin}}(\{\vec{p}_i\}) + E_{\text{pot}}(\{\vec{r}_i\}) \tag{2.5}$$

is a conserved quantity during the motion of the atoms. This can be seen by showing that the derivative of the total energy with respect to time vanishes,

$$\dot{H} = \dot{E}_{\text{kin}} + \dot{E}_{\text{pot}} = \sum_i \frac{\vec{p}_i\dot{\vec{p}}_i}{m_i} + \sum_i \frac{\partial E_{\text{pot}}}{\partial \vec{r}_i}\dot{\vec{r}}_i = \sum_i \vec{v}_i\vec{f}_i - \sum_i \vec{v}_i\vec{f}_i = 0. \tag{2.6}$$

H is also called the *Hamiltonian* of the system.

Note: Measuring the total energy (or any other conserved quantity!) and checking whether it is constant in a molecular dynamics simulation is a way of testing if the time step Δt used in the numerical integration is small enough. We will discuss numerical integration in detail below.

A generalization of Newton's equations of motion are *Hamilton's equations of motion*,

$$\dot{\vec{r}}_i = \frac{\partial H}{\partial \vec{p}_i} \tag{2.7}$$

$$\dot{\vec{p}}_i = -\frac{\partial H}{\partial \vec{r}_i}, \tag{2.8}$$

and it is straightforward to show that these equations reduce to Newton's equations of motions for the Hamiltonian given by Eq. (2.5). Hamilton's equation of motion remain valid when positions $\vec{r}_i$ and momenta $\vec{p}_i$ are replaced by generalized coordinates that consider constraints, such as for example the angle of a (rigid) pendulum. These equations will become important when we discuss statistical mechanics and temperature control in molecular dynamics simulations using *thermostats*, where a generalized degree of freedom is the internal state of the heat bath that controls the temperature. A full derivation of Hamilton's equations of motion is given in Chap. **??**.

2.2 Integration algorithms

The main ingredient in any molecular dynamics simulation, regardless of the underlying model, is the numerical solution of Eqs. (2.3). A plethora of algorithms have been developed over the years, but for most practical purposes the Velocity-Verlet algorithm is used nowadays. For instructive purposes we will start out with a simple integration method, the Euler integration, before discussing Velocity-Verlet.

2.2.1 Euler integration

In order to follow the trajectories of all atoms, we need to integrate the above differential equation. On a computer, a continuous differential equation needs to be replaced by a discrete equation. Equations (2.3) are continuous in time and hence need to be discretized. (Note that our system is already discrete spatially since we are dealing with mass points, but each of these points corresponds to a physical object, so this is not the result of a discretization

procedure.) The simplest numerical integration scheme is the forward Euler algorithm, in which forces and velocities are assumed to be constant over time intervals Δt.

To see this, we write the above differential equation as

$$d\vec{v}_i = \frac{\vec{f}_i(t)}{m_i}\, dt \quad \text{and } d\vec{r}_i(t) = \vec{v}_i(t)\, dt \tag{2.9}$$

i.e., we move the differential dt of $\dot{\vec{v}}_i = d\vec{v}/dt$ to the right hand side of the equation. We can now straightforwardly integrate the equation from time t to time $t + \Delta t$ while assuming that $\vec{f}_i$ and $\vec{v}_i$ remain constant. This yields

$$\vec{v}_i(t + \Delta t) - \vec{v}_i(t) = \frac{\vec{f}_i(t)}{m_i}\Delta t \tag{2.10}$$

$$\vec{r}_i(t + \Delta t) - \vec{r}_i(t) = \vec{v}_i(t)\Delta t \tag{2.11}$$

which is obviously only a good approximation for small Δt! This algorithm is called Euler integration.

The same equation can be derived by Taylor-expanding $\vec{r}_i(t + \Delta t)$ up to first order in Δt. The integration error of this algorithm is hence $O(\Delta t^2)$. The Euler algorithm is not reversible, i.e. starting from time $t+\Delta t$ and integrating backwards one ends up with a different result at time t. Applying the Euler algorithm with timestep $-\Delta t$ gives

$$\vec{v}_i(t) - \vec{v}_i(t + \Delta t) = -\frac{\vec{f}_i(t + \Delta t)}{m_i}\Delta t \tag{2.12}$$

$$\vec{r}_i(t) - \vec{r}_i(t + \Delta t) = -\vec{v}_i(t + \Delta t)\Delta t \tag{2.13}$$

These equations cannot be re-arranged to give Eqs. (2.10) and (2.11). Forward Euler integration is generally not a good algorithm and requires very small time steps.

2.2.2 Leap-frog integration

Leap-frog assumes positions are defined at times t_i and velocities at times $t_i + \Delta t/2$, and can be derived from an argument similar to the one given above. Specifically, we combine the results of a Taylor expansion $\pm\Delta t/2$, yielding

$$\vec{v}_i(t + \Delta t/2) - \vec{v}_i(t - \Delta t/2) = \frac{\vec{f}_i(t)}{m_i}\Delta t \tag{2.14}$$

$$\vec{r}_i(t + \Delta t) - \vec{r}_i(t) = \vec{v}_i(t + \Delta t/2)\Delta t. \tag{2.15}$$

Note that Eq. (2.14) is similar to Eq. (2.10), except the force is evaluated at the mid-point. The resulting algorithm is reversible. Applying the Leap-frog algorithm with timestep $-\Delta t$ gives

$$\vec{v}_i(t - \Delta t/2) - \vec{v}_i(t + \Delta t/2) = -\frac{\vec{f}_i(t)}{m_i}\Delta t \tag{2.16}$$

$$\vec{r}_i(t) - \vec{r}_i(t + \Delta t) = -\vec{v}_i(t + \Delta t/2)\Delta t \tag{2.17}$$

Bring the terms on the left hand side to the right and vice-versa, and you arrive at the original equations for forward integration. Leap-frog is therefore *reversible*.

2.2.3 Verlet integration

Let us now Taylor expand $\vec{r}_i(t \pm \Delta t)$ up to third order in $\pm\Delta t$,

$$\vec{r}_i(t \pm \Delta t) = \vec{r}_i(t) \pm \vec{v}_i(t)\Delta t + \frac{1}{2m_i}\vec{f}_i(t)\Delta t^2 \pm \frac{1}{6}\dddot{\vec{r}}_i(t)\Delta t^3 + O(\Delta t^4). \tag{2.18}$$

Note that only the odd exponents see the sign of $\pm\Delta t$. The sum of this equation for expansion in $+\Delta t$ and $-\Delta t$ gives the positions update,

$$\vec{r}_i(t + \Delta t) + \vec{r}_i(t - \Delta t) = 2\vec{r}_i(t) + \frac{1}{m_i}\vec{f}_i(t)\Delta t^2 + O(\Delta t^4). \tag{2.19}$$

Eq. (2.19) is called the Verlet algorithm. Instead of requiring the positions $\{\vec{r}_i(t)\}$ and velocities $\{\vec{v}_i(t)\}$ it requires the positions of the current $\{\vec{r}_i(t)\}$ and past $\{\vec{r}_i(t - \Delta t)\}$ times for the integration.

The difference between the expansion for $+\Delta t$ and $-\Delta t$ yields the velocities,

$$\vec{r}_i(t + \Delta t) - \vec{r}_i(t - \Delta t) = 2\vec{v}_i(t)\Delta t + O(\Delta t^3). \tag{2.20}$$

Note that in order to compute the velocities at time t in the regular Verlet algorithm, we need to know the positions at time $t + \Delta t$. Verlet and Leap-Frog are identical algorithms, since Leap-Frog stores the velocities at the intermediate time $t + \Delta t/2$. It is usually useful to be able to know both, positions and velocities, at time t. This problem is solved by the Velocity-Verlet algorithm, described in the following section.

2.2.4 Velocity-Verlet integration

Let us now also Taylor expand $\vec{r}_i(t)$ up to third order in Δt at $\vec{r}_i(t + \Delta t)$, i.e. we integrate backwards in time from $t + \Delta t$ to t. This gives

$$\vec{r}_i(t) = \vec{r}_i(t + \Delta t) - \vec{v}_i(t + \Delta t)\Delta t + \frac{1}{2m_i}\vec{f}_i(t + \Delta t)\Delta t^2 - \frac{1}{6}\dddot{\vec{r}}_i(t)\Delta t^3 + O(\Delta t^3) \tag{2.21}$$

Equation (2.18) is the positions update of the Velocity-Verlet algorithm. The sum of Eq. (2.18) and Eq. (2.21) gives the velocity update in the Velocity-Verlet algorithm:

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\Delta t + \frac{1}{2m_i}\vec{f}_i(t)\Delta t^2 \tag{2.22}$$

$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t) + \frac{1}{2m_i}\left(\vec{f}_i(t) + \vec{f}_i(t+\Delta t)\right)\Delta t, \tag{2.23}$$

Note that this algorithm is often split in the form of a predictor-corrector scheme since this saves computation time and the necessity to keep past forces around. The predictor step is

$$\vec{v}_i(t+\Delta t/2) = \vec{v}_i(t) + \frac{1}{2m_i}\vec{f}_i(t)\Delta t \tag{2.24}$$

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \vec{v}_i(t+\Delta t/2)\Delta t \tag{2.25}$$

where $\vec{v}_i(t+\Delta t/2)$ is the predicted velocity. After this we compute new forces, $\vec{f}_i(t+\Delta t)$. We then correct the velocities via

$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t+\Delta t/2) + \frac{1}{2m_i}\vec{f}_i(t+\Delta t)\Delta t \tag{2.26}$$

The Velocity-Verlet algorithm is the integration algorithm used in most molecular dynamics codes. It has the additional properties that is it *symplectic*, which means it conserves phase-space volume. We will come back to what this mean when talking about statistical mechanics.

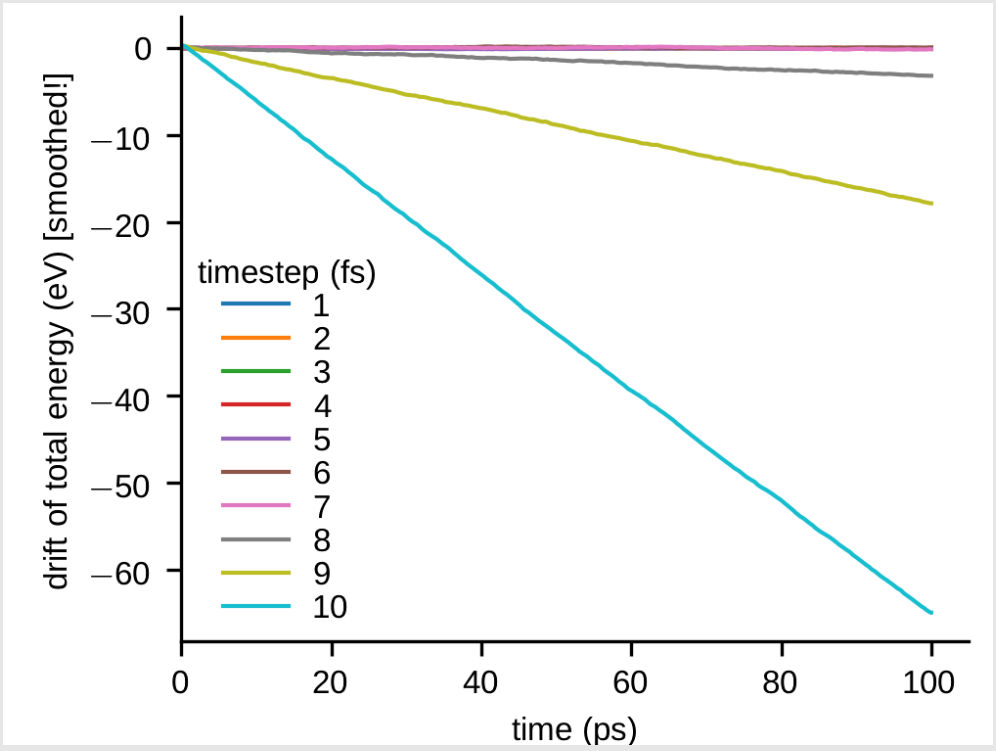
Code example: We can implement the velocity-verlet algorithm in a few lines of C++ code using vectorized `Eigen` operations. The prediction step

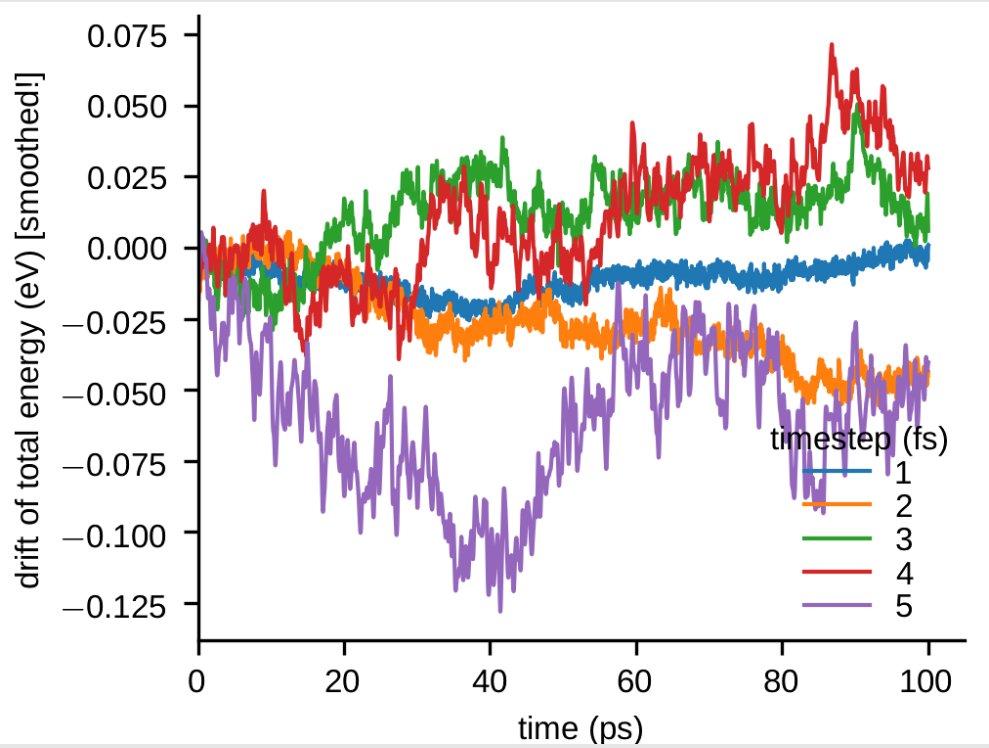
```
void verlet_step1(Atoms &atoms, double timestep,
                  double mass) {
    atoms.velocities += 0.5 * atoms.forces * timestep /
   mass;
    atoms.positions += atoms.velocities * timestep;
}
```

implements Eq. (2.24). We then compute new forces and correct the velocities via

```
void verlet_step2(Atoms &atoms, double timestep,
                  double mass) {
    atoms.velocities += 0.5 * atoms.forces * timestep /
   mass;
}
```

Note: The timestep in MD simulations has to be on the order of femtoseconds, in order to resolve the fastest atomic vibrations. For example, in simulations with metals and Embedded Atom Method (EAM) potentials, $\Delta t = 1$ fs is typically a safe choice. How can we check that the timestep is sensible? One possibility is to simply let a configuration in time using the Velocity-Verlet algorithm. This is sometimes called the micro-canonical or NVE ensemble. (NVE because number of atoms, volume and energy is constant.) We then record the evolution of the total (kinetic plus potential) energy, which should be constant. Due to the approximations described above, discrete time integration schemes introduce numerical errors. If Δt is larger than a critical value, the integration error grows unstable and causes a noticeable drift of the total energy. The figures below show the results of such a simulation. A system of 108000 Au atoms was simulated for 100 ps with various values of Δt. The y-axis shows the difference between the current and initial values of the total energy. The data was smoothened to suppress high-frequency fluctuations in the figure. For this system, even 5 fs would still be an acceptable time step.




0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.125
drift of total energy (eV) [smoothed!]
0
20
40
60
80
100
time (ps)
timestep (fs)
1
2
3
4
5

Chapter 3

Pair potentials

Context: Interatomic forces or interatomic potentials determine the material that we want to study. There is a plethora of interatomic potentials of varying accuracy, transferability and computational cost available in the literature. We here discuss simple pair potentials and point out algorithmic considerations.

3.1 Introduction

The expression for $E_{\text{pot}}\left(\{\vec{r}_i\}\right)$ is the *model for the material* that we use in our molecular dynamics calculations. It determines whether we model water, proteins, metals, or any other physical object. Models are typically characterized by their *accuracy*, their *transferability* and the *computational cost* involved. (Computational cost also includes the *computational complexity*.) At constant computational cost, there is always a tradeoff between accuracy and transferability. Accuracy and transferability can typically only be improved at the expense of additional computational cost.

- *Accuracy:* How close to we can get to a reference metric, experimentally measured or theoretical. For example, we can compare the vacancy formation energy to experimental values, and compute accuracy as the absolute value of the energy difference $E_{\text{vac}} - E_{\text{vac}}^{\text{exp}}$, which can be 1 eV, 0.1 eV (typical), 0.01 eV (computationally expensive!). (The vacancy formation energy is the energy required to remove a single atom from a solid. The resulting "hole" in the solid is called a vacancy.)

- *Transferability:* The ability for a model to satisfy different accuracy metrics. Let's assume we get the vacancy formation energy right to

within 0.1 eV of the experimental value. Does the interstitial formation energy, i.e. the energy to insert an additional atom between lattice sites, give the same value? If so, then the potential is transferable between these two situations. *Most interatomic potentials are not generally transferable,* and they need to be tested when used in new situations, e.g. when the potential has been used to study crystals, but you want to use it to study a glass.

- *Computational cost:* The number of floating point operations determine how expensive it is to compute an energy or a force. (Nowadays, actual electrical power requirements for doing the calculation would be a better measure.) This is related to computational complexity, that describes how the computational cost (i.e. the number of operations required to compute the result) scales with the number of atoms. Ideally we would like $O(N)$ complexity (i.e. a system with twice as many particles takes twice the computing time), but many methods do not scale linearly. Quantum methods (tight-binding, density-functional theory) are usually $O(N^3)$ or worse.

3.2 Pair potentials

We have already encountered the simplest (and oldest) form of interaction potential, the pair potential. The total energy for a system interacting in *pairs* can be written quite generally as

$$E_{\text{pot}}\left(\{\vec{r}_i\}\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V\left(r_{ij}\right) = \sum_{i<j} V(r_{ij}) \tag{3.1}$$

where $r_{ij} = |\vec{r}_i - \vec{r}_j|$ is the distance between atom i and atom j. $V(r_{ij})$ is the pair interaction energy or just the pair potential and we assume that the interaction is pair-wise additive. The sum on the right ($\sum_{i<j}$) runs over all pairs while sum on the left double counts each pair and therefore needs the factor 1/2. We have already seen a combination of the electrostatic potential and Pauli repulsion as an example of a pair-potential earlier.

Forces are computed by taking the negative gradient of this expression. The force on atom k is given by

$$\vec{f}_k = -\frac{\partial E_{\text{pot}}}{\partial \vec{r}_k} = -\frac{1}{2}\sum_{ij}\frac{\partial V}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \vec{r}_k} = -\frac{1}{2}\sum_{ij}\frac{\partial V}{\partial r_{ij}}\hat{r}_{ij}\left(\delta_{ik} - \delta_{jk}\right) = \sum_{i}\frac{\partial V}{\partial r_{ik}}\hat{r}_{ik}, \tag{3.2}$$

where $\hat{r}_{ik} = \vec{r}_{ik}/r_{ik}$ is the unit vector pointing from atom k to atom i. Note that these forces are symmetric, i.e. the term $\partial V/\partial r_{ik}\hat{r}_{ik}$ shows up in the expression not only for the force on atom k, but also (with an opposite sign) for the force on atom i. This is Newton's third law, a consequence of momentum conservation. (The sum over all forces needs to be equal to the applied external forces.) A typical implementation would therefore loop over all *pairs* between atoms, compute this pair term, then add it to the array entries holding the forces for both atoms.

3.2.1 Dispersion forces

An important contribution to interatomic and intermolecular interactions is the London dispersion force. This interaction is attractive, and acts between all atoms, even noble gases. Its origin lies in fluctuations of the atomic dipole moment. (This is a quantum mechanical effect, but the simplest model would be an electron orbiting a nucleus with a rotating dipole moment.) This fluctuating dipole *induces* a dipole in a second atom and these interact. The interaction decays as r^{-6} at short distances. London dispersion forces are one of the forces that are often subsumed under the term van-der-Waals interaction.

3.2.2 Lennard-Jones potential

The Lennard-Jones potential combines dispersion forces with an empirical r^{-12} model for Pauli repulsion. It is typically used for the interaction of noble atoms or molecules, i.e. systems that have closed electronic shells and therefore do not form covalent bonds. The interactions described by the Lennard-Jones potential are often called nonbonded interactions, because the typical interaction energy is on the order of $k_B T$ (with room temperature for T). Thermal fluctuation can thereby break this bond, hence the term nonbonded.

One typical form to writing the Lennard-Jones potential is

$$V(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{3.3}$$

where ε is an energy and σ a length. The potential has a minimum as $r = 2^{1/6}\sigma$ and is repulsive for shorter distances and attractive for larger distances. For a noble gas (e.g. Argon), $\varepsilon \sim 0.01$ eV and $\sigma \sim 3$ Å.

3.3 Short-ranged potentials

Implementing Eq. (3.1) naively leads to a complexity of $O\left(N^2\right)$ because the sum contains N^2 terms. The trick is to cut the interaction range, i.e. set energies and forces to zero for distances larger than a certain cut-off distance r_c. This is possible because $V(r) \to 0$ as $r \to \infty$. Potentials for which this asymptotic decay is fast enough can be cut-off and are called short-ranged. Note that we have already encountered a case in Chap. 1 for which this is not possible: the Coulomb interaction that has the form $V(r) \propto 1/r$, which decays to 0 too slowly.

A simple way to see why this is not possible for the Coulomb interaction is to lump the charge-neutral infinite solid into charge-neutral dipoles. The effective interaction between dipoles then falls of as $V^{\text{eff}}(r) \propto 1/r^3$. The contribution to the energy from all dipoles at distance r is $V(r)\, r^2 \propto 1/r$. The full energy is obtained by integrating this function over r, but the integral does not converge! This illustrates the problem. The discrete sum is convergent, but only conditionally so, i.e. the outcome depends on the order of summation. We therefore can only cut interactions that decay as r^{-4} or faster.

The potential energy with a cutoff looks as follows:

$$E_{\text{pot}}\left(\{\vec{r}_i\}\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{\{j|r_{ij}<r_c\}} V\left(r_{ij}\right) \tag{3.4}$$

The difference to Eq. (3.1) is that the second sum runs only over *neighbors* of i, i.e. those atoms j whose distance $r_{ij} < r_c$ where r_c is the cutoff radius. This sum has $N\bar{n}$ elements where $\bar{n}$ is a constant, is the average number of neighbors within the cutoff radius r_c. The complexity of an algorithm that implements the above sum is hence $O(N)$.

A simple pair potential is often shifted by a constant to make the pair interaction energy continuous at $r = r_c$ (since in general $V(r_c) \neq 0$). The potential energy expression is then $E_{\text{pot}} = \sum_{i<j}\Big(V\left(r_{ij}\right) - V(r_c)\Big)$. Note that only by shifting the potential, forces and potential energy become consistent. Since only the forces affect the dynamics, the potential energy must be continuous and the integral of the forces, otherwise the Hamiltonian H is not a conserved quantity. The shifted potential fulfills these requirements, the unshifted one does not.

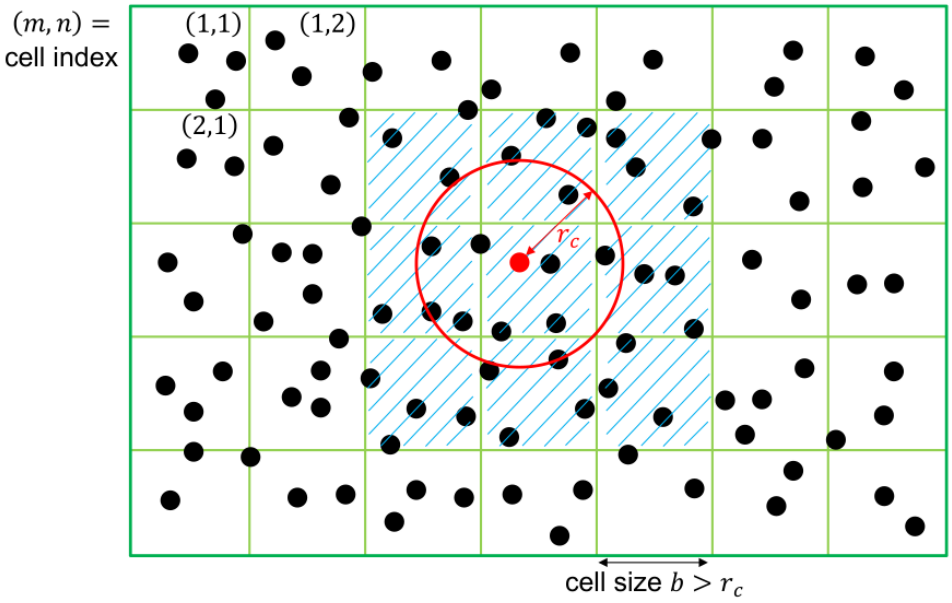


Figure 3.1: Illustration of the typical data structure used for an $O(N)$ neighbor search in a molecular dynamics simulation. For searching the neighbors within a cutoff r_c of the red atom, we only need to consider the candidate atoms that are in the cells adjacent to the red atom.

3.4 Neighbor list search

The sum Eq. (3.4) runs over all neighbors. One important algorithmic step with complexity $O(N)$ in molecular dynamics codes is to build a *neighbor list*, i.e. find all pairs i-j with $r_{ij} < r_c$. This is usually done using a *domain decomposition* (see Fig. 3.1) that divides the simulation domain in cells of a certain size and sorts all atoms into one of these cells. The neighbor list can then be constructed by looking for neighbors in neighboring cells only. If the cell size b is larger than the cutoff radius, $b > r_c$, then we only need to look exactly the neighboring cells.

We will here illustrate a typical neighbor search using the two-dimensional example shown in Fig. 3.1. Let us assume that each atom has a unique index $i \in [1, N]$, where N is the total number of atoms. (Note: in C++ and other common languages, indices start at 0 and run to $N - 1$.) A neighbor search algorithm first builds individual lists $\{B_{k,mn}\}$ that contain the indices off all atoms in cell (m, n), i.e. $k \in N_{nm}$ where N_{nm} is the number of atoms in this cell. The cell can simply be determined by dividing the position of the atom by the cell size b, i.e. atom i resides in cell $m_i = \lfloor x_i/b \rfloor$ and $n_i = \lfloor y_i/b \rfloor$

where $\lfloor \cdot \rfloor$ indicates the closest smaller integer. The lists $\{B_{k,mn}\}$ are most conveniently stored in a single contiguous array; for purposes of accessing individual cells, a second array is required that stores the index of the first entry of the cell (m, n). Note that this second array's size is equal to the number of cells, and can become prohibitively large when the system contains a lot of vacuum.

The neighbor search then proceeds as follows: for atom i, compute the cell (m_i, n_i) in which this atom resides and then loop over all atoms in this cell and in cells $(m_i \pm 1, n_i)$, $(m_i, n_i \pm 1)$ and $(m_i \pm 1, n_i \pm 1)$. In two dimensions, this yields a loop over 9 cells, in three-dimensions there the loop runs over 27. If the distance between these two atoms is smaller than the cutoff r_c, we add it to the neighbor list. Note that if the cell size b is smaller than r_c, we need to include more cells in the search.

Chapter 4

Temperature control

Context: Most molecular dynamics calculations are carried out in *thermal equilibrium*. Equilibrium is typically maintained by coupling the molecular calculation to a virtual *heat bath*, with which it exchanges energy but no particles. This chapter discusses properties of thermal equilibrium and introduces simple algorithms for heat-bath coupling.

4.1 Introduction

In order to talk about temperature control, we need to discuss the properties of thermal equilibrium. This is the realm of *statistical mechanics* or *statistical thermodynamics* that is discussed in more detail in Chapter **??** and Appendix **??**. A key outcome is that the velocity components are distributed according to a Boltzmann distribution. The velocity magnitude is then distributed according to a Maxwell-Boltzmann distribution.

A thermostat implicitly models the coupling to a heat bath much larger than the atomistic system under investigation. Because it is much larger, its temperature will not change when energy flows from and to the heat bath. The atomistic system becomes canonical and its statistics follows the *canonical ensemble*. An ensemble here describes which parameters are constrained, and the canonical ensemble is often also called the *NVT-ensemble*, because particle number N, volume V and temperature T are constrained (fixed). An ideal thermostat guarantees relaxation of the distribution of atomic degrees of freedom to the *canonical distribution function* (see Chapter **??**).

We will here start with a mechanistic treatment of thermostats and underpin it with more rigorous theory in Chapter **??**. The present chapter teaches the basic concepts required for an implementation of simple thermostatting schemes. Thermostats can be roughly categorized into constraint

methods (velocity rescaling and Berendsen), stochastic methods (Andersen, Langevin and dissipative particle dynamics) and extended system methods (Nosé-Hoover). Constraint and extended system methods are deterministic, i.e. they follow the same path when starting from the same initial state. In this chapter we will only discuss the simple constraint methods. We will come back to more advanced methods for temperature control later in these notes.

4.2 Simple themostatting schemes

4.2.1 Velocity rescaling

The crudest (and simplest) form of fixing the temperature in a molecular dynamics simulation to a value of T_0 is by velocity rescaling. Since the instantaneous temperature is

$$\frac{3}{2}Nk_BT = \sum_i \frac{1}{2}mv_i^2, \tag{4.1}$$

we obtain a temperature of T_0 if we rescale all velocities by

$$\vec{v}_i \to \lambda\vec{v}_i \quad \text{with} \quad \lambda = \sqrt{\frac{T_0}{T}} \tag{4.2}$$

after every time step. This is a very intrusive way of setting the temperature and should not be used in any practical situations, but it is a good illustration of how a simple constraint method works.

4.2.2 Berendsen thermostat

The Berendsen et al. (1984) thermostat uses a damping or acceleration term to control the temperature. The governing equations of motion of the Berendsen thermostat are

$$m\dot{\vec{v}}_i = \vec{f}_i + \frac{m}{2\tau}\left(\frac{T_0}{T} - 1\right)\vec{v}_i \tag{4.3}$$

where τ is a relaxation time constant. The factor in front of the velocity is a damping coefficient. The coefficient vanishes for $T = T_0$, Eq. (4.3) then reduces to Newton's equation of motion. However, it has a positive sign (=speeds up particles) for $T < T_0$ and has negative sign (=slows down particles) for $T > T_0$. From Eq. (4.3) we can easily derive a differential

equation for the evolution of the temperature:

$$3k_B \frac{dT}{dt} = \sum_i m\vec{v}_i \cdot \dot{\vec{v}}_i \tag{4.4}$$

$$= \sum_i \left[\vec{v}_i \cdot \vec{f}_i + \frac{1}{2\tau} \left(\frac{T_0}{T} - 1 \right) m v_i^2 \right] \tag{4.5}$$

$$= -\frac{dE_{\text{pot}}}{dt} + \frac{3k_B(T_0 - T)}{\tau} \tag{4.6}$$

This can be written as

$$\frac{dT}{dt} = -\frac{T - T_0}{\tau} + S \tag{4.7}$$

where $S = -\frac{1}{3k_B}\frac{dE_{\text{pot}}}{dt}$ is the change of *potential* energy and constitutes an additional temperature (energy) source.

For $S = 0$, this equation is solved by

$$T(t) = T_0 + (T_1 - T_0)e^{-t/\tau} \tag{4.8}$$

The temperature relaxes exponentially from the intial value T_1 towards T_0. We directly see that τ in Eq. (4.3) is indeed the relaxation time constant.

Note that Eq. (4.8) suggests an implementation of the Berendsen thermostat in terms of velocity rescaling. During at single time step $\Delta t \ll \tau$, the temperature changes from T to $T_0 + (T - T_0)e^{-\Delta t/\tau}$. We can implement this as velocity rescaling, Eq. (4.2), with

$$\lambda = \sqrt{\frac{T_0}{T} + \left(1 - \frac{T_0}{T}\right) e^{-\frac{\Delta t}{\tau}}} \approx \sqrt{1 + \left(\frac{T_0}{T} - 1\right)\frac{\Delta t}{\tau}} \tag{4.9}$$

where T is the current (measure) temperature and T_0 is the target temperature.

A Berendsen thermostat therefore constitutes a gentle way of rescaling velocities. The relaxation time τ determines the strength of the coupling between thermal bath and atomistic system. The velocity rescaling limit $\lambda \to \sqrt{T_0/T}$ is obtained as $\tau \to 0$. Thermostats should be tuned as weak as possible and as strong as necessary to disturb the system the least while still allowing it to reach the target temperature within the simulation time. There is the additional requirement $\tau \gg \Delta t$ (where Δt is the time step), otherwise equation Eq. (4.7) will not be sampled properly numerically. The velocity rescaling thermostat discussed above is bad because it is very strong, but also because it violates $\tau \gg \Delta t$.

4.3 Equilibrating a molecular simulation

A "happy" molecular dynamics simulation will nicely run at constant temperature. Simulations are only this happy once they are *equilibrated* and this equilibration implies that the positions $\{\vec{r}_i\}$ are such that the system resides somewhere near a (potentially local) minimum in the potential energy landscape. When we set up a new simulation, we have to guess a set of $\{\vec{r}_i\}$ that are often far away from this minimum. (For crystalline solids this guess is simple, since we typically know the crystal structure that we are interested in. For liquids, the guess is more difficult since the overall structure is disordered.) Since the forces $\{\vec{f}_i\}$ point towards the minimum, the system will evolve in this direction and the potential energy E_{pot} will decrease over time, $dE_{\text{pot}}/dt < 0$. Equation (4.7) tells us, that this leads to an increase in temperature since $S > 0$.

A common problem is that this temperature can be large enough to vaporize the system, i.e. the temperature increases above the vaporization point. The first step in any molecular dynamics simulation is hence to *equilibrate* the system while avoiding a temperature rise above the point of vaporization (or melting if you are setting up a solid). This can be achieved by running a calculation with a Berendsen thermostat and a strong coupling (i.e. a small τ). Once the system has equilibrated, the value of τ can be adjusted to a more reasonable relaxation time that does not disturb the calculation too much. Good values for τ are between 1 ps and 10 ps.

Note that if we *continuously* pump energy into our system, for example because we deform it externally, then Eq. (4.7) acquires a non-zero source term, $S > 0$. Assuming S is constant over time, the final temperature is shifted to $T_0{}' = T_0 + S\tau$. This temperature offset gets smaller with increasing coupling strength $1/\tau$.

Chapter 5

Embedded-atom method potentials

Context: We here introduce a more complex interatomic potential that is suitable for modeling metals, the embedded atom method potential. It belongs to the class of *many-body* interatomic potentials and can be used to model mechanical or thermodynamic properties of metals.

5.1 Introduction

Metals are often cubic crystals with anisotropic mechanical properties. Crystals with cubic symmetry have three independent elastic constants, C_{11}, C_{12} and C_{44} that roughly describe the resistance to volume change, dilational shear and simple shear. The original driving force behind the development of the embedded atom method (EAM) was to overcome the zero Cauchy pressure $P_C = (C_{11} - C_{44})/2$ for solids obtained for pair potentials: Pair potentials always satisfy the *Cauchy relation* $C_{11} = C_{44}$, hence there are only two independent elastic constants for cubic solids. [Compare: For an *isotropic* solid there are also two independent elastic constants, but this condition is different, bulk modulus $K = (C_{11} + 2C_{12})/3$ and shear modulus $G = C_{44} = (C_{11} - C_{12})/2$.] The Cauchy relation can be relaxed by adding an energy term that depends on the volume per atom $v = V/N$ (Vitek, 1996)

$$E(\{\vec{r}_i\}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V(r_{ij}) + NU(V/N) \tag{5.1}$$

The *volume dependent term* contributes only to deformation modes that do not conserve the volume, i.e. C_{11} or C_{12}. This hence breaks the Cauchy

relationship $C_{12} = C_{44}$ and gives a non-zero Cauchy-pressure $P_C = (C_{12} - C_{44})/2$

While a potential of the type given by Eq. (5.1) can be adjusted to give the correct elastic constant (and can therefore be accurate), *it cannot be used for e.g. free surfaces* (and is therefore not transferable). This has historically driven the development of more advanced methods for modeling solids such as the EAM described here. Note that EAM potentials are not confined to the realm of solids but can also be used for studying properties of melt, or the transition between solid and melt.

5.2 Functional form

The EAM is based on the assumption that the energy of an impurity in a host crystal lattice is a functional of the overall electron density $\rho(\vec{r})$ (that leads to an attraction), plus some form of repulsion (i.e. due to Pauli exclusion). This can be written as $E_{\text{pot}} = \mathcal{F}\left[\rho(\vec{r})\right] + \phi$, where $\mathcal{F}$ is called the embedding functional that tells us the relationship between energy and electron density and ϕ some repulsive interaction.

We view each individual atom in the system as an impurity in the host consisting of all other atoms (Daw and Baskes, 1983). $\mathcal{F}$ is then approximated by a *function* that depends on the *local* electron density ρ_i at atom i:

$$E_{\text{pot}}(\{\vec{r}_i\}) = \sum_i \mathcal{F}(\rho_i) + \frac{1}{2}\sum_{i,j} \phi(r_{ij}) \tag{5.2}$$

Note the first sum is over atoms, not pairs, and the second term is a simple pair interaction. The missing ingredient is now the local electron density ρ_i, which we approximate from the local density of the nuclei. This assumes that each atom in the vicinity of atom i contributes a certain number of electrons to the position of atom i.

The embedding function $\mathcal{F}(\rho)$ is negative and (typically) decreases monotonously with increasing density. The more closely a structure is packed the lower the energy. The repulsive term that is physically due to electrostatic and Pauli repulsion then stabilizes the structure. This is balance between attractive and repulsive contribution a common feature of most interatomic potentials, and we have already seen it for the Lennard-Jones potential.

The local density of the atomic system is easily computed from

$$\rho_i = \sum_j f(r_{ij}) \tag{5.3}$$

If $f(r)$ is a step function that drops to zero at a distance r_c then ρ_i becomes the coordination number, i.e. the number of atoms within a sphere of radius r_c. By normalizing the step function with the volume of the sphere, it becomes clear that ρ_i is some measure of the average atomic density within a distance r_c from atom i. However, a step function is not differentiable. All distance dependent functions are therefore smoothly connected to zero at a distance r_c (the cutoff). This makes the whole functional form differentiable at least once!

Examples of early EAMs are Gupta (1981), Finnis and Sinclair (1984) and Cleri and Rosato (1993). They all employ the specific functional forms

$$\mathcal{F}(\rho) = -A\sqrt{\rho} \tag{5.4}$$

$$f\left(r_{ij}\right) = e^{-2q(r_{ij}-r_0)} \tag{5.5}$$

$$V\left(r_{ij}\right) = Be^{-p(r_{ij}-r_0)} \tag{5.6}$$

where A, B, q, p and R_0 are parameters. For example, Cleri and Rosato (1993) give parameters for the elements Ni, Cu, Rh, Pd, Ag, Ir, Pt, Au, Al, Pb, Ti, Zr, Co, Cd, Zn and Mg. Note that the cutoff radius r_c in most potentials based on the embedded-atom approach reaches out to second nearest neighbors or further, e.g. to fifth nearest neighbor for fcc metals in the Cleri and Rosato (1993) potential. These potentials do not describe fundamental forces of nature but they must be parametrized for a specific material. The parametrization also includes choice of cutoff radius r_c.

5.3 Parameterization

There exist different strategies to actually determine the parameters of a potential. Cleri and Rosato (1993), as an example, have five parameters and they fit the potential directly to *experimental values* of the cohesive energy, lattice constant and the three cubic elastic constants.

Some authors adjust the either embedding function or repulsive pair potential to reproduce the *universal equation of state* (see Ferrante et al. (1983); Rose et al. (1984)). For example, Foiles et al. (1986) obtain $f(r_{ij})$ from the electron density of free atom calculations, and assume the pair repulsion is entirely electrostatic, $V\left(r_{ij}\right) = Z_i\left(r_{ij}\right) Z_j(r_{ij})/r_{ij}$ (with atomic charges Z_i actually depending on the distance between atoms, $Z\left(r_{ij}\right) = Z_0\left(1+\beta R^{\nu}\right)\exp\left(-\alpha r_{ij}\right)$ where Z_0, β, ν and α are parameters). The embedding function $F(\rho)$ is then adjusted to reproduce the universal equation of state. Note that Foiles et al. (1986) have more parameters in their model than Cleri and Rosato (1993)!

A more modern approach is *force matching* due to Ercolessi and Adams (1994). Force matching potentials are fit to a set of calculations carried out

with a more accurate and more transferable but also more expensive method (e.g. a quantum chemical method) at finite temperature. This generates a molecular dynamics trajectory that has configurations with nonzero forces on each atom. (Fitting to equilibrium properties such as Cleri-Rosato means fitting to structures where all forces are zero.) The potential parameters are then fit to reproduce these forces. This method has the advantage that, in principle, an unlimited set of fitting target can be generated easily and the potential can be fit to a large number of parameters. An example of a force-matched EAM is Grochola et al. (2005). It has no fixed functional form, but splines are used to represent the three functions $\mathcal{F}(\rho)$, $f(r)$ and $V(r_ij)$. Figure 5.1 shows these functions for the Grochola et al. (2005) potential.

Note: While early EAM potentials had a purely attractive embedding contribution $\mathcal{F}(\rho)$ and a purely repulsive pair contribution $\phi(r)$, this condition is relaxed in more complex potential. As can be seen from Fig. 5.1, Grochola et al. (2005)'s potential includes a repulsive contribution from the embedding term.

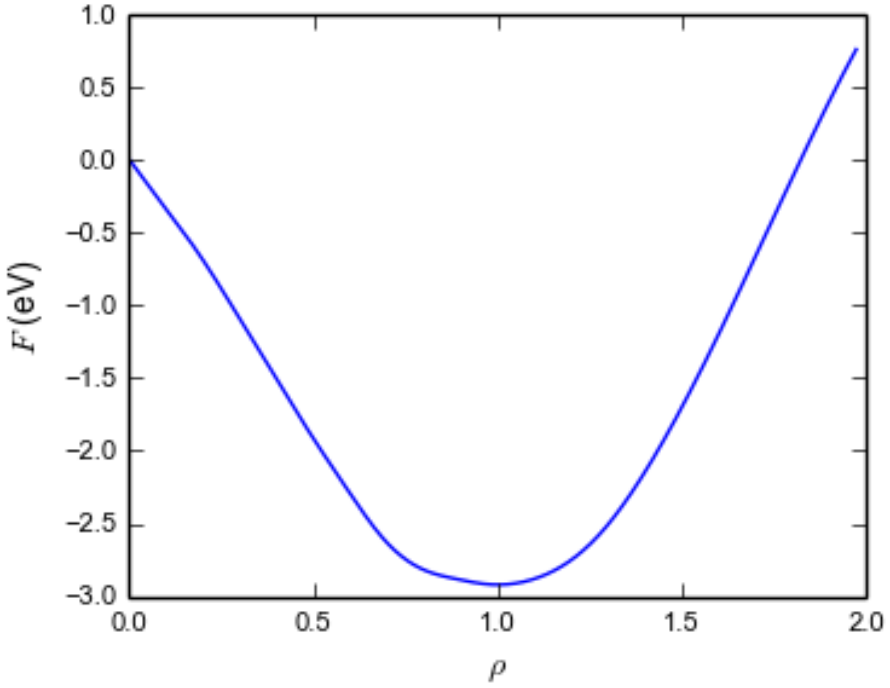
Note: Note that these two approaches, fitting to experimental ground-state data and force-matching, are quite different from a philosophical point of view. It has been argued by Sukhomlinov and Müser (2016), that the potential should contains as few parameters as possible (Occam's razor!) to achieve best transferability. Potential with many parameters are often accurate for the fitting data set but not accurate outside and hence not transferable. This problem is typically referred to as overfitting.

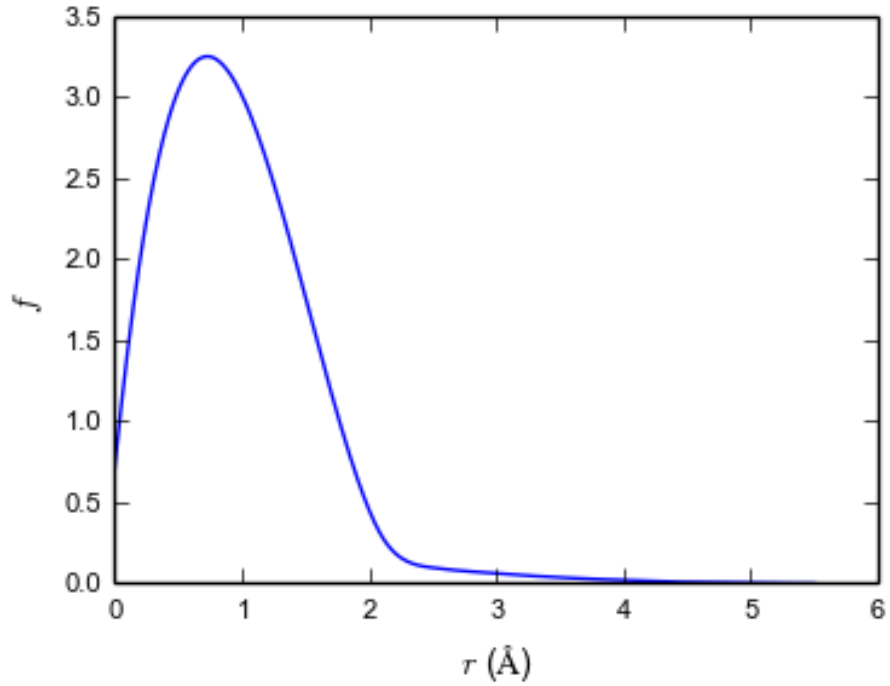
5.4 Forces

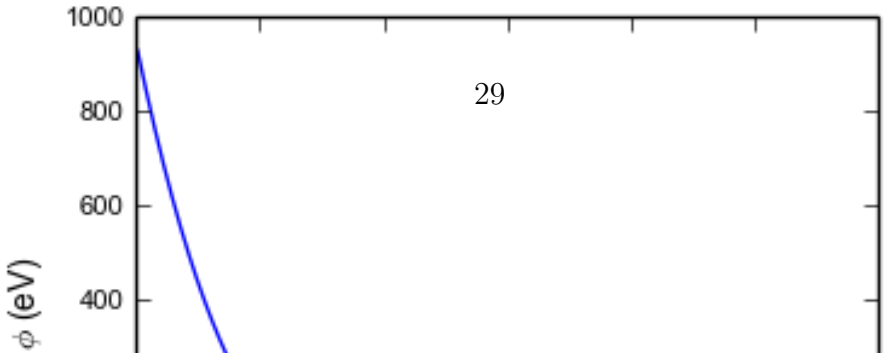
From the total energy expression we can straightforwardly derive forces, $\vec{f_k} = -\partial E/\partial \vec{r}_k$, leading to

$$\vec{f_k} = -\sum_i \frac{\partial \mathcal{F}(\rho_i)}{\partial \rho_i} \frac{\partial \rho_i}{\partial \vec{r}_k} - \frac{1}{2} \sum_{i,j} \frac{\partial V}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \vec{r}_{ij}} \tag{5.7}$$

$$= -\sum_i \frac{\partial \mathcal{F}(\rho_i)}{\partial \rho_i} \sum_j \frac{\partial f}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \vec{r}_k} - \frac{1}{2} \sum_{i,j} \frac{\partial V}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \vec{r}_k} \tag{5.8}$$


1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
F (eV)
0.0
0.5
1.0
1.5
2.0
ρ


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
f
0
1
2
3
4
5
6
r (Å)


1000
800
600
400
ϕ (eV)

Note that $\partial r_{ij}/\partial \vec{r}_k = \left(\delta_{ik} - \delta_{jk}\right) \hat{r}_{ij}$. Hence

$$\vec{f}_k = -\sum_i \frac{\partial \mathcal{F}\left(\rho_i\right)}{\partial \rho_i} \sum_j \frac{\partial f}{\partial r_{ij}} \left(\delta_{ik} - \delta_{jk}\right) \hat{r}_{ij} - \frac{1}{2} \sum_{i,j} \frac{\partial V}{\partial r_{ij}} \left(\delta_{ik} - \delta_{jk}\right) \hat{r}_{ij} \tag{5.9}$$

$$= -\sum_i \left(\frac{\partial \mathcal{F}\left(\rho_k\right)}{\partial \rho_k} \frac{\partial f}{\partial r_{ki}} \hat{r}_{ki} - \frac{\partial \mathcal{F}\left(\rho_i\right)}{\partial \rho_i} \frac{\partial f}{\partial r_{ik}} \hat{r}_{ik} \right) - \frac{1}{2} \sum_i \left(\frac{\partial V}{\partial r_{ki}} \hat{r}_{ki} - \frac{\partial V}{\partial r_{ik}} \hat{r}_{ik} \right) \tag{5.10}$$

Using $\hat{r}_{ik} = -\hat{r}_{ki}$ gives

$$\vec{f}_k = \sum_i \left(\frac{\partial \mathcal{F}\left(\rho_k\right)}{\partial \rho_k} + \frac{\partial \mathcal{F}\left(\rho_i\right)}{\partial \rho_i} \right) \frac{\partial f}{\partial r_{ik}} \hat{r}_{ik} + \sum_i \frac{\partial V}{\partial r_{ik}} \hat{r}_{ik} \tag{5.11}$$

Energies and forces are typically implemented analytically in a molecular dynamics code. Derivation (and correct implementation) of the force can be tedious for complicated potential expressions!

Chapter 6

Parallel computers and the Message Passing Interface

Context: This chapter sets the stage for discussing parallelization of the molecular dynamics simulation method introduced in the previous chapters. We first need to talk about parallel hardware architectures and how to program for them. The specific programming model that we will employ is known under the term *Single Program Multiple Data*. The Message Passing Interface (MPI) is a library that facilitates programming for massively parallel machines under this programming model.

6.1 Parallel hardware architectures

Parallel hardware has become ubiquitous over the past decade. Most central processing units (CPUs) in computers, phones or other hardware have multiple cores that can execute instructions in parallel. Massively parallel computing systems combine multiple CPUs into nodes that share a common memory. These nodes are then combined into the full compute system through a network interconnect.

Parallel architecture are often hierachical and have parallelization at different levels. Notable is vectorization at the core-level, share memory parallelization for multicore architectures and distributed memory parallelization for large computing systems that communicate via an interconnect (a network connection).

6.2 Scaling consideration

Software that runs on parallel computers needs to scale. Scaling describes how the time to returning the result changes as the number of available compute units (cores) changes. The simplest model for scaling assumes that our code can be divided into a fraction f_s that needs to be executed on a single core while a fraction f_p scales perfectly, i.e. its execute time is $\propto 1/p$ where p is the number of available processes or cores. (Note that $f_s + f_p = 1$ since they are fractions.) This leads to Amdahl's law that describes the speedup S as a function of p:

$$S = pf_p + f_s p \tag{6.1}$$

6.3 Programming model

The Message Passing Interface (MPI) is an application programming interface (API) for distributed memory parallelization. (A code parallelized with MPI also works on shared memory machines!) The programming model underlying MPI is called single program multiple data (SPMD): The identical program is executed multiple times but operates on different datums.

6.3.1 Example: Monte-Carlo estimate of the number π

As the simplest example of a parallelization, we consider a Monte-Carlo estimate of the number π.

Chapter 7

Domain decomposition

Context: Parallelization in molecular dynamics typically occurs through *domain decomposition*. The simulation domain is divided into subdomains, each of which runs within an MPI process. This distributes the workload among different compute units. Communications occurs only at the interface of the subdomain, either to exchange atoms between subdomains or to communicate *ghost atoms* that are required for the computation of correct forces in short-range interatomic potentials.

7.1 Simulation domain

Our atomic system has so far lived in an infinite space consisting of vaccum. We have made no reference to a simulation domain and the code developed up to Milestone 07 makes not reference to such a domain. We now introduce domain decomposition and for this need a simulation domain, i.e. a region of space Ω in which our atoms can reside. This domain can be periodic, which we will discuss in more detail in the next chapter.

We will assume that the simulation has its origin at $(0, 0, 0)$ and is spanned by three linearly independent vectors $\vec{a}_1$, $\vec{a}_2$ and $\vec{a}_3$. Any atomic position can then be expressed as

$$\vec{r}_i = s_{i,1}\vec{a}_1 + s_{i,2}\vec{a}_2 + s_3\vec{a}_{i,3} \tag{7.1}$$

with $s_\alpha \in [0, 1)$. s_α must remain in this interval since we do not allow atoms outside of the simulation domain. The vector $\vec{s}_i$ is the scaled position of the atom i. Using the *domain matrix* $\underline{A} = (\vec{a}_1, \vec{a}_2, \vec{a}_3)$, we can express this more compactly as $\vec{r}_i = \underline{A} \cdot \vec{s}_i$. Conversely, we obtain the scaled positions from $\vec{s}_i = \underline{A}^{-1} \cdot \vec{r}_i$.

In what follows, we assume rectilinear domains, i.e. $\vec{a}_1 = (L_x, 0, 0)$, $\vec{a}_2 = (0, L_y, 0)$ and $\vec{a}_3 = (0, 0, L_z)$ where L_x, L_y and L_z are the linear dimensions of the domain. The methods that are described in the following are straightforwardly extended to arbitrary (tilted) domains.

7.2 Decomposition into Cartesian domains

We decompose the full system into $N_x \times N_y \times N_z$ subdomains. For a rectilinear domain, this means teach subdomain has linear dimensions of L_x/N_x, L_y/N_y and L_z/N_z. Each subdomain propagates its own atoms. When atoms leave the subdomain, they are transferred to the respective neighboring domain. We call this process *atom exchange*.

Domain decomposition algorithms for MD simulations have started to appear in the literature around 1990. Some of the earliest references to this type of algorithm are Brugè and Fornili (1990); Liem et al. (1991); Chynoweth et al. (1991); Pinches et al. (1991); Brown et al. (1993); Plimpton (1995).

7.3 Ghost atoms

The atoms within each subdomain are not sufficient to compute the forces upon these atoms. In order to compute forces for atoms near the domain boundary, we need to transfer atoms that sit outside of the subdomain from the neighboring subdomains. These atoms are called ghost atoms. All atoms up to a distance r_{G} from the subdomain boundary are transferred. For a Lennard-Jones potential, $r_{\mathrm{G}} = r_c$ but for the EAM potential discussed here $r_{\mathrm{G}} = 2r_c$. This is because a force in the EAM potential is affected by an atom that sits twice the cutoff radius r_c away.

7.4 Communication pattern

The basic communication pattern involves two `MPI_Sendrecv` commands per Cartesian direction. The atoms that are send (either exchanged or as ghost atoms) must be serialized into a send buffer. Given that serialization has occured into the buffers `send_left` and `send_right`, the communication pattern looks as follows:

```
1 MPI_Sendrecv(&send_left, left_, &recv_right, right_, comm_)};
2 MPI_Sendrecv(&send_right, right_, &recv_left, left_, comm_)};
```

Here `comm_` contains the MPI communicator and `left_` and `right_` the MPI ranks of the processes that host the subdomain to the left and the

right, respectively, of the current subdomain. The buffers `recv_left` and `recv_right` hold the serialized atomic information received from the left and right, respectively. This information needs to be deserialized into the respective atom data type.

Bibliography

H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, 81(8):3684–3690, 1984. URL https://doi.org/10.1063/1.448118.

D. Brown, J. H. R. Clarke, M. Okuda, and T. Yamazaki. A domain decomposition parallelization strategy for molecular dynamics simulations on distributed memory machines. *Comput. Phys. Comm.*, 74(1):67–80, 1993. URL https://doi.org/10.1016/0010-4655(93)90107-N.

F. Brugè and S. L. Fornili. Concurrent molecular dynamics simulation of spinodal phase transition on transputer arrays. *Comput. Phys. Comm.*, 60(1): 31–38, 1990. URL https://doi.org/10.1016/0010-4655(90)90076-D.

S. Chynoweth, U. C. Klomp, and L. E. Scales. Simulation of organic liquids using pseudo-pairwise interatomic forces on a toroidal transputer array. *Comput. Phys. Comm.*, 62(2):297–306, 1991. URL https://doi.org/10.1016/0010-4655(91)90102-Q.

F. Cleri and V. Rosato. Tight-binding potentials for transition metals and alloys. *Phys. Rev. B*, 48(1):22–33, 1993. URL https://doi.org/10.1103/PhysRevB.48.22.

M. S. Daw and M. I. Baskes. Semiempirical, quantum mechanical calculation of hydrogen embrittlement in metals. *Phys. Rev. Lett.*, 50(17):1285–1288, 1983. URL https://doi.org/10.1103/PhysRevLett.50.1285.

F. Ercolessi and J. B. Adams. Interatomic Potentials from First-Principles Calculations: The Force-Matching Method. *EPL*, 26(8):583–588, 1994. URL https://doi.org/10.1209/0295-5075/26/8/005.

J. Ferrante, J. Smith, and J. Rose. Diatomic Molecules and Metallic Adhesion, Cohesion, and Chemisorption: A Single Binding-Energy Relation. *Phys. Rev. Lett.*, 50(18):1385–1386, 1983. doi: 10.1103/PhysRevLett.50.1385. URL http://www.ncbi.nlm.nih.gov/pubmed/23357448.

M. W. Finnis and J. E. Sinclair. A simple empirical N-body potential for transition metals. *Phil. Mag. A*, 50(1):45–55, 1984. URL https://doi.org/10.1080/01418618408244210.

S. M. Foiles, M. I. Baskes, and M. S. Daw. Embedded-atom-method functions for the fcc metals Cu, Ag, Au, Ni, Pd, Pt, and their alloys. *Phys. Rev. B*, 33(12):7983–7991, 1986. URL https://doi.org/10.1103/PhysRevB.33.7983.

G. Grochola, S. P. Russo, and I. K. Snook. On fitting a gold embedded atom method potential using the force matching method. *J. Chem. Phys.*, 123 (20):204719, 2005. URL https://doi.org/10.1063/1.2124667.

R. P. Gupta. Lattice relaxation at a metal surface. *Phys. Rev. B*, 23(12): 6265–6270, 1981. URL https://doi.org/10.1103/PhysRevB.23.6265.

S. Y. Liem, D. Brown, and J. H. R. Clarke. Molecular dynamics simulations on distributed memory machines. *Comput. Phys. Comm.*, 67(2):261–267, 1991. URL https://doi.org/10.1016/0010-4655(91)90021-C.

M. R. S. Pinches, D. J. Tildesley, and W. Smith. Large scale molecular dynamics on parallel computers using the link-cell algorithm. *Molecular Simulation*, 6(1-3):51–87, 1991. URL https://doi.org/10.1080/08927029108022139.

S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.*, 117(1):1–19, 1995. URL https://doi.org/10.1006/jcph.1995.1039.

J. Rose, J. Smith, F. Guinea, and J. Ferrante. Universal features of the equation of state of metals. *Phys. Rev. B*, 29(6):2963–2969, 1984. URL https://doi.org/10.1103/PhysRevB.29.2963.

S. V. Sukhomlinov and M. H. Müser. Constraints on phase stability, defect energies, and elastic constants of metals described by EAM-type potentials. *J. Phys.: Condens. Matter*, 28(39):395701, 2016. URL https://doi.org/10.1088/0953-8984/28/39/395701.

V. Vitek. Pair potentials in atomistic computer simulations. *MRS Bull.*, 21 (2):20–23, 1996. URL https://doi.org/10.1557/S088376940004625X.