universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"Training a Neural Network Potential for Molecular Dynamics Simulations of CO_2 with Ab Initio Precision"

verfasst von / submitted by

Matthias Kiss, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2019 / Vienna, 2019

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 876
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Physik
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Christoph Dellago

Acknowledgements

I want to thank my thesis advisor Professor Christoph Dellago for the opportunity to write my thesis in the department of Computational Physics at the University of Vienna. He was always there with good advice if I had questions. I also want to thank Andreas Singraber whose Neural Network Potential Package 'n2p2' I used for this thesis and who spent many hours helping me out when needed. It was also a great pleasure to share an office with my fellow master students Nils Clees and Jakob Michl and to be able to bounce ideas off them.

Abstract

In this thesis a neural network (NN) is trained to reproduce the potential energy surface (PES) of systems of CO_2 molecules using the Neural Network Potential Package (n2p2).
This approach has previously been shown to be successful when applied to systems of H_2O and Cu_2S molecules [1, 2].
The process is started by creating configurations of CO_2 molecular crystals corresponding to the solid phases I to IV as well as VII. Then the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [3] together with a flexible version of the EPM2 empirical potential [4, 5] are used to generate trajectories, starting at the previously created configurations, along multiple isotherms with varying pressure. This creates many different configurations with as many different energies in various liquid and solid states. Afterwards from those configurations a small but diverse sub-sample of a few hundred configurations is chosen. The total energy of the configurations and the forces acting on each of the atoms in each configuration in this sub-sample is then recalculated with the Vienna Ab initio Simulation Package (VASP) [6, 7, 8, 9, 10] using the PBE functional [11]. This yields the necessary data to train the NN with ab-initio precision. The reason to first calculate trajectories with LAMMPS and then to recalculate a part of the resulting configurations with VASP, instead of doing all calculations with VASP to begin with, is the large computational cost of ab initio calculations compared to methods using empirical potentials like the EPM2.
Once the training data are created and the NN is trained, a comparison between molecular dynamics (MD) simulations done with the n2p2 and with VASP is made. These tests focus on the radial distribution function (RDF) and the velocity auto correlation function (VACF) of CO_2 at different state points. It can be shown that the NNP is able to reproduce the RDF and VACF of the DFT-MD simulations, however further improvements are possible.

Zusammenfassung

Im Zuge dieser Arbeit wird ein neuronales Netzwerk (NN) trainiert, um die potentielle-Energie-Oberfläche (PES) eines Systems bestehend aus CO_2 Molekülen, mithilfe des Neural Network Potential Package (n2p2) zu rekreieren.
Diese Herangehensweise hat sich schon zuvor als erfolgreich erwiesen, bei der Anwendung auf Systeme bestehend aus H_2O Molekülen [1].
Der Prozess beginnt damit Konfigurationen von CO_2 Molekülkristallen zu erstellen, die den festen Phasen I bis IV, sowie VII entsprechen. Anschließend wird der Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [3] zusammen mit der flexiblen Variante des empirischen Potentials EPM2 [4, 5] verwendet, um Trajektorien, beginnend bei den vorher

erzeugten Konfigurationen, entlang mehrerer Isothermen mit variierendem Druck, zu erzeugen. So werden viele verschiedene Konfigurationen mit ebenso vielen verschiedenen Energien, in sowohl flüssiger, als auch fester Phase, erzeugt. Danach wird ein kleiner aber diverser Teil, bestehend aus mehreren hundert Konfigurationen, aus allen verfügbaren Konfigurationen ausgewählt. Die Gesamtenergie und Kräfte für alle Atome in allen ausgewählten Konfigurationen werden dann mithilfe des Vienna Ab-initio Simulation Package (VASP) [6, 7, 8, 9, 10] unter Verwendung des PBE Funktionals [11], neu berechnet. Die daraus resultierenden Daten werden benötigt, um das NN mit ab-initio Präzision trainieren zu können. Der Grund dafür, zuerst Trajektorien mit LAMMPS zu berechnen und dann einen Teil davon mit VASP neu zu berechnen liegt in dem großen rechnerischen Aufwand von ab-initio-Methoden wie VASP, im Vergleich zu Methoden die empirische Potentiale verwenden wie LAMMPS.
Nachdem die Trainingsdaten erzeugt und das NN trainiert wurde wird ein Vergleich zwischen Molekulardynamik (MD) Simulationen gemacht, die sowohl mit VASP, als auch mit n2p2 durchgeführt werden. Dieser Vergleich konzentriert sich auf die radiale Verteilungsfunktion (RDF) und die Geschwindigkeitautokorrelation (VACF) von CO_2 Systemen in verschiedenen Zuständen. Es kann gezeigt werden, dass das NNP in der Lage ist die RDF und VACF der DFT-MD Simulation zu reproduzieren, jedoch sind weitere Verbesserungen möglich.

Contents

1 Introduction

The purpose of this work is to train a neural network potential (NNP) in order to reproduce the potential energy surface of different systems of molecular CO_2 with ab-initio precision while only using a fraction of the computational cost of pure ab-initio methods. The same approach was previously successfully employed for H_2O [1] and CU_2S [2] using the neural network potential package (n2p2). The n2p2 software package will also be used in this thesis for the training of the NNP.

One reason why CO_2 was chosen to be simulated by the NNP in this thesis is the potential future prospect of combining it with the NNP for H_2O and being able to do simulations of H_2O-CO_2 mixtures. Such mixtures could be of interest due to the ability to form clathrate hydrates, structures were gas molecules are bound in a cage of crystalline H_2O. It has been discovered that there are vast deposits of CH_4 bound in such clathrate hydrate structures in oceanic and permafrost environments [12]. The CH_4 could possibly be replaced by CO_2 in those structures, thereby permanently storing the CO_2 [13, 14]. This could be used to reduce atmospheric levels of CO_2, whose steady increase is a driving force behind man made climate change. The study of such H_2O-CO_2 mixtures using NNPs with ab-initio precision on unprecedented timescales and system sizes could provide further insight on this topic.

Machine learning (ML) techniques such as Neural Networks (NN), support vector machines [15] and Gaussian processes [16] have been used in physics and chemistry for the last two and a half decades [17, 18]. This work will only use the ML method of NNs. Today applications for NNs can be found over a wide area of topics, even in everyday life. The uses of NNs reach from shape detection up to face recognition [19] as well as material science [20] or even, as will be the case in this thesis, for MD simulations. However the use of NNs for predicting the forces and energies of systems of atoms to be used in MD simulations is a relatively new application. It was introduced in the form that is relevant for this thesis by Behler and Parrinello in 2007 [21]. Their approach of using NNPs to fit the potential energy surface of density functional theory (DFT) simulations is the foundation of the n2p2 software package that was used in this thesis for the training of the NNP for CO_2. The training of the NNP is only one of multiple functionalities of n2p2. It can also use an already trained NNP to make predictions about the forces and energies of atomic configurations it has never seen before. Through a plugin for the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [3] this capability can be used to do complex MD simulations using the NNP the same way one would use an empirical potential, but with the precision of a costly ab-initio simulation.

The following thesis will first give a short introduction on the structure of the high pressure phases of CO_2, which will be the main focus of the training of the NNP (Chapter2). This is followed by the fundamental principles of MD simulations needed to not only to create suitable atomic configurations for the training of the NNP but also to later verify the results

of the training (Chapter 3). The next chapter (4) will discuss how to calculate the inter atomic and inter molecular forces needed to do MD simulations. This includes descriptions of empirical potentials, density functional theory and the neural network potential. This is followed by an in depth discussion of the training process of a NNP with ab-initio precision using n2p2 (Chapter 5). This is followed by Chapter 6 which will compare MD simulations using the trained NNP with MD simulations using the density functional theory approach the NNP is based on. Finally there will be concluding remarks in Chapter 7.

2 Carbon dioxide (CO_2)

CO_2 is a very important building block of our planet's atmosphere. It is a byproduct of the burning of organic matter and the respiration of animals and enables plants to do photosynthesis and produce oxygen. CO_2 also plays an important role in how our atmosphere absorbs heat and might be a key component in the recent heating of the planet's climate.
The study of CO_2 could be beneficial in finding possible forms of geological storage of CO_2 within the ground [22], thereby decreasing its atmospheric concentration.
Under high pressure conditions CO_2 displays a rich phase behavior, with five known molecular crystals (phases I, II, III, IV and VII described in this chapter), 2 non molecular crystals (phase V [23] and phase VI [24]), as well as amorphous CO_2 [25]. The focus of this chapter lies on the five molecular phases due to the importance of crystal structure on the MD simulations that will be used during the training process of the neural network (NN).

2.1 Physical properties of CO_2

CO_2 is a linear molecule consisting of two oxygen atoms and one carbon atom with an intra molecular angle of 180° and a bond length of 1.16 Å under standard conditions for temperature and pressure as pictured in figure 2.1. Under these conditions CO_2 is found in its gas phase, although CO_2 has a rich phase diagram including multiple high pressures molecular and non-molecular solid phases, as seen in figure 2.2. This work focuses on the molecular solid phases as well as the fluid phase.

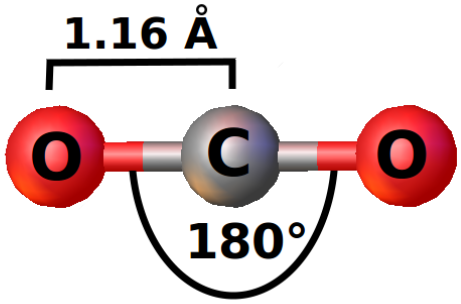

Figure 2.1: The CO_2 molecule consists out of two oxygen and one carbon atom. The CO-bond length is 1.16 Å and the OCO-angle is 180°.

Due to its linearity the dipole moment of CO_2 vanishes and the quadrupole moment becomes the dominant moment. This is relevant for designing empirical force fields to describe CO_2 in molecular dynamics (MD) simulations as discussed in Chapter 3.

2.2 Phasediagram of CO2

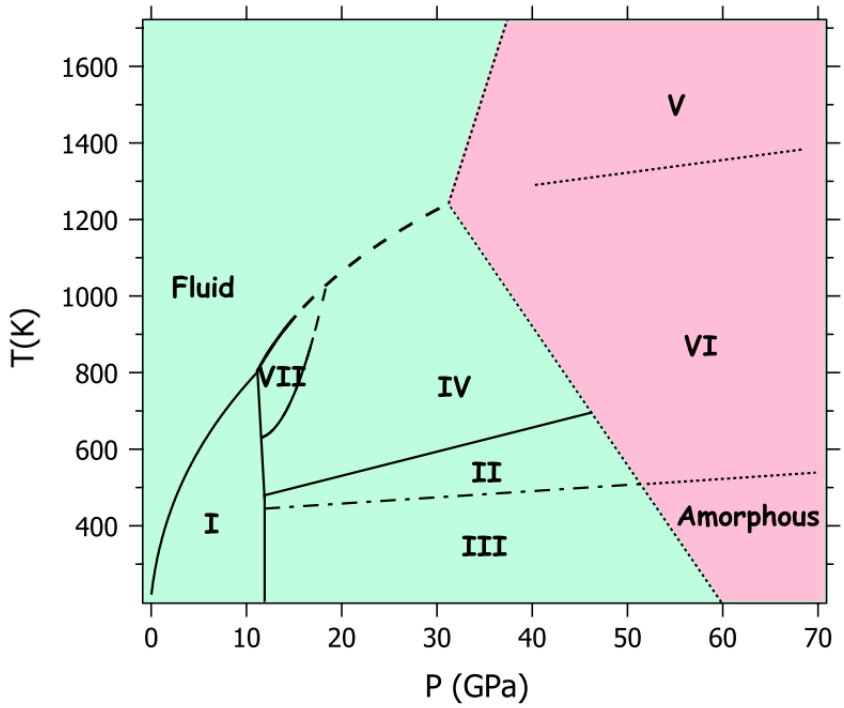

Figure 2.2: Phase diagram of CO_2 as a function of pressure (P) and temperature (T). The roman numerals I to VII denote the stable regions of the known solid phases. In the green area CO_2 is present in molecular form, while in the red area the solids are formed from non-molecular CO_2. Figure taken from [26].

For the training of the NN (see Chapter 5) configurations of liquid and solid CO_2 are used. The configurations are chosen from trajectories of molecular dynamics (MD) simulations over the range of the molecular phases in the phase diagram seen in figure 2.2 (green area). More information about these simulations can be found in Chapter 5.
The most straightforward approach to run such an MD simulation on solid CO_2 is for the first configuration of the simulation to already be of the correct crystal structure. During the first steps of the simulation this structure is then relaxed according to the specific settings of temperature and pressure of each individual simulation.
Therefore it is necessary to explore the structure of the molecular crystals of the CO_2 phases I, II, III, IV and VII so that configurations of each structure can be created and subsequently used for MD runs.

2.2.1 Phase I - Dry Ice

CO_2 phase I, commonly referred to as dry ice, is a molecular crystal that starts to form at low pressures and temperatures, shown in figure 2.2 in the green area marked with a "I". Phase I corresponds to the cubic Pa3 space group with 4 molecules per unit cell and was first described by Olinger in 1982 [28] using powder x-ray diffraction and was the first time a solid phase of CO_2 was described. For the starting configuration of the MD simulations in this thesis a unit cell with a=4.939 Å, measured at a pressure of 11.8 GPa, was used [29]. The CO bond length is slightly elongated at that pressure and has a value of r=1.172 Å [27].

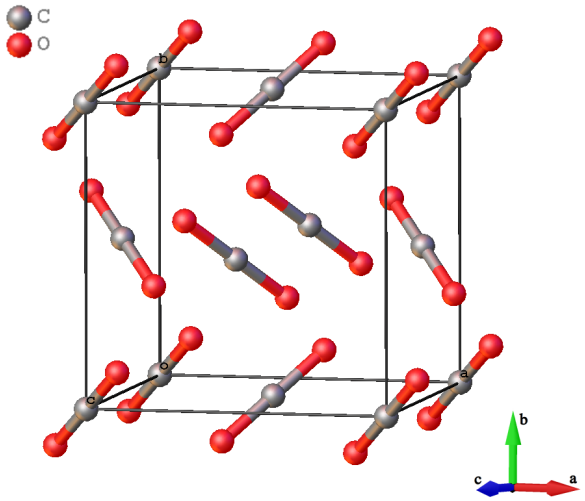

Figure 2.3: One cell of CO_2 phase I that belongs to the cubic Pa3 spacegroup, with a=4.939 Å and a CO bond length r=1.172 Å at p=11.8 GPa. All cell parameters are taken from [27].

Figure 2.3 shows a cell containing 14 molecules built according to these parameters. A cell of 32 molecules (2x2x2 unit cells) was used to initialize the MD simulations to generate training configurations for the NN.

2.2.2 Phase II

The existence of a second solid phase of CO_2, titled "dry ice II", was first found by Liu [30] in 1983. Subsequent studies found this phase to be of the tetragonal space group $P4_2mnm$ [31, 32] containing 2 molecules in the unit cell. This phase can be found at high pressures and intermediate temperatures as seen in figure 2.2 in the area denoted by a "II".

To build a starting point for the later MD simulations the parameters a=b=3.516 Å,

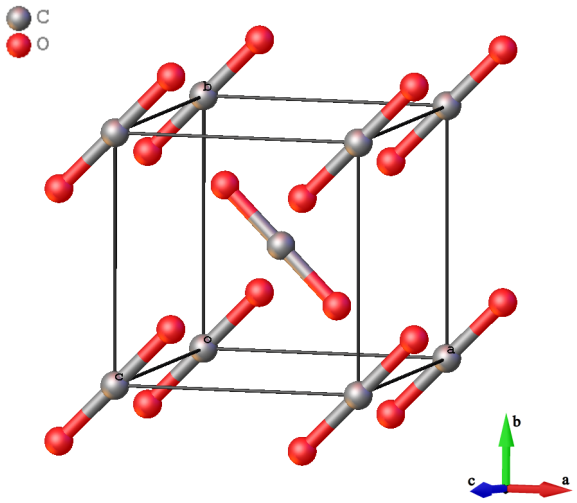

Figure 2.4: Crystal structure of CO_2-II, also named dry ice II. The cell is of the orthorombic $P4_2mnm$ spacegroup, the parameters for which are a=b=3.516 Å, c=4.104 Å, CO bond length r=1.14 Å at p=25.8 GPa and T=295 K according to [32].

c=4.104 Å were used to describe the cell of CO_2-II. The bond length of r=1.14 Å is described to decrease under increasing pressure in this phase according to experimental results from Datchi et. al. [32] which is in agreement with DFT calulations according to the same author. Those parameters were taken from x-ray powder diffraction measurements at a pressure and temperature of 25.8 GPa and 295 K, respectively [32]. In figure 2.4 the structure of the CO_2-II crystal can be seen modeled with the previously mentioned parameters. In order to generate the training configurations for the NN, MD simulations with 54 molecules (3x3x3 unit cells) were performed.

2.2.3 Phase III

In 1983 yet another new solid phase of CO_2 was found by Hanson [33]. This phase was determined to be an orthorombic Cmca structure with 4 molecules in the unit cell [29]. Phase III seems to be structurally closely related to phase I and can be obtained from low temperature compression of phase I, see figure 2.2 for the phase diagram. It is also thought to be a meta stable phase with phase II being of lower free energy [27]. There is a large hysteresis in the phase transitions involving phase III though which makes it difficult to find exact boundaries for this phase.
The x-ray diffraction experiment at 11.8 GPa on phase III [29] yielded the parameters for the Cmca cell that were used in this work: a=4.330 Å, b=4.657 Å, c=5.963 Å with an angle of ϕ=52° in the bc-plane. The CO-bond length in this phase is slightly larger than the one of phase I. For a pressure of 11.8 GPa a bond length of r=1.173 Å was found [27]. Figure 2.5 a cell can be seen representing the Cmca structure of phase III using the cell parameters just shown. The MD simulation for the training configurations was performed with a crystal of 32 molecules (2x2x2 unit cells), the same as with phase I.

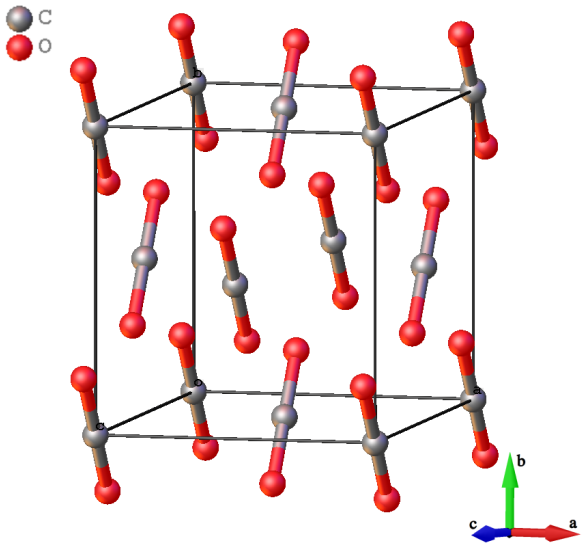

Figure 2.5: Orthorombic Cmca structure of CO_2 phase III. This structure is very similar to the structure of phase I but with an additional 52° angle of the CO_2 molecules relative to the bc-plane. The cell parameters are a=4.330 Å, b=4.657 Å, c=5.963 Å and bond length r=1.173 Å measured at p=11.8 GPa. Those parameters can be found in [29]

2.2.4 Phase IV

The solid phase CO_2-IV was first found and described by Yoo [34] in 2000. A previous study that assumed to have found CO_2-IV [35] was shown by Yoo to have been mistaken and to describe the already known phase CO2-III. Interestingly at first it was assumed that phase IV was an "intermediate bonding state" that would connect the molecular phases I and III to the non-molecular phase V. The CO_2 molecules in this intermediate bonding state were assumed to have a strongly elongated CO-bond as well as being non-linear.
Later Datchi [26] could produce single crystals of CO_2-IV and used x-ray diffraction to show that phase IV is indeed a molecular phase with bond angle of 180° and a slightly decreased CO-bond length. That work also produced the phase diagram of CO_2 seen in figure 2.2 and showed that phase IV crystallizes in the rhombohedral $R\bar{3}c$ space group with 24 molecules in the unit cell, pictured in figure 2.6. The cell parameters found in the experiment performed at a pressure of 15.2 GPa are: a=b=8.628 Å, c=10.604 Å with a CO bond length of r=1.155 Å. For the MD simulation that was later performed to acquire training configurations for the NN one unit cell with 24 molecules was used. Using more than one unit cell for this structure would have been too costly when doing DFT calculations later on.

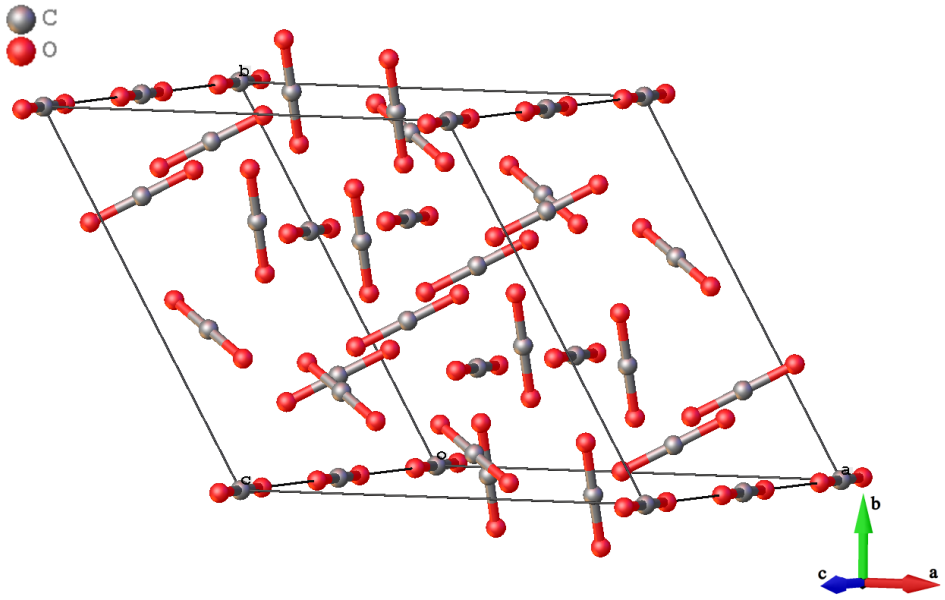

Figure 2.6: The rhombohedral $R\bar{3}c$ spacegroup was assigned to CO_2 phase IV by Datchi [26]. The parameters used to generate the displayed cell a=b=8.628 Å, c=10.604 Å, CO bond length r=1.155 Å, measured at p=15.2 GPa were taken from [26].

2.2.5 Phase VII

Phase VII of CO_2 was found by Datchi in 2006 [36] and described as belonging to the orthorombic Cmca spacegroup, with 4 molecules in the unit cell. This is the same structure as the previously discovered phase III, see figure 2.5. Interestingly phase VII and III are being found in isolated and unconnected areas in the phase diagram of CO_2, which can be seen in figure 2.2.
There are slight differences between phase VII and III though. The angle of the molecules in the bc-plane is ϕ=54.5° in phase VII as opposed to ϕ=52° in phase III. The cell parameters

found for phase VII, measured at 12.1 GPa, are a=4.313 Å, b=4.746 Å and c=5.948 Å [36]. These parameters were used for a 32 molecule cell (2x2x2 unit cells) for the MD simulations performed to acquire configurations for the NN training.
There is also a theoretical study that suggests that phase III and VII are in fact a completely identical phase [37], but for this thesis the experimental findings of Datchi were used.

2.2.6 Liquid and Gas Phase

The NN is also trained using liquid CO_2 configurations. The liquid configurations are produced by taking a configuration of solid CO_2, melting it in one simulation and then adjusting pressure and temperature to the required range. Because the high pressure regime was of the highest interest in this thesis, mostly configurations close to the border of the solid molecular phases were chosen for liquid CO_2.

3 Molecular Dynamics

Molecular dynamics (MD) simulations are a method used to calculate the trajectory of atoms and molecules based on the forces that act on each atom/molecule. This method can be used to visualize microscopic processes like the crystallization of a fluid or the melting of a crystal, but the simulated processes can be as complex as the folding of a protein. Such processes can otherwise only be researched experimentally by using x-ray diffraction and similar methods which are limited in their applicability. MD simulations also let us make connections between those microscopic processes and structural and dynamic properties such as the radial distribution function or the diffusion coefficient of a liquid.
In this chapter it will be described how to numerically integrate the equations of motion for systems of many bodies, as well as how to treat thermodynamics (TD) in that case. The explanations will follow the books *Understanding Molecular Dynamics* by Frenkel and Smit [38] as well as *Computer Simulation of Liquids* by Allen and Tildesley [39], *The Art of Molecular Dynamics Simulation* by D.C. Rapaport [40] and *Statistical Mechanics: Theory and Molecular Simulation* by Mark E. Tuckerman [41].
In Chapter 4 the second fundamental problem of MD simulations will be discussed: calculating the inter atomic forces needed for solving the equations of motion. There are several approaches to find the relevant forces that directly influence the accuracy of any MD simulation, such as empirical potentials and ab-initio methods. However the framework of MD simulations as discussed in this chapter is independent of how the forces are calculated.

3.1 Equations of motion

As stated in the introduction, the goal of MD simulations is to numerically solve Newton's equations of motion. The Hamiltonian $\mathcal{H}$ of a system is defined as,

$$\mathcal{H}(\mathbf{q}, \mathbf{p}) = \mathcal{V}(\mathbf{q}) + \mathcal{K}(\mathbf{p}). \tag{3.1}$$

The Hamiltonian depends on the set of general coordinates $\mathbf{q}_i$, which will be the Cartesian coordinates of each particle i in the simulation,

$$\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N), \tag{3.2}$$

as well as the set of momenta $\mathbf{p}_i$ for each particle i,

$$\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N). \tag{3.3}$$

The Hamiltonian is the sum of the potential energy $\mathcal{V}$ and the kinetic energy $\mathcal{K}$ of a system. The kinetic energy can be written as:

$$\mathcal{K} = \sum_{i=1}^{N} \sum_{\alpha} p_{i\alpha}^2 / 2m_i, \tag{3.4}$$

where m_i is the mass for particle i and $p_{i\alpha}$ are the components α of the momentum $\mathbf{p}$ of particle i.
The potential energy $\mathcal{V}$ depends on the specific system that is considered and the force $\mathbf{F}_i$ is derived via the gradient of the potential energy $\mathbf{F}_i = -\nabla_i \mathcal{V}$. Multiple ways to derive a suited potential are presented in Chapter 4. For now the potential and forces will be treated most generally.
The equations of motion in the Lagrange formulation are based on the fundamental principle of minimizing the action that governs many aspects of how nature functions. This principle can be described using the Euler-Lagrange equation:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}_k} - \frac{\partial \mathcal{L}}{\partial \mathbf{q}_k} = 0, \tag{3.5}$$

using the Lagrangian function $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$,

$$\mathcal{L} = \mathcal{K} - \mathcal{V}. \tag{3.6}$$

In MD usually Cartesian coordinates $\mathbf{r}_i$ are used to describe the position of atom i in a system of atoms. This leads from equation 3.5 to Newton's equation of motion:

$$m_i \ddot{\mathbf{r}}_i = \mathbf{F}_i. \tag{3.7}$$

The generalized momentum $\mathbf{p}_k$ is defined as the derivative of the Lagrangian in regard of the time derivative of the generalized coordinate $\dot{\mathbf{q}}_k$,

$$\mathbf{p}_k = \frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}_k}. \tag{3.8}$$

The Hamiltonian form of the equations of motion then can be written as:

$$\dot{\mathbf{q}}_k = \frac{\partial \mathcal{H}}{\partial \mathbf{p}_k} \qquad \text{and} \qquad \dot{\mathbf{p}}_k = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}_k}. \tag{3.9}$$

This leads to the general definition of the Hamiltonian,

$$\mathcal{H} = \sum_k \dot{\mathbf{q}}_k \mathbf{p}_k - \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}). \tag{3.10}$$

In the case of MD and as described in Chapter 4 the potential $\mathcal{V}$ is independent of velocities of the atoms and does not change in time. Therefore the Hamiltonian is reduced to equation 3.1 which is equal to the total energy and if Cartesian coordinates $\mathbf{r}_i$ are used again Hamilton's equations of motion are,

$$\dot{\mathbf{r}}_i = \frac{\mathbf{p}_i}{m_i} \qquad \text{and} \qquad \dot{\mathbf{p}}_i = -\nabla_{r_i} \mathcal{V} = \mathbf{F}_i, \tag{3.11}$$

which is equivalent to Newton's equation of motion (see equation 3.7).
This means that in order to solve the equations of motion for a system with N atoms either a system of $3N$ second order differential equations (equation 3.7) or a system of $6N$ first order differential equations (equation 3.11) has to be solved.
In the next section methods will be presented to solve such systems of differential equations numerically, which is the fundamental problem of MD simulations.

3.2 Integration of the equations of motion

The basic idea behind solving the equations of motion (see equations 3.7 and 3.11) for large systems of atoms as are present in MD simulations is to use finite difference approaches.
This means that the equations of motion are first discretized and then solved for a small time step Δt using the current positions and velocities of the atoms at time t as well as the forces acting on each atom. The new positions at time $t + \Delta t$ are then used to update the forces acting on the now slightly different system. Together with the updated velocities the updated forces are then used to calculate the new positions and velocities after another small time step Δt.
A very commonly used integration scheme is the velocity Verlet algorithm [42]. This integration scheme is also available in LAMMPS [3], a software package for MD simulations that was used in this thesis.
The velocity verlet algorithm has a few key features that make it so suited for the field of MD. It does explicitly evolve positions as well as velocities, as opposed to some algorithms that only evolve the positions and then derive the velocities from those. It is time-reversible, which is a fundamental property of Hamilton's equations of motion that has to be preserved by any integration scheme. It is symplectic, which means that it preserves phase-space volume. This last property is important because even though the numerical integration does not exactly reproduce the Hamiltonian of the system it does reproduce a so called shadow Hamiltonian that is close to the actual Hamiltonian at all times. This means that there is no energy drift like with the naive Euler algorithm for example, because the energy of the shadow Hamiltonian is conserved.
The derivation of the velocity Verlet algorithm starts with the Taylor series expansion of the position $\mathbf{r}_i$ of an atom i at time $t + \Delta t$ with respect to its position at time t up to terms of second order in Δt:

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \dot{\mathbf{r}}(t)\Delta t + \ddot{\mathbf{r}}_i(t)\frac{\Delta t^2}{2} + \mathcal{O}(\Delta t^3), \tag{3.12}$$

where $\dot{\mathbf{r}}_i = \mathbf{v}_i$ and $\ddot{\mathbf{r}}_i = \mathbf{F}/m_i$ can be used and $\mathbf{v}_i$ is the velocity of atom i. This leads to,

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \mathbf{F}_i(t)\frac{\Delta t^2}{2m_i} + \mathcal{O}(\Delta t^3). \tag{3.13}$$

This is the first part of the velocity Verlet algorithm that yields the position of atom i after the time Δt has passed depending on the position $\mathbf{r}_i$, velocity $\mathbf{v}_i$ and force $\mathbf{F}_i$ of atom i at time t. In order to get the time evolution of the velocity one starts by doing the backwards Taylor expansion starting at time $t + \Delta t$ and expanding back to time t:

$$\mathbf{r}_i(t) = \mathbf{r}_i(t + \Delta t) + \mathbf{v}_i(t + \Delta t)\Delta t + \mathbf{F}_i(t + \Delta t)\frac{\Delta t^2}{2m_i} + \mathcal{O}(\Delta t^3). \tag{3.14}$$

The next step is to substitute the term $\mathbf{r}_i(t + \Delta t)$ in equation 3.14 with the right hand side of equation 3.13 and then solve for $\mathbf{v}_i(t + \Delta t)$:

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \frac{\Delta t}{2m_i}[\mathbf{F}_i(t) + \mathbf{F}_i(t + \Delta t)]. \tag{3.15}$$

Using the two equations 3.13 and 3.15 one can now evolve the position and velocity of atom i. This is the velocity Verlet algorithm. Equation 3.15 can only work if the force is independent of the velocity, which fortunately is the case for the MD simulations relevant for this thesis. This algorithm has one additional advantage compared to simpler algorithms, which is that it is self starting and therefore can be applied to any system for which positions and velocities of the atoms are known for any one time t.

3.3 Periodic boundaries

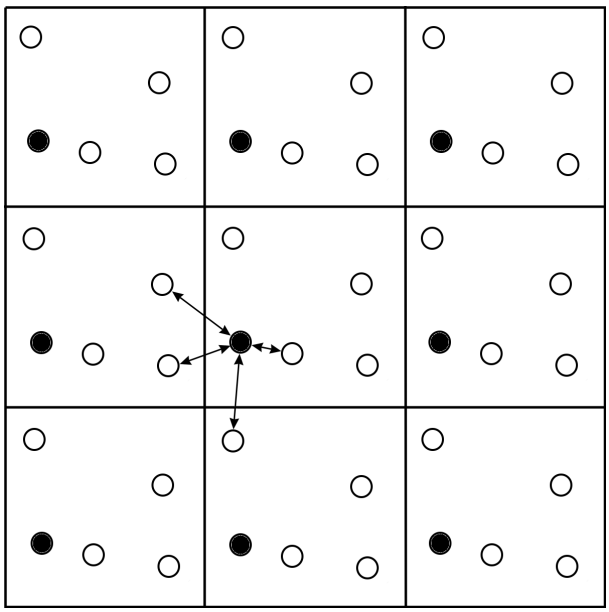

Figure 3.1: The minimum image convention for periodic boundary conditions. The original simulation box is in the center, containing five particles. The surrounding boxes are copies of the central box. Each atom i from the center box now interacts with the closest copy of each atom j. In this example the black filled atom (at the center of the four arrows) in the center box interacts only with one other atom that is in its own box. The other interactions are with the closest copy (nearest image) of its other neighbors which happen to be closer than the original. This way every atom behaves as if it would be in the middle of a bulk of atoms and surface effects can be avoided.

An important consideration regards the boundaries of the system that is simulated. Usually any given MD simulation takes place in some sort of box that does not have to be orthorombic but can be triclinic as well. A problem that arises from this approach is that towards the edge of the box surface effects can occur. Normally when trying to do a simulation on the bulk of atoms or molecules surface effects should be avoided.
A solution to this problem is offered by periodic boundary conditions. Periodic boundary conditions imply that copies of the simulation box are placed around the actual simulation box, as sketched in figure 3.1. Additionally the minimum image convention is used. This means that each atom i in the main simulation box interacts with the nearest periodic image of atom j. This also means that if atom i would leave the original simulation box through any side the copy of atom i from the image on the opposite side of the box moves into the original simulation box, leading to a constant number of particles in the restricted area of the box.
The distance vector $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ of atom j to atom i according to the minimum image convention for periodic boundary conditions for any triclinic box with arbitrary box matrix $\mathbf{h}$ (matrix containing the vectors that build the box) can be written as:

$$\begin{aligned} \mathbf{s}_i &= \mathbf{h}^{-1}\mathbf{r}_i, \\ \mathbf{s}_{ij} &= \mathbf{s}_i - \mathbf{s}_j, \\ \mathbf{s}_{ij} &\leftarrow \mathbf{s}_{ij} - \mathrm{NINT}(\mathbf{s}_{ij}) \quad \text{Minimum image convention}, \\ \mathbf{r}_{ij} &= \mathbf{h}\mathbf{s}_{ij}, \end{aligned} \tag{3.16}$$

here NINT(x) represents the nearest integer function. This algorithm first transforms the coordinates $\mathbf{r}_i$ into scaled coordinates, then calculates the distance $\mathbf{s}_{ij}$ in scaled coordinates in line 2. In the third line the actual minimum image convention is applied. The last line transforms back to the unscaled distance.
This algorithm can be used to find which image of an atom j is closest to atom i.

3.4 Averages

Generally when evaluating the results of an MD simulation one has to consider observables that can be calculated using averages. An MD simulation only approximates the solution of the equations of motions for each small time step. This means that the simulated trajectory and the real trajectory of the system in phase space diverge exponentially. This can be expressed through the Lyuapunov exponent that describes the rate at which the trajectories of two close points in phase diverge over time.
Therefore it is not useful to look at the trajectory of the positions and velocities of all atoms of one MD simulation, but one should consider values that are derived from averages. For example the average pressure and density of a simulation with a set temperature, volume and atom count are valid observables.
Therefore the time average is important, which depends on the value of the observable $A(r^N(t), p^N(t))$ at every single time step of the simulation.

$$\bar{A} = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^{\infty} A(r^N(t), p^N(t))\ dt. \tag{3.17}$$

Here $\bar{A}$ is the time average of the observable $A(r^N(t), p^N(t))$ and τ is the total duration of the simulation. Although theoretically one should have infinitely long trajectories for computing the time average, in reality simulations of finite length are commonly used.
The ensemble average is another way to calculate the average value of an observable. It assumes that given the ensemble's probability density $f(r^N, p^N)$ to observe one specific micro state one can calculate the average value of the observable $A(r^N(t), p^N(t))$,

$$\langle A \rangle = \frac{\int f(r^N, p^N)\ A(r^N(t), p^N(t))\ dr^N dp^N}{\int f(r^N, p^N)\ dr^N dp^N}. \tag{3.18}$$

The ensemble average and the time average are equivalent, if the system is ergodic,

$$\langle A \rangle = \bar{A}, \tag{3.19}$$

which means that if given unlimited time the system's trajectory will reach each possible micro state in phase space. The time the system spends in each area of phase space is proportional to the phase space volume of that area.
Usually ergodicity is assumed in MD simulations. Therefore in most MD simulations it is possible to calculate the time average of an observable, which is comparably simple, and not the ensemble average.

3.5 Velocity autocorrelation function/ radial distribution function

Later in this thesis (see Chapter 6) two quantities of a system of molecules will be calculated: the velocity autocorrelation function (VACF) and the radial distribution function (RDF). Using

these two quantities a comparison between the two different potentials for CO_2 (DFT and NNP, see Chapter 4 and 5) will be made.

3.5.1 VACF

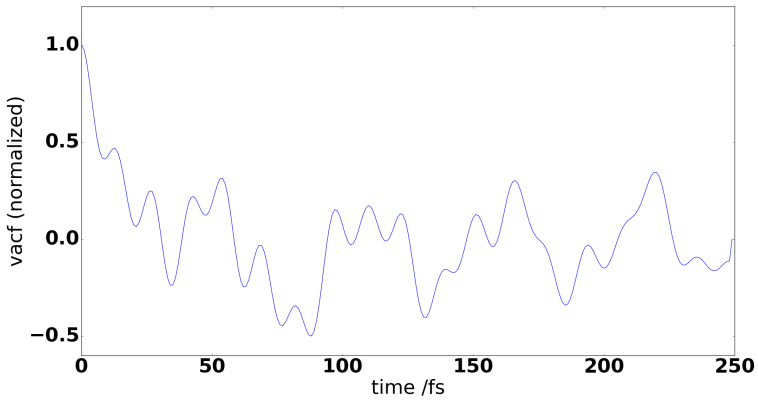

Figure 3.2: The VACF of a system of CO_2-III. The system was simulated using NVT DFT-MD calculations for 1750 steps with 16 CO_2 molecules, at a temperature T=200 K and a volume V=449.35 Å^3. It can be seen that there is a correlation in how the atoms in the solid CO_2 crystal move. This represents lattice vibrations in the crystal. The VACF is normalized to have a value of one at the time zero.

The VACF can be used to investigate some of the dynamical properties of the system. It can be used to calculate certain phonon frequencies or the diffusion coefficient of a system. Therefore comparing the VACF for MD simulations done with the two different methods of ab-initio MD and NNP can yield information about how similar the system's dynamics derived from both methods are.
In figure 3.2 an example of a VACF for an ab-initio MD simulation of CO_2-III is presented. The same VACF will be compared to the results of the NNP-MD simulations in Chapter 6. The VACF $c_v(t)$ can be calculated using the velocity data at each time step as follows:

$$\begin{aligned} c_v(t) &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i(t_0)\cdot\mathbf{v}_i(t), \\ t &= t_0 + n\cdot\Delta t, \end{aligned} \tag{3.20}$$

where N is the number of atoms in the system, $\mathbf{v}_i(t_0)$ is the velocity of atom i at a set time step t_0, $\mathbf{v}_i(t)$ is the velocity of the same atom at any later time step t and n is the discrete simulation step that corresponds to time t. This equation is then computed for each time step and the trajectory of the VACF can be plotted. Additionally the VACF can be not only averaged over all atoms but also over different starting points t_0 where n stays unchanged. This means that when computing the VACF for a trajectory with N total time steps and corresponding velocities only the first starting point yields a VACF with N entries. Starting at the second point in time the VACF has only $N-1$ entries and so forth until the last point can not be correlated anymore because there are no further points left. Therefore the first VACF entry can be averaged over the $N-1$ other first entries of all the VACFs. The second entry can be averaged over $N-2$ other second entries of all the VACFs. Up to the last entry that only exists in the first VACF and can not be averaged at all. This leads to an increasing statistical error for each subsequent point in the VACF. In Chapter 6 different VACFs for CO_2 are presented.

3.5.2 RDF

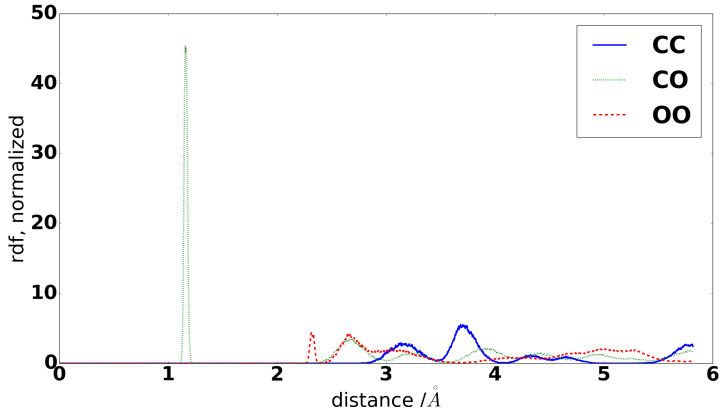

Figure 3.3: This figure shows the three relevant RDFs in a system of CO_2-III. The RDF presented was produced by averaging over 1750 configurations of a NVT DFT-MD trajectory with the parameters N=16 CO_2 molecules, T=200 K and V=449.35 Å^3. There is one RDF for each combination of atoms (CC, CO, OO). The first peak of the CO-RDF represents the bond length of the CO_2 molecules in this simulation and the first peak in the OO-RDF is the distance between the two oxygen atoms of one molecule. The other peaks are all representations of the distances between neighboring molecules in the crystal lattice of CO_2-III. The RDF is normalized in a way, that it converges to one for an infinite distance.

As opposed to the VACF the RDF (or pair correlation function $g(r)$) can be used to investigate the static makeup of a system. It contains information about inter atomic as well as inter molecular distances. In a solid phase this can be used to verify if the crystal structure is the same for both simulation methods. However also for a liquid the RDF contains important information, for example the C-O bond length. In figure 3.4 the RDF for an ab-initio MD simulation is presented. It is the same simulation as for the VACF in the previous section and will also be used to test the results of the NNP-MD simulations in Chapter 6.
The RDF states how many atoms are found at distance r around a reference atom i in a system of atoms and compares it to the ideal gas. It is then averaged using each of the N atoms in the system as the reference atom i once. The RDF for an MD simulation can be calculated by:

$$g(r) = \frac{1}{4\pi r^2 \rho N} \left\langle \sum_{i \neq j} \delta(r - r_{ij}) \right\rangle , \tag{3.21}$$

here $\rho = N/V$ is the density (where N is the number of atoms in the volume V), $\delta()$ is the Dirac delta function and r_{ij} is the distance between atom i and atom j. In Chapter 6 some RDFs are shown for solid CO_2.

3.6 Thermodynamic ensembles

When doing MD with the velocity Verlet algorithm the energy E is approximately preserved due to the symplectic nature of the algorithm. At the same time the number of atoms N and the volume of the simulation box V stay unchanged. Therefore time averages computed in an MD simulation with the velocity Verlet algorithm naturally correspond to the NVE or micro canonical ensemble. There are other ensembles though that can be very useful for

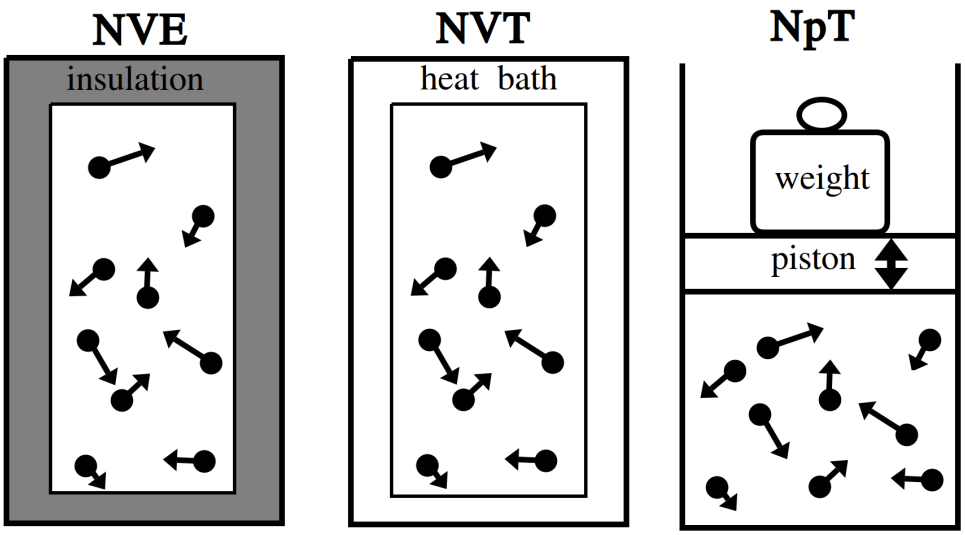

Figure 3.4: The three thermodynamic ensembles that are relevant for this thesis are the NVE, NVT and NpT ensemble. The NVE ensemble can be pictured as a box with a fixed volume and particle number that is separated from the outside world through a layer of perfect insulation and therefore can not exchange energy with its surroundings. The NVT ensemble is a box with fixed volume and number of particles surrounded by a heat bath of constant temperature T. The inside of the box can exchange energy with the heat bath and therefore always stays at a constant temperature T. The NpT ensemble is a box with variable volume and fixed number of particles. One end of the box can be imagined as attached to a movable piston that is pressed downward by a constant weight. This means that the volume of the box can change and the pressure in the box is constant.

MD simulations, mainly the NVT or canonical ensemble and the NpT or isothermal-isobaric ensemble. Both of those ensembles are more closely related to the conditions that are found in experimental physics, where it is easier to control temperature T and pressure p than the total energy E of the system. However the velocity Verlet algorithm produces simulations that behave according to the NVE ensemble. This means that so called thermostats and barostats are needed to modify the equations of motion in such a way that a constant temperature and pressure are achieved and to make the simulation behave as if it was in the NVT or NpT ensemble. In the following sections the Nosé-Hoover thermostat [43, 44] and the Parrinello-Rahman barostat [45] will be discussed. These two methods are the basis of thermostatting and barostatting in the LAMMPS software package. To combine barostatting with thermostatting LAMMPS uses the approach introduced by Shinoda et. al. [46] that is a computationally efficient way to combine a Parrinello-Rahman type barostat with a Nosé-Hoover type thermostat.

3.6.1 Nosé-Hoover thermostat

The Nosé-Hoover thermostat [43, 44] modifies the equation of motion by adding an additional degree of freedom and thereby extending the phase space the dynamics are calculated in. This additional degree of freedom comes in the form of a one particle heat bath that checks if the system's temperature is above or below the target temperature and then rescales the velocity accordingly.

This leads to the extended Hamiltonian:

$$\mathcal{H}_{\mathrm{NH}}(p^N, r^N, p_s, s) = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i s^2} + \mathcal{V}(r^N) + \frac{p_s^2}{2Q} + g\mathrm{k_B}T\ln(s), \tag{3.22}$$

here the thermostat variable s rescales the kinetic energy of the system, the corresponding conjugate momentum is p_s. The variable Q determines how quickly the thermostat reacts and is called the mass of the thermostat, despite not having the unit of a mass but actually energy·time^2. Further T is the target temperature of the system and the parameter g is chosen such that the micro canonical distribution in the now 6N+2 dimensional phase space that includes the imaginary particle results in a canonical distribution in the 6N dimensional phase space spanned by the original system. For the Nosé-Hoover thermostat a value of $g = 3N$ can be found that has to be used. It is also required that the simulation is ergodic.
The extended equations of motion of the Nosé-Hoover thermostat have the following form:

$$\begin{aligned}
\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t'} &= \frac{\mathbf{p}_i'}{m_i}, \\
\frac{\mathrm{d}\mathbf{p}_i'}{\mathrm{d}t'} &= \mathbf{F}_i - \zeta\mathbf{p}_i', \\
\frac{\mathrm{d}\zeta}{\mathrm{d}t'} &= \frac{1}{Q}\left(\sum_{i=1}^{N}\frac{\mathbf{p}_i'^2}{m_i} - g\mathrm{k_B}T\right), \\
\frac{\mathrm{d}\ln(s)}{\mathrm{d}t'} &= \zeta,
\end{aligned} \tag{3.23}$$

where $\mathbf{t}_i'$, $\mathbf{p}_i'$ and ζ are defined as:

$$\begin{aligned}
\mathbf{p}_i' &:= \frac{\mathbf{p}_i}{s}, \\
\mathrm{d}t' &:= \frac{\mathrm{d}t}{s}, \\
\zeta &:= \frac{sp_s'}{Q},
\end{aligned} \tag{3.24}$$

only the first three equations in 3.23 are needed to calculate the dynamics of r^N, p^N and ζ. However using all four equations as well as $g = 3N$ the total energy of the Hamiltonian is conserved. This is not the same as the total energy of the NVT ensemble that is being simulated but can be used to check the simulation for errors. The transformations that were done to the variables in order to arrive at the equations of motion are not canonical. Therefore the new equations of motion lose their symplectic property and the total energy of the NVT ensemble is not conserved anymore.
Looking at equations 3.23 one can now also see how the Nosé-Hoover thermostat keeps the temperature stable: the equation for the friction variable ζ (eq. 3.23, line 3) compares the actual kinetic energy to the average kinetic energy at the target temperature and the sign of the right hand side of the equation changes accordingly. In the equation for the momenta (eq. 3.23, line 2) this leads to an increase in momentum if the instantaneous kinetic energy of the system is smaller than the average at the goal temperature and to a decrease if it is larger. Effectively this resembles a heat bath that either transfers energy to the system if the system is colder than the heat bath or energy is transferred out of the system if it is hotter than the heat bath.
It should also be noted that the thermostat does not react instantaneously to the friction variable but rather depends on the time derivative of said friction variable. The speed of this

reaction is determined by the mass of the thermostat Q. This can be noticed in simulations in the form of a slow (compared to the simulation time step) oscillation of the instantaneous temperature around the target temperature.

3.6.2 Parrinello-Rahman barostats

The Parrinello-Rahman barostat [45] is a way for the cell shape and volume of a MD simulation to change according to the difference between an externally applied goal stress and the internal stress. The goal of the barostat is to keep the pressure of the system at a constant level similar to what a termostat does for the temperature. Parrinello-Rahman type barostats can be used in combination with the previously discussed Nosé-Hoover thermostat to control pressure and temperature in order to sample time averages that correspond to NpT ensemble averages, under the assumption of ergodicity.
The idea behind the Parrinello-Rahman barostat is to extend the equations of motion by adding nine additional degrees of freedom in the form of the nine components of the three vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ that span the simulation box. By describing the dynamics of the box vectors depending on the difference between internal and external stress the shape and volume of the box can be adjusted so that the pressure is kept constant on average.
The box matrix $\mathbf{h}$ is defined as the matrix with the three vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ as its columns. This way the coordinates $\mathbf{r}_i$ of an atom can be expressed in terms of the scaled coordinates $\mathbf{s}_i$ as $\mathbf{r}_i = \mathbf{h}\mathbf{s}_i$, where each component of $\mathbf{s}_i$ is between 0 and 1. This allows for the positions of all atoms to be easily scaled by a change in the shape and volume of the box. The Hamiltonian of the Parrinello-Rahman barostat in the case of hydrostatic pressure being applied has the form:

$$\mathcal{H}_{\mathrm{PR}} = \sum_i \frac{1}{2} m_i (\mathbf{h}\frac{\mathrm{d}\mathbf{s}_i}{\mathrm{d}t})^2 + \mathcal{V}(r^N) + \frac{1}{2} W \mathrm{Tr}\left(\frac{\mathrm{d}h^T}{\mathrm{d}t}\frac{\mathrm{d}h}{\mathrm{d}t}\right) + p\Omega, \tag{3.25}$$

here Ω is the cell volume, p is the pressure that is imposed on the system, Tr() is the trace of a matrix and W has the dimension of a mass and functions similar to the mass of the thermostat in the previous section. W should be chosen in a way that the relaxation time the system needs to compensate differences between the external pressure and the internal stress is of the magnitude of $\frac{L}{c}$ where L is the box size and c is the speed of sound in the system. This Hamiltonian leads to the following equations of motion:

$$\begin{aligned} \frac{\mathrm{d}^2\mathbf{s}_i}{\mathrm{d}t^2} &= -\frac{\mathbf{F}_i}{m_i} - \mathbf{G}^{-1}\frac{\mathrm{d}\mathbf{G}}{\mathrm{d}t}\frac{\mathrm{d}\mathbf{s}_i}{\mathrm{d}t}, \\ \frac{\mathrm{d}^2\mathbf{h}}{\mathrm{d}t^2} &= \Omega \mathbf{W}^{-1}(\mathbf{h}^T)^{-1}(\boldsymbol{\pi} - \mathbf{p}), \end{aligned} \tag{3.26}$$

here $\mathbf{G} = \mathbf{h}^T\mathbf{h}$ is the metric tensor, and $(\boldsymbol{\pi} - \mathbf{p})$ is the difference between the internal stress and the externally applied pressure. This means that the box matrix $\mathbf{h}$ depends on its current shape and volume in correlation to the difference between internal stress and external pressure and the force in the first line of the equation of motion is influenced by G and the derivative of G which depends on the box matrix **h** itself.
On its own this Parinello-Rahman barostat actually generates a NpH ensemble with constant enthalpy H. However in combination with a thermostat the enthalpy is not conserved anymore and a NpT ensemble is the result.
It is advisable to first equilibrate a system just with a thermostat before using an additional barostat because poorly equilibrated systems can lead to large oscillations of the pressure.

3.7 LAMMPS

The Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a software package used for a multitude of different MD simulations [3].
In this thesis it was used in it's simplest form to simulate bulk CO_2 using the modified EPM2 empirical force field [4]. Additionally LAMMPS was used with the interface developed for n2p2 [47] in order to do MD simulations with the neural network potential developed for CO_2 in this thesis.
LAMMPS is a very useful tool for anyone who is interested in doing MD simulations due to it offering options to implement different customizable empirical potentials as well as different thermostats and barostats.
The use for simple bulk simulations as in this thesis does not even graze the surface of what LAMMPS is capable of doing.

4 Calculating Inter-Atomic and Inter-Molecular Forces

In order to carry out MD simulations of any kind it is necessary to calculate the forces between the atoms and molecules that are to be simulated. In this chapter three methods will be presented to calculate those forces: empirical force fields, ab-initio methods and the neural network potential.
In this thesis all three methods were utilized. First an empirical force field was used for MD runs with LAMMPS (see Chapter 3) to create a diverse set of configurations of CO_2 molecules. Those configurations were then fed to the Vienna Ab-initio Simulation Package (VASP) which uses density functional theory to calculate the inter atomic forces and energy of each of those systems from first principle, described in section 4.2. With the forces and energies from VASP then the NN is trained with n2p2 (see Chapter 5) and used to do MD simulations. To test how well the NNP reproduces the results of the DFT calculations VASP is used again, this time to do an NVT MD simulation that will be compared to a similar NVT MD simulation done using the NNP, which is described in Chapter 6.

4.1 Empirical force fields

Empirical force fields are a very straightforward way to describe the inter atomic potential for given types of atoms or molecules. The idea behind empirical force fields is to look at the pairwise interaction of single atoms or molecules like CO_2 or even more complex organic molecules, and reconstruct an approximation for how the potential energy of the system behaves. From this potential energy then the force field can be derived that describes the forces acting on each atom.
Usually this potential includes the Coulomb interaction due to charges, a Lennard-Jones-Potential or similar for non bonded interactions and in the case of molecules an intra molecular potential.
The big advantage of force fields when used for MD simulations is their computationally simple form that allows to run simulations with a large number of atoms for long run-times.
In the following sections the relevant ideas about empirical force fields that are needed to understand the force field that was used for MD simulations on CO_2 in this thesis will be discussed. All simulations that utilize force fields were performed using the software package LAMMPS [3].

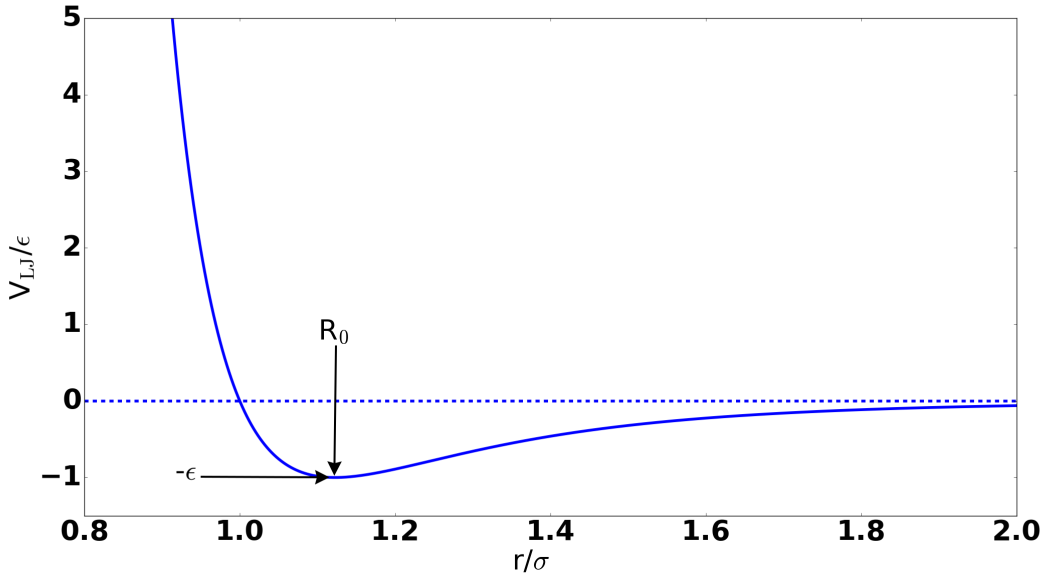

Figure 4.1: Lennard-Jones-Potential for the interaction of two atoms (see equation 4.1). $R_0 = 2^{1/6}\sigma$ denotes the inter atomic distance for the potential energy minimum, which is the equilibrium distance of the two atoms. The potential is repulsive when two atoms are close to each other and attractive at greater distances.

4.1.1 Lennard-Jones potential

The Lennard-Jones-Potential, $V_C LJ$ (eq. 4.1) [48] is often used to describe the interaction between two atoms that are not bonded, not charged and interact only through van der Waals forces, which are weak, undirected and act on a short range [49]. The Lennard-Jones-Potential uses the distance between two atoms r_{ij} as a variable and the two parameters σ, that determines the distance at which the potential energy moves from positive to negative values, and ϵ, that sets the minimum of the energy.

$$V_{\mathrm{LJ}}(r) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]. \tag{4.1}$$

The Lennard-Jones-Potential only treats the interaction between pairs of atoms, so in a large system of e.g. argon the total force on one atom is the sum of the pairwise calculated forces between that atom and its neighbors within a certain cutoff distance, as seen in figure 4.1. The cutoff distance can be justified by how quickly the potential energy converges towards zero and therefore becomes negligible. This approach throws all the quantum mechanical phenomena, that are responsible for the attraction and repulsion of atoms at different distances, into one equation. This of course is an approximation which simplifies the actual interaction between the atoms described, but makes it possible to calculate the force acting on a given atom with very low computational cost.
As the name empirical force field suggests, empirical data is necessary in order to determine the parameters σ and ϵ. This empirical data can be for example the experimental parameters of the crystal structure and evaporation heat of metals [50] or even data about the inter atomic interaction from ab-initio simulations [51].
For the EPM force field (see section 4.1.5) the experimental vapor-liquid coexistence line of CO_2 was used to fit the parameters [4]. This means that this model is most reliable close to this coexistence line. If a simulation is done with this model that is far removed from the

coexistence line then it has to be first shown that the model can still produce relevant results there. However in the case of EPM it has indeed been shown that the model can be applied more generally, not only to the area of vapor-liquid coexistence.
It is always a goal for empirical force fields to be applicable to as wide a range of problems as possible while staying as simple as possible at the same time. This means the transferability of a force field should be high.

4.1.2 Coulomb potential

The Coulomb potential V_{C} describes the potential created by static electrical charges. It has the form:

$$V_{\mathrm{C}}(r_{ij}) = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}}, \tag{4.2}$$

here ϵ_0 is the vacuum permittivity and r_{ij} is the distance of the two charges q_i and q_j. The force acting between the two charges is the Coulomb force. This force is responsible for most long range interactions in simulations due to the slow rate at which it decays to zero and therefore has to be treated in a more complex way than the Lennard-Jones potential with its simple cutoff (see section 4.1.1). In LAMMPS this is handled by a so called PPPM (Particle-Particle Particle-Mesh) solver for periodic potentials, that is closely related to the PME (Particle-Mesh Ewald) solver used by other MD software packages. The class of PPPM algorithms was developed by Hockney and Eastwood between 1973 and 1980 [52].
The basic idea is to separate a long range force (here the Coulomb force) into a short range force F_{sr} and a smoothly varying force R. While F_{sr} is calculated directly between each particle-particle pair (PP) within a certain cutoff, similarly to how other short range interactions like the LJ-potential are treated, the long range interaction is calculated with the particle-mesh (PM) approximation in Fourier space. The advantage is that this way the long range interaction can be efficiently treated due to its quick convergence in Fourier space.

4.1.3 Force fields for more than one type of atoms

One Lennard-Jones-Potential alone only sufficiently describes systems of identical atoms that do not form molecules.
The next more complex type of problem is to describe a system with two species of atoms, A and B which are not bonded. In this case there needs to be one set of Lennard-Jones parameters, ϵ and σ, for each of the three possible combinations of atoms: AA, BB and $AB{=}BA$. This means that when all pairwise interactions with one atom, for example of species A, are calculated there needs to be one Lennard-Jones-Potential for the interaction with its neighbors of type A and one for the interactions with neighbors of type B. The same is true for an atom of species B and its neighbors of both types. This leads to three pairs of σ and ϵ.
The parameters for the pairs AA and BB have to be found as described in section 4.1.1. The parameters for the combination AB can be found by using the so called combining rules (also known as mixing rules).
The most commonly used mixing rules are the Lorentz-Berthelot rules, also sometimes referred to as arithmetic mixing rules, found by Lorentz and Berthelot [53, 54]:

$$\begin{aligned} \epsilon_{AB} &= \sqrt{\epsilon_A \epsilon_B}, \\ \sigma_{AB} &= \frac{1}{2}(\sigma_A + \sigma_B), \end{aligned} \tag{4.3}$$

similar to the Lorentz-Berthelot mixing rules are the geometric mixing rules that are also commonly used for force fields in MD simulations:

$$\begin{aligned} \epsilon_{AB} &= \sqrt{\epsilon_A \epsilon_B}, \\ \sigma_{AB} &= \sqrt{\sigma_A \sigma_B}, \end{aligned} \tag{4.4}$$

where the calculation of ϵ_{AB} is the same but the calculation of σ_{AB} differs. With these rules it is possible to find the Lennard-Jones parameters for a pair of atoms with two different types. This approach can also be extended to more than two types of atoms, because the Lennard-Jones-Potential can only be used for pair interactions.

4.1.4 Force fields for molecules

So far only non-bonded interactions of atoms were considered, but what happens if the system to be simulated consists of molecules?
In the case of most molecules the bonds between the atoms can not be described by the Lennard-Jones-Potential. The reason is that molecular bonds usually are very strong and often directed, as for example in CO_2.
Generally when dealing with molecules there are two levels of interactions: one is molecules interacting with other molecules and the other is the atoms within one molecule interacting with each other.
The interaction between two molecules can be treated similarly to the interaction between two atoms. Each molecule has one or more Lennard-Jones interaction sites, that work the same way as for a single atom, as well as charges that are responsible for the long range Coulomb interaction. For example the EPM force field for CO_2 has one charge and one Lennard-Jones interaction site on each of its three atoms. These are responsible for the forces that act between individual molecules.
The interaction of the bonded atoms within one molecule can be treated two ways: by using constraints or by using intra molecular potentials.
Constraint algorithms, like RATTLE and SHAKE, work by enforcing an additional constraint force that ensures that the bond length and angle of the atoms in a molecule are always at a specified value [55]. Molecules treated this way are rigid and there is no significant deviation in bond length and angle other than through the iterative solution process of those algorithms. This does also suppress all intra molecular vibrations, which is justified because the vibrations are decoupled from the movement of the molecule as a whole and are of such a high frequency that they can be replaced by an effective bond length. Unfortunately the training configurations for a NN need to, among other things, display a certain variation in bond length and angle, in order to sample the potential energy surface of a system. Therefore intra molecular potentials were chosen in this work instead of constraints.
The intra molecular potentials work by introducing potentials between bonded atoms that define a bond-force that acts on the atoms within one molecule on top of the forces from other molecules. There are many forms and shapes for such potentials with the most common ones being harmonic potentials. In order to describe a directed bond the bond-stretching potential V_{bond} and the angle-bending potential V_{angle} are both needed:

$$V_{\mathrm{bond}}(r) = \frac{K_{\mathrm{b}}}{2}(r_{ij} - r_0)^2, \tag{4.5}$$

$$V_{\mathrm{angle}}(r) = \frac{K_{\mathrm{a}}}{2}(\phi_{ijk} - \phi_0)^2, \tag{4.6}$$

where K_a and K_b are the force constants for each potential, the variable r_{ij} denotes the distance between two atoms i and j and r_0 is the equilibrium distance of the atoms. Similarly ϕ_{ijk} denotes the angle between three atoms i, j, k and ϕ_0 is the equilibrium angle. The bond stretching potential is responsible for two atoms vibrating about an equilibrium bond length, while the angle-stretching potential does the same for an equilibrium angle. The harmonic potentials used can be pictured by replacing the bonds within a molecule with springs. The values for the constants K for each of the potentials determine how hard it is to push the atoms out of their equilibrium distance/angle as well as the frequency at which the molecule vibrates and have to be found either through experiments, such as observing the IR-spectrum of a molecule, or ab-initio calculations. So finally the result is a molecule that has a certain equilibrium bond length and bond angle and interacts with other molecules through Coulomb interaction and Lennard-Jones-Potentials. This knowledge will be applied in the next section for finding an empirical force field to use for MD simulations of CO_2.

4.1.5 Force fields for CO_2

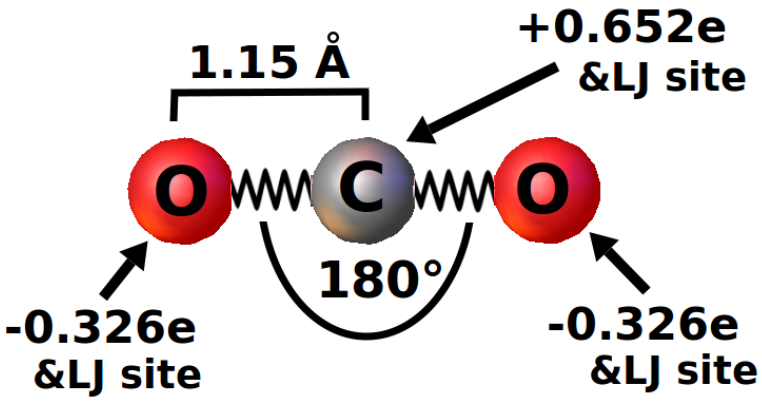

Figure 4.2: The fully flexible version of the EPM2 force field. There is a LJ interaction site as well as a charge on each atom. The equilibrium distance between the C and O atoms is set as 1.15 Å. The equilibrium angle is 180°.

There are several established force fields to describe CO_2. The two most prominent ones are TraPPE [56] and EPM or rather its enhanced version EPM2 [4]. The TraPPE force field is more commonly used but has one distinct disadvantage for this thesis: it describes the CO_2 as being rigid. This means that TraPPE uses constraints to keep CO-distance in the CO_2 molecules always at 1.16 Å and the OCO-angle always at 180°. In order to keep a 180° angle using constraints TraPPE also needs two dummy atoms that are on the same axis as the regular CO_2 atoms. Altogether this means that the interactions of the atoms within a CO_2 molecule have been modeled as all three atoms sitting on a rigid stick. This is generally a good approach for molecules like CO_2 that display a very rigid bond angle and bond length in nature. Therefore the TraPPE force field yields good results for a broad spectrum of simulations. However in order to train a NN it is necessary to include configurations in which the molecules are bent or stretched, otherwise the NN will have no information about how to treat CO_2 molecules that are not perfectly straight and not at the equilibrium bond length.

The fully flexible EPM2 is a variation of the rigid EPM2 force field. It consists of one Lennard-Jones and one Coulomb interaction site on each atom and assumes harmonic potentials, between the atoms of a molecule to treat the bond angle and bond length within

Table 4.1: Parmaters for the fully flexible EPM2 force field [4] [5]. The parameters for the C-O interaction are calulated using the Lorentz-Berthelot mixing rules, equation 4.3

element	ϵ (K)	σ (Å)	q (e)
C	28.13	2.76	0.652
O	80.51	3.03	-0.326
$K_{\mathrm{bond}} =$		10 739 kJ/mol/Å^2	
$K_{\mathrm{angle}} =$		1236 kJ/mol/rad^2	
$d_{C-O} =$		1.15 Å	

one molecule, see equation 4.7 and 4.6 as well as figure 4.2 [57]. The total potential of the flexible EPM2 force field therefore combines the Lennard-Jones potential with the Coulomb potential and with the bond and angle potentials and has the form:

$$V = V_{\mathrm{LJ}} + V_{\mathrm{C}} + V_{\mathrm{B}} + V_{\mathrm{A}},$$
$$= \sum_{\substack{i,j \\ i<j}}^{N} \left(4\epsilon_{ij} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] + \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} \right) + \sum_{\langle i,j \rangle} \frac{K_{\mathrm{b}}}{2} (r_{ij} - r_0)^2 + \sum_{\langle i,j,k \rangle} \frac{K_{\mathrm{a}}}{2} (\phi_{ijk} - \phi_0)^2, \tag{4.7}$$

where the sum over i, j to N is the sum over all pairs of atoms in the simulation, the sum over $\langle i, j \rangle$ should denote the sum over pairs of bonded atoms in one molecule and $\langle i, j, k \rangle$ is the sum over all triplets of atoms in one molecule that have a bonding angle in common.
The partial charges for the EPM2 force field are chosen to correctly reproduce the quadrupole moment of CO_2 of 4.3 Buckinghams [4]. The parameters of this force field can be found in table 4.1. The flexible EPM2 force field was also shown to yield comparable simulation results for the equation of state of CO_2 as the TraPPE and not flexible EPM2 [58].

4.2 Density functional theory

Density functional theory (DFT) is an ab-initio method to calculate the energy and forces in a system of atoms. In this context ab-initio means that the method relies on the fundamental Schrödinger equation and derives all results from it. This means it does not need empirical data other than the electronic configuration of the atoms in the simulation in order to calculate energy and forces. Empirical force fields on the other hand always need experimental data to model the force field after. This immediately leads to the biggest advantages of ab-initio MD simulations: they can do MD simulations for systems without any prior outside knowledge.
This allows for flexible molecules that can spontaneously form or disassociate as well as for the exploration of regions of the phase diagram, including new materials and alloys in material sciences, where no experimental data is available yet. This is an important area of research where ab-initio MD is used. As opposed to empirical potentials that make approximations such as using the LJ-Potential and fixed bonds, the potentials that are calculated using DFT methods are more complex and describe reality more detailed in comparison. Of course this comes at a computational cost and so the biggest disadvantage of DFT calculations is that they can only be feasibly done for very small system sizes and very short periods of time, both magnitudes lower than what can be achieved using empirical potentials.
In this thesis the Vienna Ab-initio Software Package (VASP) [6, 7, 8, 9, 10] was used for

the ab initio force and energy calculations. It is based on the density functional theory. The theoretical foundation for DFT as is used by VASP are the Kohn-Sham equations. In the following a short descritpion of the theory behind DFT is provided using a lecture by Juan Carlos Cuevas as a basis [59].
The fundamental problem of DFT is to solve the time independent, non-relativistic Schrödinger equation for a system of many atoms. In the following equations atomic units are used:

$$\hat{H}\Psi_i(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{R}_1, ..., \mathbf{R}_M) = E_i\Psi_i(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{R}_1, ..., \mathbf{R}_M), \tag{4.8}$$

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \frac{1}{2}\sum_{A=1}^{M}\frac{1}{M_A}\nabla_A^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{r_{ij}} + \sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{R_{AB}}, \tag{4.9}$$

here $\hat{H}$ is the Hamiltonian of the system with M atoms and N electrons, $\mathbf{x}_i$ and $\mathbf{R}_A$ are the coordinates of electrons and nuclei respectively. Further A and B iterate over the number of atoms M and i and j iterate over the number of electrons N, Z_A is the proton number and M_A the mass of each atom. It can be seen that this Hamiltonian does not scale with the number of atoms but rather with the number of electrons and protons. Therefore even small systems need to be approximated. This leads to the Born-Oppenheimer approximation where the atomic nuclei that are magnitudes heavier than the electrons are considered to be stationary. This means the electrons move in a field of static nuclei. Therefore the electronic Hamiltonian $\hat{H}_{elec}$ is,

$$\hat{H}_{\text{elec}} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{r_{ij}} = \hat{T} + \hat{V}_{\text{Ne}} + \hat{V}_{\text{ee}}, \tag{4.10}$$

where the total energy E_{tot} is the eigenvalue of the electronic Hamiltonian plus the constant contribution of the nucleus E_{nuc}

$$\begin{aligned} \hat{H}_{\text{elec}}\Psi_{\text{elec}} &= E_{\text{elec}}\Psi_{\text{elec}}, \\ E_{\text{tot}} &= E_{\text{elec}} + E_{\text{nuc}}, \\ E_{\text{nuc}} &= \sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{R_{AB}}. \end{aligned} \tag{4.11}$$

This approximation is the first step to be able to numerically solve the many body Schrödinger equation. The electron density $\rho(\mathbf{r})$ that gives the DFT its name is the next ingredient that is needed. It describes the probability of finding an electron within a certain volume. The two Hohenberg-Kohn theorems show that through this electron density $\rho(\mathbf{r})$ a unique Hamiltonian can be found.
The first Hohenberg-Kohn theorem states that the potential V_{Ne} that describes the attraction between electrons and nuclei is uniquely determined by $\rho(\mathbf{r})$. This fully determines the Hamiltonian because V_{ee} and $\hat{T}$ are already known and therefore the ground state energy of the system is a function of $\rho(\mathbf{r})$.
The second Hohenberg-Kohn theorem states that the true ground state electron density $\rho(\mathbf{r})$ minimizes the ground state energy E_0.
Therefore the problem DFT tries to solve is to find the electron density that minimizes the ground state energy for a given system.
This leads to the Kohn-Sham equations that are at the heart of DFT calculations.

$$\hat{H}_{\text{KS}} = \hat{T} + \hat{V}_{\text{Ne}} + \hat{V}_{\text{ee}} + \hat{V}_{\text{XC}}, \tag{4.12}$$

here the exchange-correlation potential V_{XC} is a term that contains everything that is unknown. There is no exact expression for V_{XC}, but several approaches for approximations exist, like LDA (local density approximation) and GGA (generalized gradient approximation). These equations are solved recursively starting with a guess for $\rho(\mathbf{r})$ and then iteratively improving the electron density until reasonably close to the true electron density of the system.
With this theoretical foundation it is then possible to approximate the forces and energies of systems of atoms with very high precision. This will be used to train the NN and the resulting NNP-MD simulations will be compared to the DFT-MD simulations in Chapter 6.

4.3 Neural network potentials

In this chapter two methods to approximate the forces and energies in a systems of atoms have been presented so far. The computationally simple but less accurate force fields and the computationally complex but more precise DFT method. This section will present a third and comparably new option: the Neural Network Potential (NNP). The NNP lies in between the two previously mentioned methods in terms of computational complexity as well as accuracy. The idea behind the NNP is to use a NN to create a nonlinear fit of the PES of a system based on DFT data. It can be evaluated with similar computational cost to an empirical force field, although the functional form of the NNP is very abstract compared to the force fields that were described earlier in this chapter (section 4.1). Another difference to force fields is that the NN, just as the DFT, does not need any information about bond angles, bond lengths or which atoms form a molecule. Molecules are formed from atoms due to the forces that act on them without any prior knowledge about whether those atoms should form molecules. Therefore molecules can form and disband which makes the NN way more flexible than most force fields and much faster than DFT simulations.
Forces and energies obtained using the NNP are of similar accuracy as the ones from the DFT calculations that are used to train the NN. A comparison of DFT and NNP simulation results will be shown in Chapter 6 for simulations of CO_2 and was already shown for H_2O [1].
The NNP is only one of many promising new approaches to apply machine learning potentials to MD simulations. Other approaches are for example Gaussian approximation potentials [16] or support vector machines citesvm-book [60].
Part of the reason how it is possible to get results with comparable accuracy to DFT calculations but only at a fraction of the computational cost is that DFT algorithms are general approaches that can deal with any constellation of different atoms and molecules as long as pseudo potentials for all the elements are available. The NNP however has to be trained for each specific problem. This means that it can not be applied as generally as the DFT.
Another important point about the high dimensional NN that is used in this thesis is that only the environment within a certain cutoff radius around each atom is considered to calculate the energy and forces. This means that long range interactions, like the Coulomb force, are only considered based on the influence they have on the local configuration of atoms within that cutoff radius.
In this Thesis the n2p2, Neural Network Potential Package, was used that readily comes with an interface for the LAMMPS software package [61, 47]. This section will cover the basics about neural networks and neural network potentials. Chapter 5 will go into detail about the training process of the NN that was at the core of this thesis. The NNP approach used in this thesis was first developed by Behler and Parinello [21].

4.3.1 Feed forward neural networks

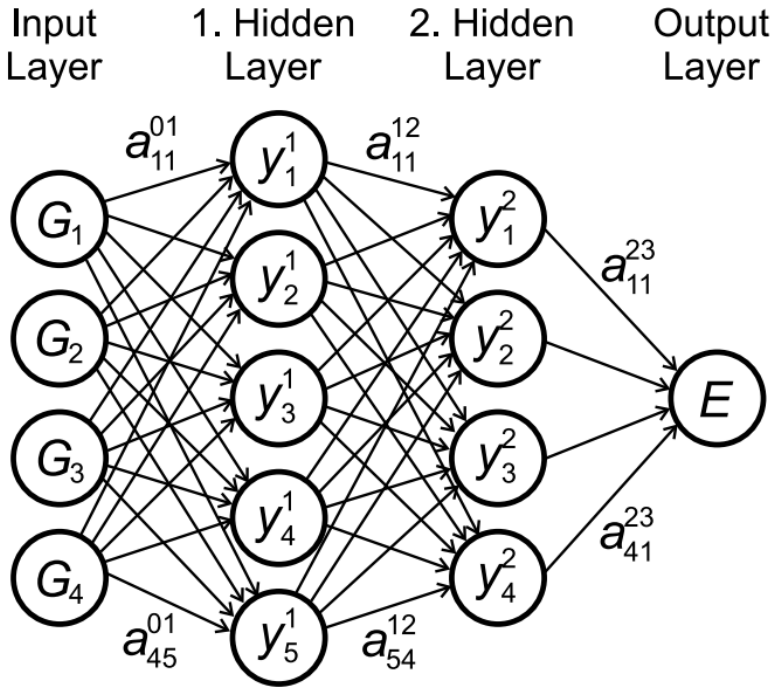

Figure 4.3: Basic structure of a feed forward NN. The displayed NN has the structure 4-5-4-1, which means that it consists of 4 input nodes, two hidden layers with 5 and 4 nodes, respectively, and an output layer with one node. In this case the input is in form of generalized coordinates, which describe the local environment around an atom. The value of the one output node is then the energy of the system described by the coordinates in the input layer. The output of the NN depends on the values of the weights $a_{ki}^{j-1,j}$ that are indicated as arrows. The Figure is taken from [62].

In order to write about neural network potentials it is important to give a short introduction to neural networks first.
The NNs that the NNP is based on are feed forward NNs. A feed forward NN is composed of three main parts or layers: the input layer, the hidden layer(s) and the output layer. Each such layer is build from nodes. Each node of each layer is connected to each node of the previous layer, which means that the information is fed into the input layer and then related forward through the NN. The output of the NN is the energy of the atomic system described in the input layer [63]. This is schematically shown for a simple feed forward NN in figure 4.3. In the input layer the NN is given information about a system of atoms. This can for example be atomic coordinates that describe the positions of all the atoms in a system. However this already leads to the first problem: translation and rotation should not change the energy of a system. However if all atomic coordinates were to be shifted along the same direction by the same distance or the system was rotated as a whole it would look to the NN like a completely new set of values for the input nodes and may lead to a different energy if the NN is not specifically trained to avoid that, which can be very complicated for larger systems. Even ordering the atoms differently and switching out the input of G_i and G_j would change the configuration the NN sees and would therefore also affect the energy that the NN predicts [62]. One remedy for this problem is to use generalized coordinates that are created

using symmetry functions (see section 4.3.3 for more detailed information). These generalized coordinates describe the local environment of a given atom up to a certain cutoff distance r_c and are the input for the first layer of the feed forward NN.
The input that is received by the nodes of the input layer is then propagated to the hidden layer nodes. The value y_i^j of the node i of a given layer j is then calculated by summing over the value y_k^{j-1} from each node k from the previous layer $j-1$, multiplying each of these values with a specific weight $a_{ki}^{j-1,j}$ and then adding the bias b_i^j. The result is then put in a nonlinear activation function f_i^j and the result is y_i^j.
This yields the following equation for the value of the node y_i^j [62],

$$y_i^j = f^j \left(b_i^j + \sum_k a_{ki}^{j-1,j} \cdot y_k^{j-1} \right). \tag{4.13}$$

Each node in the NN needs a nonlinear activation function f^j in order to be able to fit nonlinear functions. Without this activation function the NN would just create a linear combination of the coordinates to calculate the energy. In this thesis activation functions of the form $f^j(x) = \ln(1+e^x)$ were chosen for the hidden layer nodes. The output node uses the identity $f^j(x) = x$ as activation function.
By optimizing the weights $a_{ki}^{j-1,j}$ the NN can then fit the desired function. The bias b_i^j adds an additional and optional offset for the value of each node. For all purposes of this thesis the bias is set to zero.
Theoretically feed forward NNs of this form can be used to reproduce any real-valued function with any given accuracy [64, 65].
The fitting procedure of the NN to optimize the weights $a_{ki}^{j-1,j}$ using a suitable algorithm is presented in section 4.3.4.
The output layer only consists of one node. The value of this last node is the resulting energy E that corresponds to the atomic environment described in the input layer.

$$E = f^3 \left(b_1^3 + \sum_{l=1}^{4} a_{l1}^{23} \cdot f^2 \left(b_l^2 + \sum_{k=1}^{5} a_{kl}^{12} \cdot f^1 \left(b_k^1 + \sum_{j=1}^{4} a_{jk}^{01} \cdot G_j \right) \right) \right). \tag{4.14}$$

This equation describes how the feed forward NN shown in figure 4.3 calculates a total energy E for a system of four input coordinates G_i with two hidden layers of 5 and 4 nodes respectively, which can be denoted as a 4-5-4-1 network [62].

4.3.2 High dimensional neural network potentials

High dimensional NNPs are building on the idea of the simple feed forward NN, but try to overcome some of the difficulties that are faced when dealing with the high dimensional PESs of large systems.
A fully trained NN only works if the number and ordering of nodes is not changed [62]. This means that if the system size increases and the number of input nodes has to increase to represent a higher dimensional PES, the NN would have to be retrained completely. This is a serious problem considering that ideally the high dimensional NNP can represent systems of any size once trained correctly.
Another issue in feed forward NNs is that they can not reproduce certain properties of a physical system of atoms. Specifically exchanging the position of two identical atoms would change the input of the NN and therefore would be considered a new configuration that might not end in the same energy although physically the system should have the exact same energy

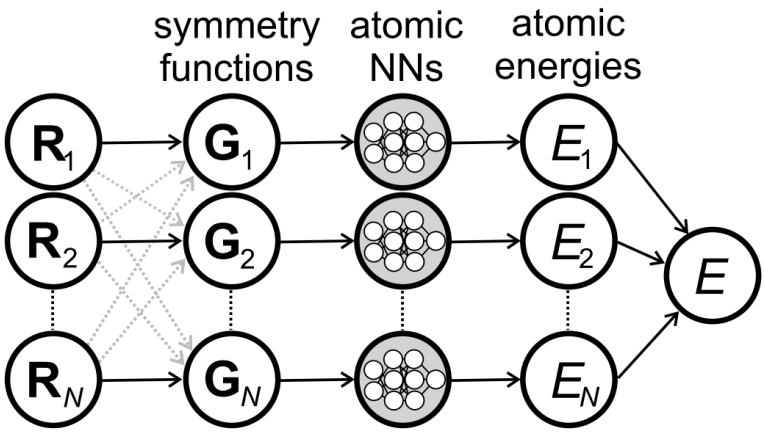

Figure 4.4: The diagram shows a high-dimensional feed forward NN where each individual atom contributes to the total energy of the system depending on its local environment up to a certain cutoff distance. First all the Cartesian coordinates R_i of the system's atoms are used to calculate a set of descriptors G_i (symmetry functions) for the environment of each atom. Those are then fed to a standard feed forward NN (atomic NNs) that yields the energy contribution E_i (atomic energies) of each atom in the system. Lastly the total energy E is the sum of the energies E_i. Figure taken from [62].

as before. In order to circumvent these problems, high dimensional NNP, as shown in figure 4.4, can be used [21].
A high dimensional NNP functions by replacing one feed forward NN that fits the total energy E of the system with multiple identical atomic NNs that only calculate the energy contributions E_i that each atom makes depending on its local environment,

$$E = \sum_i E_i. \tag{4.15}$$

For the training of the high dimensional NNP the weights of the atomic NN are optimized by requiring that the sum of the atomic energies has to equal the reference energy. It is important to note that all atomic NNs use identical structures and weights. This makes it possible for the high dimensional NNPs to solve one of the problems mentioned that arise when just one feed forward NN is used: the number of atoms in a system can easily be changed by changing the number of atomic NNs [21].
The individual energy contribution E_i of one atom depends on the positions of all neighboring atoms within a certain cutoff radius r_c. This means there needs to be a suitable method that can describe the atomic environment using a constant number of input nodes, that is independent of the number of neighbors of a given atom and that is invariant under the exchange of two identical atoms. This can be achieved using the symmetry functions that are described in the next section.

4.3.3 Symmetry functions

In the previous sections it was mentioned multiple times that the use of Cartesian coordinates as input for a NN leads to several problems. One of these problems (the independence of

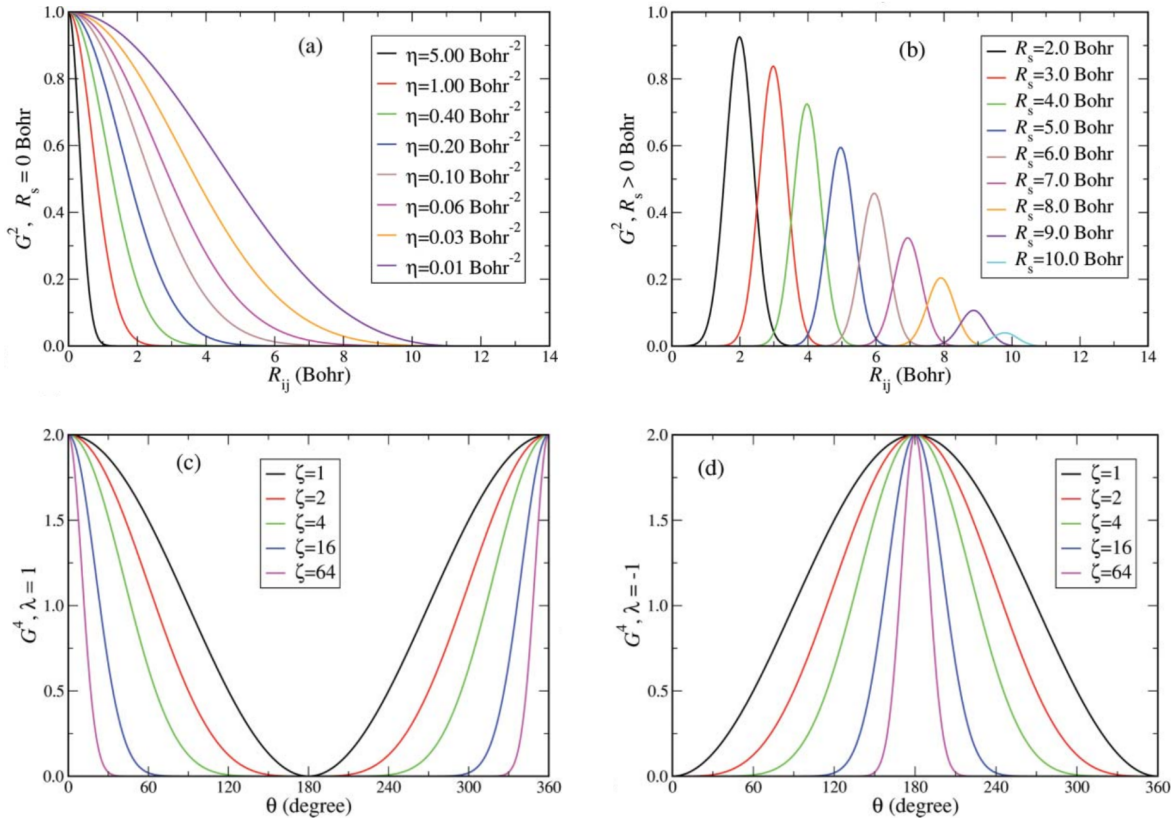

Figure 4.5: The two types of symmetry functions G^2 and G^4 in this figure correspond to the types G^1 and G^2 introduced by Behler [21] as mentioned in the text of this chapter. The upper row are radial symmetry functions that only depend on the distance R_{ij} of two atoms. The lower row are the angular parts of the angle θ dependent symmetry functions. This figure is taken from a later paper of Behler [62] where the symmetry function numbering had been changed. The figure is composed of two original figures and the lettering has been changed.

system size) can be solved by using the approach of high dimensional NNPs. The remaining issues are that the system's energy should be invariant under the following three operations:

- translation of the whole system
- rotation of the whole system
- exchange of the positions of two identical atoms

One way to address these remaining issues is to use symmetry functions to describe the local environment around an atom.
There are many different options and alternatives to using symmetry functions [66] but in this thesis atom centered symmetry functions of the type G^1 and G^2, introduced by Behler [21], are used. This is mainly due to their availability in the Neural Network Potential Package [61] that was used to train the NNs, see Chapter 5. The atom centerd symmetry functions used have the following forms, as can also be seen in figure 4.5:

$$G_i^1 = \sum_{j \neq i} e^{-\eta(r_{ij}-r_s)^2} f_c(r_{ij}). \tag{4.16}$$

The symmetry function G^1 is a radial symmetry function centered around an atom i. This means it depends on the distance r_{ij} between the atom i and all its neighbors j within a

certain cutoff distance r_c. The distance is then weighed by use of a Gauss function with the two parameters η and r_s and multiplied with the cutoff function $f_c(r_{ij})$.

$$G_i^2 = 2^{1-\zeta} \sum_{j,k \neq i} (1 + \lambda \cos \Theta_{ijk})^\zeta e^{-\eta(r_{ij}+r_{ik}+r_{jk})^2} f_c(r_{ij}) f_c(r_{ik}) f_c(r_{jk}). \tag{4.17}$$

The symmetry function G^2 is an angular symmetry function and centered around atom i as well. Therefore it is necessary to look at the angle and distances between atom i and all pairs of neighboring atoms j and k within the cutoff distance r_c. The radial part of G^2 is very similar to that of G^1 but depends on all three distances r_{ij}, r_{ik} and r_{jk} and the shift factor r_s equals zero. The angular part of G^2 depends on the angle $\Theta_{ijk} = \arccos\left(\frac{\mathbf{r}_{ij}\mathbf{r}_{ik}}{r_{ij}r_{ik}}\right)$ between the atoms as well as the additional parameters ζ and λ. While the parameters η, ζ and r_s can have any real value, the parameter $\lambda = \pm 1$.
The following cutoff function $f_c(r_{ij})$, as provided by n2p2, was used for this thesis:

$$f_c(r_{ij}) = \begin{cases} ((15 - 6r_{ij})r_{ij} - 10)r_{ij}^3 + 1 & r_{ij} < r_c, \\ 0 & r_{ij} > r_c. \end{cases} \tag{4.18}$$

The cutoff function $f_c(r_{ij})$ is used so that the symmetry functions are continuous and decay smoothly to the cutoff distance r_c. It only depends on the distance of two atoms r_ij and the cutoff distance r_c.
It is apparent that the value of a symmetry function does not change under translation or rotation of the system, as long as the distances between the atoms stay unchanged. The same goes for the exchanging of the positions of two identical atoms. As long as the same two distances are given to the symmetry function the order they are given in does not matter. The symmetry functions including the cutoff function can be seen in figure 4.5. The symmetry functions presented only depend on the distances between atoms but are independent of absolute positions.
In order to use the symmetry functions to describe the local environment of an atom a set of parameters for different symmetry functions is chosen first. After summing over all the pairs of atoms within the cutoff radius r_c each symmetry function yields one real value for each variation of parameters. Each of those numbers is then given to the NN as a value for one input node. The set of evaluated symmetry functions forms the complete input for the NN. This way the local environment of one atom can be described by a set of real numbers that always stays constant in size even if the number of neighbors of this atom is changing.
A last point to consider here is that there needs to be one atomic NN for each element in the system. Therefore there need to be symmetry functions for each combination of atoms. For example for CO_2 the center atom of the symmetry function could be either a carbon or an oxygen atom. There now needs to be one atomic NN that has only carbon centered symmetry functions as input and one atomic NN that has only oxygen centered symmetry functions as input. This means that there need to be radial symmetry functions for the combinations C-O, C-C and O-C, O-O as well as angular SFs for C-OO, C-OC, C-CC and O-OO, O-OC, O-CC. This way the NNs can distinguish between the different elements and the distribution of neighboring atoms of different elements are encoded in different symmetry functions and are distinguishable too.

4.3.4 Optimization of NN weights

The last part needed to make a NNP work is a method to optimize the weights $a_{ki}^{j-1,j}$ of the NN (see equation 4.13 and 4.14) in such a way that the high dimensional NNP fits the given

PES.
In order to fit a PES with a NNP, first reference energies, and optionally forces, are needed. These energies and forces can be acquired by using any other computational method on a configuration of atoms. It is not important for the NN what method is used to calculate the reference data, as long as it is consistent and does not change for all training configurations. Ideally the reference data is from ab-initio calculations because of their numerical accuracy and the potential speed-up from using a high dimensional NNP instead of an ab-initio method.
Once the training data in the form of the coordinates, energies and forces of a system of atoms is prepared, the NN can be trained to fit energies and forces consistent with the training method for a given atomic configuration. Here it is important that only a certain fraction of the training configurations are used for the actual training. The other part of the configurations is used as test configurations to see how well the NNP predicts their energy, without ever having seen them. The NN-forces can be calculated from the energy that is the output of the NN by applying the chain rule to the energy function of the NN [62]:

$$F_{k,\alpha} = -\frac{\partial E}{\partial R_{k,\alpha}} = -\sum_{i=1}^{N} \frac{\partial E_i}{\partial R_{k,\alpha}} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} \frac{\partial E_i}{\partial G_{i,j}} \frac{\partial G_{i,j}}{\partial R_{k,\alpha}}, \tag{4.19}$$

here $F_{k,\alpha}$ is the component of the force acting on atom k with respect to the coordinate $R_{k,\alpha}$, where $\alpha = (x, y, z)$. In the first sum N is the number of atoms in the system and M_i in the second sum is the number of symmetry functions describing atom i, which equals the number of input nodes of the atomic NN. The term $\frac{\partial E_i}{\partial G_{i,j}}$ depends on the structure of the NN, the term $\frac{\partial G_{i,j}}{\partial R_{k,\alpha}}$ depends on the symmetry functions used.
The optimization of the NN-weights in this thesis is done using a multi-stream Kalman filter that uses both energies and forces as reference and is part of the n2p2 software package [2]. This means that by applying the multi stream Kalman filter to train a high dimensional NNP with suitable training data the weights of the atomic NNs are being optimized to minimize the mean-square error of the predicted energy.

4.3.5 The Kalman filter

In the following a short summary of how the Kalman filter is applied to the training of high dimensional NNPs is presented, based on the approach described by Singraber [2]. For a more in depth coverage of the topic Kalman filter the book *Kalman Filtering and Neural Networks* [67] is an excellent resource.
Let $\mathbf{a}$ be the vector of all weights of the NN, ordered by layer. This vector of weights describes the current NNP. At the beginning of the optimization procedure these weights $\mathbf{a}(0)$ can have random values that will then be optimized with each iteration. Therefore there is a new set of weights $\mathbf{a}(t)$ after each Kalman filter iteration t. The vector $\mathbf{x}_i$, with $i = 1, ..., M$ is the vector that contains all the input node values that describe the training configuration i out of the M total training configurations. Each training configuration has one output node value $y_i^{\mathrm{Ref}}(\mathbf{x}_i)$ that was assigned by the chosen reference method and is stored in the vector $\mathbf{y}^{\mathrm{Ref}}$. Similarly the vector $\mathbf{y}(t)$ contains all M output node values $y_i(\mathbf{a}(t), \mathbf{x}_i)$ that are calculated by the NN with the weights $a(t)$.
For each iteration of the Kalman filter the error vector $\xi(t)$ is calculated. It describes how the output of the NNP $\mathbf{y}(t)$ differs from the reference output $\mathbf{y}^{\mathrm{Ref}}$,

$$\xi(t) = \mathbf{y}^{\mathrm{Ref}} - \mathbf{y}(t). \tag{4.20}$$

The derivative matrix $\mathbf{H}$ of the dimension $n \times m$ is constructed by taking the derivative of the error vector ξ with respect to all weight parameters a_i.

$$\mathbf{H}(t) = \left[\frac{\partial}{\partial a_1}, ..., \frac{\partial}{\partial a_n}\right]^T \xi^T(t). \tag{4.21}$$

Together with the $n \times n$ dimensional scaling matrix $\mathbf{A}$ and the error covariance matrix $\mathbf{P}$ the Kalman gain matrix $\mathbf{K}$ can be constructed

$$\mathbf{A}(t) = \left(\frac{1}{\eta(t)}\mathbb{1} + \mathbf{H}^T(t)\mathbf{P}(t)\mathbf{H}(t)\right)^T, \tag{4.22}$$

$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}(t)\mathbf{A}(t). \tag{4.23}$$

The factor $\eta(t)$ in the scaling matrix $\mathbf{A}$ is the learning rate that can be adjusted depending on the individual training.
The gain matrix $\mathbf{K}$ can now be used to calculate a new set of weights for the NN,

$$\mathbf{a}(t+1) = \mathbf{a}(t) + \mathbf{K}(t)\xi(t), \tag{4.24}$$

as well as the updated error covariance matrix $\mathbf{P}$:

$$\mathbf{P}(t+1) = \mathbf{P}(t) - \mathbf{K}(t)\mathbf{H}^T(t)\mathbf{P}(t) + \mathbf{Q}(t). \tag{4.25}$$

Here $\mathbf{Q}$ is the process noise matrix. This additional and artificial noise can be helpful in order not to end up in a local minimum. The algorithm so far describes what is called one training epoch. On average after each epoch the new set of weights $\mathbf{a}(t+1)$ should lead to a smaller root mean square error (RMSE):

$$RMSE = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\mathbf{y}_i^{\mathrm{Ref}} - \mathbf{y}_i)^2}, \tag{4.26}$$

due to it minimizing the NN's cost function $\Gamma = \sum_i (\mathbf{y}_i^{\mathrm{Ref}} - \mathbf{y}_i)^2$. The training is usually considered completed once the RMSE is below a certain threshold.
Both the matrix $\mathbf{P}$ and $\mathbf{Q}$ as well as the learning rate $\eta(t)$ need to be initialized carefully because the initialization can affect the outcome of the training. For the training with the n2p2 package the following initialization for the error covariance matrix is used:

$$\mathbf{P}(0) = \epsilon^{-1}\mathbb{1} \tag{4.27}$$

here ϵ can be chosen depending on the PES that the NNP should fit. For the NNP for CO_2 that is being trained in this thesis a value of $\epsilon = 10^{-2}$ was used (see Chapter5).
The artificial process noise matrix $\mathbf{Q}$ is initialized using the scheme:

$$\mathbf{Q}(t) = q(t)\mathbb{1} \tag{4.28}$$

where $q(t)$ is:

$$q(t) = \max(q_0 e^{-t/\tau_q}, q_{\min}) \tag{4.29}$$

in the n2p2 package. The parameters used in this thesis are $q_0 = 0.1$, $q_{\min} = 10^{-6}$ and $\tau_q = 2.302$. More detailed information about the complete training process of the NNP for CO_2 can be found in Chapter 5.
The Kalman filter described so far is only valid if there is only one chemical element, only

energies are considered and not forces and each output node can be directly compared to the reference data. Especially the last point is difficult if a high dimensional NNP that uses the sum of atomic energy contributions (equation 4.15) is employed. The reason being that summing over atomic contributions to the total energy is only a mathematical trick used out of necessity and can't be reproduced by using DFT or force field calculations. This means that the NNP is trained to reproduce the sum of all atomic energy contributions [21].
The forces can be treated similarly to the energies because of their analytical relationship shown in equation 4.19. In the case of n2p2 it is randomly chosen if a single force component or an energy from training configuration i is used for each Kalman filter update. This means that there is one error vector $\xi(t)_{\mathrm{E}}$ for energy updates:

$$\xi(t)_{\mathrm{E}} = [E_i^{\mathrm{ref}} - E_i(\mathbf{a}(t))], \tag{4.30}$$

where E_a^{ref} is the reference energy for configuration i and $E_a(\mathbf{a}(t))$ is the NNP prediction of the energy of the same configuration and one error vector $\xi(t)_{\mathrm{F}}$ for Force updates:

$$\xi(t)_{\mathrm{F}} = [\beta(F_{ij,\alpha}^{\mathrm{ref}} - F_{ij,\alpha}(\mathbf{a}(t)))], \tag{4.31}$$

where $F_{ij,\alpha}^{\mathrm{ref}}$ is the reference force component α for atom j in configuration i and $F_{ij,\alpha}(\mathbf{a}(t)))$ is the NNP prediction for the same force component that can be analytically derived from the energy of the same configuration using equation 4.19. Additionally the factor β, called force weight is introduced. The force weight makes it possible to adjust the relative importance of each force update for the training. A value of $\beta = 10$ was used in this thesis. Using this approach the Kalman filter algorithm in equations 4.22-4.25 will update the combined weight vector $\mathbf{a}$ minimizing the combined cost function,

$$\Gamma = \sum_{i=1}^{M}(E_i^{\mathrm{ref}} - E_i(\mathbf{a}(t)))^2 + \beta^2 \sum_{i=1}^{M}\sum_{j=1}^{N_i}(F_{ij,\alpha}^{\mathrm{ref}} - F_{ij,\alpha}(\mathbf{a}(t)))^2, \tag{4.32}$$

where N_i is the total number of atoms in configuration i. It is advisable to choose a ratio of about 50/50 for force and energy updates to ensure a balanced training.
Different chemical elements are viewed basically independently in n2p2 by using per-element Kalman filters that are not depending on each other. Due to the use of independent atomic NNs in the high dimensional NNP the NN weights for each element do not depend on each other and so the contributions to the energy of one element are not depending on the other element(s).
Another extension of the Kalman filter used in n2p2 is the multi-stream extended Kalman filter. The goal of the multi-stream extended Kalman filter is to utilize the parallel computing powers of modern multi-core processors. One important difference between the classical use of Kalman filters and the use for NN training is that when training the NN all the information that is needed is already available at the beginning. This is usually not the case in engineering applications of the Kalman filter. The multi-stream extended Kalman filter uses multiple training configurations (multiple streams) with the same weights, where each calculates a contribution to a shared weight update. This new weight update then should optimize the weights depending on different configurations at the same time. Because the information is taken from multiple independent configurations at the same time it can be parallelized on multiple processor cores and should yield better training results [2].

5 Training a Neural Network

In the previous chapters, the training process of a neural network potential was already mentioned on multiple occasions. In this chapter it will be explained in detail how a high dimensional NNP was trained in order to reproduce the PES of CO_2 over a wide range of pressures and temperatures.
First the creation of a suitable training data set will be discussed, beginning at how the first training configurations were created using LAMMPS [3] and the flexible EPM2 force field [5]. It is followed by an explanation of how VASP [6, 7, 8, 9, 10] was used to recalculate the energies and forces of these configurations so they could be used for the training of the NNP with ab-initio precision.
An introduction to n2p2 [61] will be given in section 5.2. The software package n2p2 was used for both the training process of the NNP and as plugin to the MD program LAMMPS. This chapter will be concluded by a description of the actual training process, explaining in detail how the tools provided by n2p2 were used.

5.1 Creating the training data

In order to train a high dimensional NNP a set of suitable training configurations is needed. One training configuration contains the information about the positions of all atoms in a system of CO_2 at a certain temperature and pressure at one point in time.
In theory there are multiple ways to create these configurations. One could just randomly place CO_2 molecules in randomly sized boxes or even just do the same with individual C and O atoms that are not bound as a molecule from the beginning. But this is not a suitable approach, because the goal of the NNP being trained in this thesis is to reproduce a certain part of the phase diagram of CO_2. This high pressure domain is mostly populated by solids states of CO_2. Therefore teaching the NNP a lot about randomly placed systems, almost all of which are going to be in the liquid or gas state, if they exist at all, is not going to help achieve this goal. That is why MD simulations with the fully flexible form of the EPM2 force field are used to create configurations that are more similar to what an actual system of CO_2 molecules might look like at a certain pressure and temperature. These configurations are then recalculated with VASP in order to achieve better precision for the energy and forces of each configuration.
It is important to keep in mind that the NNP is not good at extrapolating but rather at interpolating between the data points it was trained with. The NNP is nothing more than a fit with a very flexible functional form. In figure 5.1 a simple example for the difference between interpolating and extrapolating in a fit is given.
Within a certain part of the phase diagram the NNP can interpolate the forces and energies for configurations it has not yet seen if they are similar enough to the training data. However once a configuration is too different to the training data the results of the NNP may increas-

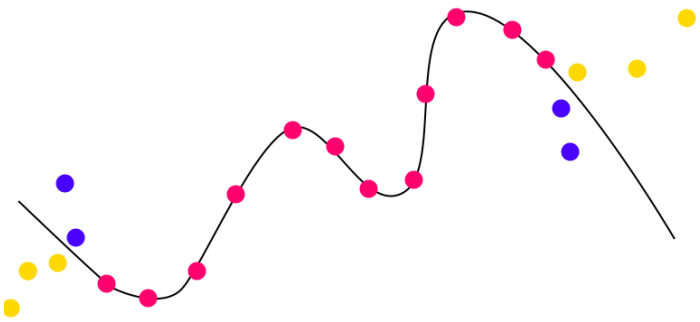

Figure 5.1: The black line represents a fit that uses the red points (representing measurements) that are displayed. Despite the red points only being available in a small interval the fit itself is a function that usually is defined on a much larger domain. If more measurements would be taken they could for example behave like the yellow or the blue points. The fit would not know that though and in this example fail to extrapolate outside of the interval spanned by the red points. However if a new measurement would be made between two of the red points chances would be very high, that the new point would be close to the fit. This would be an interpolation. This graphic represents the idea behind the NNP being good at interpolating but not extrapolating.

ingly diverge from what the reference method would produce until at some point they end up being random. Unfortunately compared to a 2D fit as sketched in figure 5.1 it is more difficult to find out the actual range in which the NNP's predictions can be trusted. One method involves the use of two NNPs that are nearly identical and are trained with the same training data. This will be explained in section 5.3.7.

5.1.1 Creating configurations with EPM2

The first step in creating a suitable training set for the NNP is to do MD simulations with the fully flexible EPM2 force field. The idea is that when doing many long MD simulation with this force field the configurations the system goes through are representative enough of actual CO_2 configurations for the NNP to gather enough information to reproduce the PES. The MD simulations should be so long that 10,000-100,000 steps (at a usual time step of 1 fs) are between each configuration chosen for the training. This is to ensure that the training configurations are different enough from each other to get the maximum information out of each configuration. The actual forces and energies of the configuration will later be calculated with ab-initio precision by VASP. The problem with just doing the whole process with VASP is that it would be too computationally costly to do all the hundreds of thousands of MD simulation steps needed for a few training configurations. Therefore it is more economic to sample the phase space using an MD simulation with an empirical force field. At the end of the simulations the handful of saved configurations that will actually be used to train the NNP can then be fed to VASP, which just recalculates the energies and forces.
In total the phase space was sampled simulating hundreds of millions of femto seconds of tra-

jectories at different pressures and temperatures. The idea was to simulate along of isothermals and varying the pressure throughout one simulation to get a thorough sampling of the relevant part of phase space. The simulation parameters can be seen in table 5.1
For the beginning of the simulations perfect crystals with the structures presented in Chapter 2 were used for each of the five phases. They were then equilibrated to the starting point of the simulation using a combination of Langevin and Nosé-Hoover thermostats. It turned out that when doing simulations of CO_2 with the flexible EPM2 force field, the first equilibration had to be done using a Langevin thermostat or the kinetic energy of the system would be quickly transformed into potential energy which led to large deviations in pressure and density when compared to experimental data. After the correct equilibration the EPM2 force field was tested to be within the expected range of the experimental values when comparing pressures at a set temperature over a wide range of temperatures and volumes.
Additionally a few MD simulations of the liquid phase were performed by melting a CO_2-II crystal with 52 molecules and then equilibrating to the right temperature and pressure the same way as with the solid phases. The liquid configurations were included to give the NNP more flexibility. The issue with this approach is that there is no guarantee that the configurations sampled with EPM2 are close to actual configurations, meaning are still valid samples when recalculated by VASP. One wants configurations that are ideally slightly out of equilibrium but not completely random. This is also why the flexible version of the EPM2 force field was important to use. Due to the flexible angle and bond length of the molecule there are configurations with non equilibrium bond lengths and angles. This means that the NNP has information of what the potential between the atoms within a molecule looks like at different inter atomic distances/angles. The same is done with the molecules. By varying pressure and temperature all sort of different inter molecular distances are sampled. In the end the fit is better the more different points in phase space within the relevant region are sampled.
Due to the flexible empirical force field coupled with the long simulations through out the phase diagram of CO_2 it was assumed that enough random variation would be included in the training configurations produced. An additional step to assure enough variation would be to take the training configurations produced by the empirical force field and add little random displacements to some of the atoms. This was tried at first but later deemed unnecessary when it showed no significant increase in the fit quality of the NNP once a sufficiently large part of the phase diagram was sampled. When doing small tests with only a few different configurations and therefore very limited information about the system, this approach did improve the NNPs predictions though.
After all the MD simulations with the flexible EPM2 force field were finished, 400 configurations were chosen randomly from the total pool of possible configurations (several ten thousands throughout all phases). The only condition for the choosing was that the ratio of the different phases was proportional to their relative area in the p-t phase diagram (see figure 2.2) in order to get a larger number of training configurations representing phases with a wider range in pressure and temperature.
The chosen configurations were then directly used to train a preliminary NNP to see whether the training data could be used to reproduce the EPM2-PES. This was done to see if there were fundamental flaws in the training data before using the computationally expensive DFT method to improve the training data. After the preliminary NNP worked as expected in a few tests, the training data was finally fed to VASP. Later an additional 60 configurations were chosen from multiple simulations of CO_2-III with 32 molecules at temperatures of 100 K to 500 K and pressures between 10 GPa and 40 GPa, using the method described in section 5.3.7. This lead to a total of 460 configurations in the training set.

Table 5.1: NpT simulation parameters for EPM2. The simulation step size was constant at 1 fs. Each simulation only covered one phase to avoid phase transitions within any simulation. In each simulation the barostat was set to increase the pressure from the starting to the end value. The length of the simulations was chosen so that the barostat increased by 1 bar per 100 simulation steps.

phase	temperature/K	start-pressure/GPa	end-pressure/GPa
fluid	400	0.8	1.32
	600	2	4.85
	800	8	11.18
	1000	12	16.77
	1200	20	27.36
	1400	25	33.1
	1600	27	35.6
I	300	0.44	3.235
	350	0.955	7.4975
	400	1.47	11.76
	450	2.205	11.73
	500	2.94	11.7
	550	4.045	11.66
	600	5.15	11.62
	650	6.545	11.545
	700	7.94	11.47
	750	9.555	11.32
II	450	11.96	14.11
	475	11.97	29.12
	525	18.67	51.04
	600	30.74	48.84
	675	42.22	46.78
III	200	11.92	59.95
	250	11.925	58.625
	300	11.93	57.3
	350	11.935	55.9
	400	11.94	54.5
	450	11.95	53.17
	500	43.64	51.62
IV	950	17.64	39.2
	1000	18.23	37.8
	1050	19.41	36.33
	1100	21.91	35.01
	1150	25.0	33.685
	1225	30.0	31.62
VII	650	11.46	12.64
	700	11.32	14.11
	750	11.17	14.92
	800	11.02	15.73
	850	12.27	16.39
	900	13.52	17.05
	950	15.36	17.565
	1000	17.2	18.08

5.1.2 Recalculating configurations with DFT

```
SYSTEM = CO2 molecules
ISTART = 0                # start new job from scratch.
ENCUT = 910               # equals ENMAX*1.3.
ISMEAR = -5               #-5 is accurate energy calculation for non-metals.
SIGMA = 0.05
IBRION = -1               #-1 = No update,just Energy calculation.
ISIF=0                    # 0 = only position can change.
NSW = 0                   # max number of ionic steps for relaxation
NELMIN=8                  # minimum number of self consistency steps until convergence
PREC = Accurate           # accurate makes forces more accurate but takes more time
EDIFF = 1E-06             # Convergence for force
EDIFFG = 0.01             # breaking condition for ionic relaxation
NPAR = 4                  # total number of cores for parallelization
LPLANE = .TRUE.
NSIM = 4
LWAVE = .FALSE.           # no WAVECAR file (very large file)
LCHARG = .FALSE.          # no CHGCAR and CHG files (very large files)
```

Figure 5.2: The INCAR file defines all the settings used in a VASP calculation. This is a picture of the INCAR file that was used to recalculate the energies and forces of the configurations that were first created through MD simulations with the flexible EPM2 force field.

In order to be able to make NNP predictions with ab-initio precision there first needs to be training data calculated with said precision. In the case of this thesis configurations were first created using a force field approach and then a chosen few training configurations were given to the DFT software package VASP. This means that the atomic positions were assigned by the flexible EPM2 force field but the actual forces and energies were calculated with ab-initio precision. Here it is not important that the configurations were created with a force field and therefore lower accuracy, as long as the information about forces and energies comes from a more precise method. In figure 5.2 all the parameter for the VASP INCAR file can be seen. This is the main file used for any VASP calculation and controls all important settings. Additionally a 3x3x3 grid of K-points was used. This grid was automatically generated by VASP. The POTCAR file that contains all information about the potentials used to describe the two atoms carbon and oxygen in the training configurations was created using the PAW_PBE C_h 06Feb2004 potential for C and the PAW_PBE O_h 06Feb2004 potential for O both using the PBE [11] functional. It is very important that all VASP calculations are done with the same settings. If that would not be the case and for example two identical configurations would have a different total energy that would lead to inconsistencies in the NNP. This could lead to the NNP not correctly predicting energies either in a small area of the phase space or not working at all depending on how many and how large the inconsistencies are. The theory behind ab-initio calculations is sketched in Chapter 4.2. The reason for the small number of training configurations (460) lies in the computational complexity of every single VASP calculation on a 3x3x3 k-points grid.
So far no actual NNP training was involved. However the creation of a suitable set of training configurations can be seen as one of if not the most important part of the training process. The NNP can only be as good as the data that is used to train it.

5.1.3 Limitations of the training data

Due to the nature of this work as a master's thesis a few practical limitations were set. The number of training configurations and the number of atoms within each configuration strongly

influence the quality of the NNP. In this case between 24 and 54 molecules of CO_2 were used in a total number of 460 training configurations. The limitation here is the computational cost of calculating energies and forces of more and especially larger configurations with VASP. It turned out to be a sufficient training set to do MD simulations on solid CO_2 within the part of the phase diagram that was sampled. An in depth analysis of the exact phase space that is accessible through this NNP was not done though. The results of the training of the CO_2 NNP are presented in Chapter 6
With this limited training set the scalability of the NNP was also restricted. Depending on the exact simulation setup MD simulations with 100 to 200 molecules were possible but became quickly unstable once passing 200 molecules. This is an issue that can be attributed to the small training set as well as small number of molecules in each configuration. The small number of molecules and small cell sizes in the training configurations restrict the possible number of molecules in MD simulations with the NNP. The reason therefore could be that possible long range interactions can not be seen by the NNP in those configurations. For the testing of the NNP simulations on CO_2-III were used. The average cell parameters of the configurations of CO_2-III in the training set are $a = 8.3 \pm 0.2$ Å, $b = 9 \pm 0.2$ Å and $a = 11.6 \pm 0.2$ Å. The cutoff radii for the symmetry functions of the NNP are between 5.8 Å and 8.4 Å. This means the NNP takes atoms from the periodic images of the simulation box into account for the local environment of each atom which might hide some details in the long range interactions in the system.

5.2 Training a neural network with n2p2

In this thesis n2p2 was used for the training of the high dimensional NNP for CO_2.
The Neural Network Potential Package n2p2 [61] is a software package developed by A. Singraber to train and employ NNs in the context of high dimensional NNPs. It is based on the methodology proposed by Behler and Parinello [21]. The n2p2 software package is made up from multiple tools most of which were used in this thesis. The package was actively developed for the whole duration of this thesis. Therefore some details in how n2p2 was used might differ slightly from the most recent version.
The n2p2 software package offers multiple functions that are needed to train a high dimensional NNP, as well as a lot of additional functions that make it easier to start the training process.

5.2.1 Configuration of n2p2

On the user end the training of a NNP with n2p2 needs three input files with the following names:

- input.nn
- input.data
- scaling.data

The input.nn file contains all the settings for the program. A picture of the settings file that was used to train the CO_2 NNP can be seen in figure 5.3 and 5.4. The first section of the input.nn file encloses keyword-value pairs, while the second half sets the parameters for all symmetry functions that are used.
The input.data file is created from the training configurations. It contains all the information

```
################################################################################
# DATA SET NORMALIZATION
################################################################################
# This section was automatically added by nnp-norm.
mean_energy  -2.7476173964736977E-01
conv_energy   2.6531755162071505E+02
conv_length   1.1333162178611913E+01
################################################################################

################################################################################
# GENERAL NNP SETTINGS
################################################################################
number_of_elements              2          # Number of elements.
elements                        C O        # Specification of elements.
#atom_energy                    C 0.0      # Free atom reference energy (C).
#atom_energy                    O 0.0      # Free atom reference energy (O).
cutoff_type                     6 0.0      # Cutoff type.
#scale_symmetry_functions                  # Scale all sym. funcs. with min/max values.
scale_symmetry_functions_sigma             # Scale all sym. funcs. with sigma.
scale_min_short                 0.0        # Minimum value for scaling.
scale_max_short                 1.0        # Maximum value for scaling.
#center_symmetry_functions                 # Center sym. funcs., i.e. subtract mean value.
global_hidden_layers_short      2          # Number of hidden layers.
global_nodes_short              20 20      # Number of nodes in each hidden layer.
global_activation_short         p p l      # Activation func. for hidden and output layer.
# normalize_nodes                          # Normalize input of nodes.

################################################################################
# ADDITIONAL SETTINGS FOR TRAINING
################################################################################
epochs                          1000       # Number of training epochs.
updater_type                    1          # Weight update method (0=Grad. Desc. 1=Kalman).
parallel_mode                   4          # Training parallelization (0 = Serial, 1-4 = MSEKF).
update_strategy                 0          # Update strategy (0 = Combined, 1 = Per-element).
selection_mode                  2          # Update candidate selection (0=Random, 1=Sort, 2=Threshold).
memorize_symfunc_results                   # Keep symmetry function results in memory.
random_seed                     50         # Seed for initial random weights and train/test splitting.
test_fraction                   0.1        # Fraction of structures kept for testing.
use_short_forces                           # Use forces for training.
force_weight                    10.0       # Weight of force updates relative to energy updates.
short_energy_fraction           1.000      # Fraction of energy updates per epoch.
short_force_fraction            0.0034722  # Fraction of force updates per epoch.
short_energy_error_threshold    1.00       # RMSE threshold for energy update candidates.
short_force_error_threshold     1.00       # RMSE threshold for force update candidates.
rmse_threshold_trials           3          # Maximum number of RMSE threshold trials.
# repeated_energy_update                   # After force update perform energy update for structure.
# use_old_weights_short                    # Restart fitting with old weight parameters.
weights_min                     -1.0       # Minimum value for initial random weights.
weights_max                     1.0        # Maximum value for initial random weights.
# precondition_weights                     # Precondition weights with initial energies.
# nguyen_widrow_weights_short              # Initialize NN weights according to Nguyen-Widrow scheme.
write_trainpoints               10 5       # Write energy comparison.
write_trainforces               10 5       # Write force comparison.
write_weights_epoch             10 5       # Write weights.
write_neuronstats               10 5       # Write neuron statistics.
write_trainlog                             # Write training log file.
####################
# GRADIENT DESCENT #
####################
gradient_type                   0          # Gradient descent type (0 = Fixed step size).
gradient_eta                    1.0E-4     # Gradient descent parameter eta (fixed step size).
############################
# KALMAN FILTER (STANDARD) #
############################
                                #General Kalman filter parameters:
kalman_type                     0          # type (0 = Standard, 1 = Fading memory).
kalman_epsilon                  1.0E-2     # epsilon (sigmoidal: 0.01, linear: 0.001).
kalman_q0                       0.10       # q0 ("large").
kalman_qtau                     2.302      # qtau (2.302 => 1 order of magnitude per epoch).
kalman_qmin                     1.0E-6     # qmin (typ. 1.0E-6).
                                #Standard Kalman filter parameters:
kalman_eta                      0.1        # eta (0.001-1.0).
kalman_etatau                   2.302      # etatau (2.302 => 1 order of magnitude per epoch).
kalman_etamax                   1.0        # etamax (1.0+).

################################
# KALMAN FILTER (FADING MEMORY) #
################################
                                #General Kalman filter parameters:
# kalman_type                    1         # type (0 = Standard, 1 = Fading memory).
# kalman_epsilon                 1.0E-1    # epsilon (sigmoidal: 0.01, linear: 0.001).
# kalman_q0                      0.00      # q0 ("large").
# kalman_qtau                    2.302     # qtau (2.302 => 1 order of magnitude per epoch).
# kalman_qmin                    0.0E-6    # qmin (typ. 1.0E-6).
                                #Fading memory Kalman filter parameters:
# kalman_lambda_short            0.96000   #lambda (forgetting factor 0.95-0.99).
# kalman_nue_short               0.99950   #nu (0.99-0.9995).
```

Figure 5.3: This is the first section of the input.nn file that was used for this thesis. It defines all settings, including the Kalman filter, to run n2p2. The '#' symbol marks comments.

```
###############################################################################################
# SYMMETRY FUNCTIONS
###############################################################################################

# Radial symmetry function (type 2):
#symfunction_short <element-central> 2 <neighbor> <eta> <rshift> <rcutoff>

# Narrow Angular symmetry function (type 3):
#symfunction_short <element-central> 3 <neighbor1> <neighbor2> <eta> <lambda> <zeta> <rcutoff>

# Radial symmetry functions for element C
symfunction_short C  2 O    0.500   2.000  11.000
symfunction_short C  2 O    0.500   5.000  11.000
symfunction_short C  2 O    0.500   6.000  11.000
symfunction_short C  2 O    0.500   7.000  11.000
symfunction_short C  2 O    0.500  10.000  11.000

symfunction_short C  2 C    0.500   7.000  14.000
symfunction_short C  2 C    0.400  9.000  15.000
symfunction_short C  2 C    0.800  11.000  16.000
symfunction_short C  2 C    0.600  12.000  16.000
symfunction_short C  2 C    0.700  13.000  16.000

# Radial symmetry functions for element O
symfunction_short O  2 C    0.500   2.000  11.000
symfunction_short O  2 C    0.500   5.000  11.000
symfunction_short O  2 C    0.500   6.000  11.000
symfunction_short O  2 C    0.500   7.000  11.000
symfunction_short O  2 C    0.500  10.000  11.000

symfunction_short O  2 O    0.700   4.000  11.000
symfunction_short O  2 O    0.700   5.000  11.000
symfunction_short O  2 O    0.500   7.000  14.000
symfunction_short O  2 O    0.500   9.000  15.000
symfunction_short O  2 O    0.500  10.000  15.000

# Angular symmetry functions for element C
symfunction_short C  3 C  C    0.125  -1.000   3.000  11.000
symfunction_short C  3 C  C    0.125   1.000   3.000  11.000
symfunction_short C  3 C  C    0.003  -1.000   1.000  11.000
symfunction_short C  3 C  C    0.003   1.000   1.000  11.000
symfunction_short C  3 C  C    0.003  -1.000   3.000  11.000
symfunction_short C  3 C  C    0.003   1.000   3.000  11.000

symfunction_short C  3 C  O    0.010   1.000   1.000  11.000
symfunction_short C  3 C  O    0.010   1.000   3.000  11.000
symfunction_short C  3 C  O    0.003  -1.000   1.000  11.000
symfunction_short C  3 C  O    0.003   1.000   1.000  11.000
symfunction_short C  3 C  O    0.003   1.000   3.000  11.000

symfunction_short C  3 O  O    0.010  -1.000   1.000  11.000
symfunction_short C  3 O  O    0.010   1.000   1.000  11.000
symfunction_short C  3 O  O    0.010  -1.000   3.000  11.000
symfunction_short C  3 O  O    0.010   1.000   3.000  11.000
symfunction_short C  3 O  O    0.003  -1.000   1.000  11.000
symfunction_short C  3 O  O    0.003   1.000   1.000  11.000
symfunction_short C  3 O  O    0.003  -1.000   3.000  11.000
symfunction_short C  3 O  O    0.003   1.000   3.000  11.000

# Angular symmetry functions for element O
symfunction_short O  3 C  C    0.010  -1.000   1.000  11.000
symfunction_short O  3 C  C    0.010   1.000   1.000  11.000
symfunction_short O  3 C  C    0.003  -1.000   1.000  11.000
symfunction_short O  3 C  C    0.003   1.000   1.000  11.000
symfunction_short O  3 C  C    0.003   1.000   3.000  11.000

symfunction_short O  3 C  O    0.010  -1.000   1.000  11.000
symfunction_short O  3 C  O    0.010   1.000   1.000  11.000
symfunction_short O  3 C  O    0.010  -1.000   3.000  11.000
symfunction_short O  3 C  O    0.010   1.000   3.000  11.000
symfunction_short O  3 C  O    0.003  -1.000   1.000  11.000
symfunction_short O  3 C  O    0.003   1.000   1.000  11.000
symfunction_short O  3 C  O    0.003  -1.000   3.000  11.000
symfunction_short O  3 C  O    0.003   1.000   3.000  11.000

symfunction_short O  3 O  O    0.010  -1.000   1.000  11.000
symfunction_short O  3 O  O    0.010   1.000   1.000  11.000
symfunction_short O  3 O  O    0.010   1.000   3.000  11.000
symfunction_short O  3 O  O    0.003  -1.000   1.000  11.000
symfunction_short O  3 O  O    0.003   1.000   1.000  11.000
symfunction_short O  3 O  O    0.003   1.000   3.000  11.000
```

Figure 5.4: This part of the input.nn file is used to set up the symmetry functions which are the input for the atomic NNs. The symmetry functions called type 2 and 3 here are the same as the ones called type 1 and 2 described in equation 4.16 and 4.17. The cutoff radius is in units of Bohr.

about forces and energies, as well as atomic coordinates of each configuration in the training set.
The scaling.data file is created after input.data and after the symmetry functions in input.nn are set. It holds the information about the distribution (min, max, mean and standard deviation) of values the symmetry functions yield for each atom of each configuration. This information is used by the training algorithm.
This means that first the input.data file has to be created using suitable training configurations. Then input.nn is set up so that the chosen symmetry functions and other settings are in agreement with the training data. Afterwards scaling.data is created using the two previous files.
The keywords in input.nn have to be manually set depending on how the user wants to train the NNP. Some of the settings seen in figure 5.3 are sufficiently explained by the comments to their right or the documentation of n2p2 [61]. For example *number_of_elements* stand for the number of chemical elements in the simulation, in this case there are carbon and oxygen, so two. Others like the *parallel_mode* are very technical and influence the performance of the code and not the actual training. What follows is a short summary of the most important keywords and short comments as to why specific settings were chosen for this thesis. Often the settings that were used were chosen based on trial and error because it is very difficult if not impossible to find analytical leads as to which values to choose for certain parameters. This also means that there might be better options for some of the settings, but the chosen settings were shown to produce a solid NNP for CO_2. See Chapter 6 for a comparison between this NNP and a reference DFT simulation. For further information please refer to the online documentation of n2p2 [61].

cutoff_type

This keyword sets the cutoff function f_c that is used for the symmetry functions (see 4.16 and 4.17). The first value (here 6) sets the type of symmetry function, while the second number (here 0.0) sets the inner cutoff up to which f_c is defined as 1. Cutoff type number 6 has the form

$$f_c = ((15 - 6x)x - 10)x^3 + 1 \tag{5.1}$$

The other available types of cutoff functions can be found in the documentation of n2p2. The choice of the cutoff function is an individual choice and was changed multiple times throughout the work on this thesis. However for the NNP of CO_2 presented here there were no significant differences between the different cutoff functions that were tried out.

global_*

For this thesis a NNP with atomic NNs that have two hidden layers with 20 nodes each and one output node are used. The number of input nodes depends on the number of symmetry functions that describe the local environment of each element. In figure 5.4 it can be seen that there are 30 symmetry functions for C atoms and 29 symmetry functions for O atoms. The more hidden layers and nodes per such hidden layer the NN has the more complex of a fit it can produce. However a more complex NN also needs more information to be able to produce a reasonable fit. In this thesis the 29-20-20-1 and 30-20-20-1 NNs were shown to produce adequate fits for the number of training configurations and the complexity of the CO_2 systems that were used. This does not mean that this is the best possible setup, just the best setup that was found through trial and error in a limited time. The activation functions that are needed to be able to fit nonlinear functions are chosen to be $p = \ln(1 + \mathrm{e}^x)$ going into the two hidden layers and $l = x$ going into the output layer.

selection_mode

The selection_mode determines how the next energy or force for the training within an epoch is chosen. If mode 0 is chosen then the next energy or force is chosen randomly from all possible ones in the training data. If mode 1 is chosen then the configurations within the training data are sorted by the RMSE of their energy/forces, depending on which is being trained, and then the one with the highest RMSE is chosen for the next update. When choosing the last option, mode 2, a mix of both systems is employed, where multiple random forces/energies are looked at and the first one with an RMSE above a certain threshold is chosen. Mode 2 was chosen for this thesis because it takes into account that using configurations with a large RMSE is important for the next update for a faster and better convergence of the training. It also chooses randomly between a few configurations which makes sure that not one configuration is used for the training a lot more often than others and therefore certain local minima can be avoided.

memorize_symfunc_results

This is a keyword that is activated without an additional value. If activated the values of all symmetry functions are stored and reused when needed. This means a large amount of values is saved in the RAM because each symmetry function is evaluated once for each atom in each configuration. Therefore the needed RAM usually is in the tens of Gigabytes and this keyword should be used with care otherwise one might exceed the available RAM of the machine the training is run on. The tool nnp-scaling calculates the approximate amount of storage needed, see section 5.3.3.
The advantage of activating this function is not having to recalculate the same symmetry functions every epoch or even update. This usually drastically decreases the time needed to train a NNP. The time saved depends on the size of the training set and the number of symmetry functions. If possible this keyword should always be activated. As long as the necessary RAM is available there are no downsides to using this function.

force_weight

The force weight defines the relative importance of force updates compared to energy updates. Please see Chapter 4.3.5 for more information on the force weight parameter for the Kalman filter. It was found by trial and error that a force weight of around β=10 (see equation 4.32) is a good approach. Especially for MD simulations the forces between each atom are more important for good results than the energy alone. The force weight ensures that there is enough emphasis on the force updates throughout the training process.

short_*

The keywords starting with short_ are responsible for how many updates of energies and forces are done per epoch respectively. For the short_energy_fraction a value of 1 should be chosen, so that all energies in the training data are used for each training epoch. Because there is only one energy but $3N$ force components per training configuration, where N is the number of atoms in each configuration, the short_force_fraction should be set in a way that there is approximately a 50/50 chance to have either a force or an energy update. This means that all energies but only a randomly chosen fraction of the forces is used for the training each epoch. The previously mentioned force_weight then is responsible for giving forces more overall importance in the training. The two threshold keywords are responsible for the RMSE threshold used if selection_mode equals 2.

write_*

These keywords determine how often the different output files of the training are updated. The first number states after how many epochs the updates happen and the second number is responsible for writing output every epoch up to the stated number. Here for the first five epochs the output is written to the file every epoch and after that the output is written every tenth epoch.

Kalman filter keywords

The Kalman filter parameters have in theory been covered in Chapter 4.3.5 and are consistent with the parameter names introduced in that chapter. A detailed approach as to how to chose Kalman filter parameters is described by Singraber [2]. In this thesis the choice of Kalman filter parameters was based on Singraber's results. A few different parameter combinations were tried out over the course of the work on this project and the one with the best results were taken. The chosen Kalman filter parameters can be seen in figure 5.3.

5.3 Training the neural network potential

The software package n2p2 offers multiple tools that can be utilized to first prepare and then carry out the training of a NNP. In this section all the relevant tools will be given a short introduction and explanation as to how they were used for this thesis. All modules are named with the prefix *nnp* and then a short name. Not all modules had to be used, because n2p2 can be applied to a wider range of problems than just the training of a NNP. Generally to train a NNP first the data-set for input.data has to be created, then the settings in the input.nn file have to be adjusted the first time. Next the symmetry functions need to be set in the input.nn file. It should be noted that the settings and symmetry functions in input.nn can be changed throughout the steps of the training process and do not have to be perfect at the beginning. The last step before beginning the training of the NNP is to create the scaling.data file. The following sections will present the n2p2 tools used in the order they were applied. Before using any of those tools the input.nn and input.data files need to be prepared. For additional information see the online documentary of n2p2 [61].

5.3.1 nnp-norm

In the first step the input.data and input.nn files are needed. The program nnp-norm calculates the mean energy, force and volume per atom for each atom of each configuration in input.data as well as the corresponding standard deviations. It then transforms all energies, forces and lengths to normalized internal units and creates a new input.data file as well as a header in input.nn that shows the conversion factors (see figure 5.3 at the beginning of the input file). This is done because the settings for the Kalman filter depend on how large the values are that it encounters in the training data. Those settings can be more generally applied if a normalized data set is used. The normalization and rescaling of the input for NNs is not necessary and in theory the same training should be accomplished in both cases. However for practical purposes it is encouraged to do so and can it lead to a faster training and a lower chance of getting stuck in a local minimum.

5.3.2 nnp-dist

In order to find a suitable set of symmetry functions it can be helpful to look at the radial and angular distribution functions (RDF and ADF, respectively) of the training data. This can be easily done by using nnp-dist. It is a straightforward way to calculate the desired distributions for each combination of two or three atoms. Once nnp-dist was used the RDF can be plotted and overlaid with different radial symmetry functions. The process of choosing the right symmetry functions builds on trial and error and symmetry functions have to be found that fit to the overall shape of the RDF, meaning that their maxima are at relevant points, where the RDF has a significant structure and has not yet converged. The number of symmetry functions needed depends on the complexity of the distribution function of the system but the exact number to use for the later training has to be found by trial and error. It is also important not to choose a cutoff that is too large because this will slow down the training as well as the MD simulations that are done later. Those simulations also depend on calculating the symmetry functions in every time step. The results of plotting the RDF of all training configurations used in this thesis and then overlaying the radial symmetry functions can be seen in figure 5.5. This approach works better for the radial symmetry functions and not as well for the angular symmetry functions that depend on angles as well as distances and can't be visualized as easily. For the angular symmetry functions first a large set of parameters is chosen. In this thesis the first set of parameters for the angular symmetry functions was chosen based off the angular symmetry functions that were used for the NNP for water [1]. Those symmetry functions are then reduced using nnp-prune, which will be explained in the its own section.
It appears that there are many equivalent sets of symmetry functions that yield comparable training results and there is no right set of symmetry functions.

5.3.3 nnp-scaling

The nnp-scaling tool is used to analyze the chosen symmetry functions in regards to the training data in the input.data file. It calculates the values of each symmetry function for each atom in the whole data set and produces histograms about the symmetry function values and neighbor distributions. The file scaling.data is produced as well and contains the min, max, mean and standard deviation for each symmetry function. It also gives an estimate about how much RAM would be needed to store all symmetry function values at run time, which drastically increases performance because otherwise the same symmetry functions have to be calculated over and over during the training. Whenever changes are made to the symmetry functions or training data nnp-scaling has to be run again.
The scaling.data file that is produced has two purposes. It is needed for the training because the training tool accesses the symmetry function statistics and it is very helpful when doing MD simulations with the NNP by enabling extrapolation warnings. Extrapolation warnings are triggered whenever the current value of a symmetry function is outside the interval between the minimum and maximum value listed in scaling.data. The idea is that by leaving that interval there needs to be some sort of extrapolation done by the NNP instead of interpolation. In the case of many extrapolation warnings during an MD simulation the NNP might not have been trained to correctly fit the current state of the system. In this case additional training might be necessary.

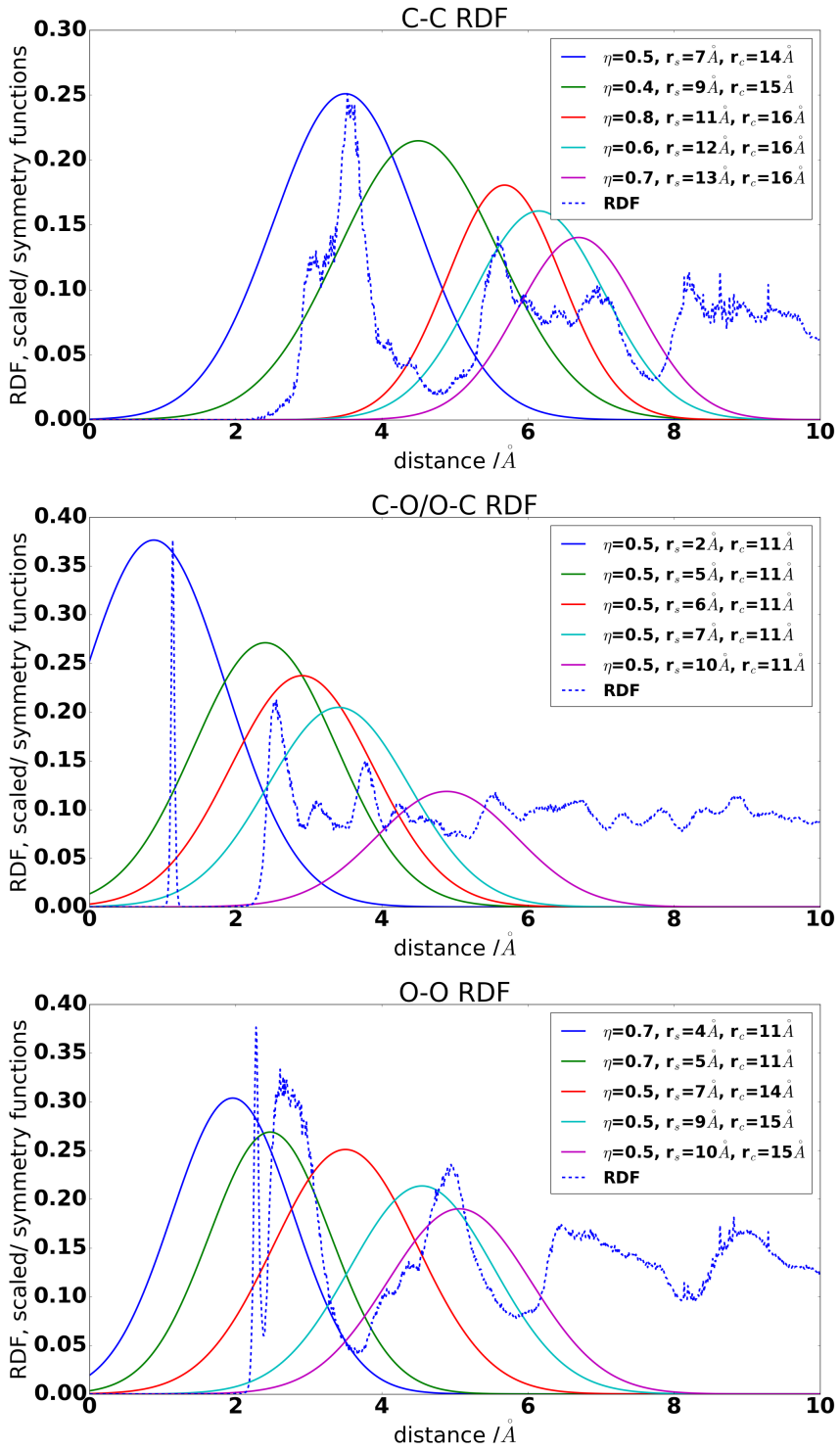

Figure 5.5: The radial distribution function (RDF) calculated from all configurations in the training set and the set of radial symmetry functions used in this thesis. The symmetry functions are chosen in a way that they cover the significant part of the RDF. The RDF for CO_2 can be split in three parts, C-C, C-O and O-O. The symmetry functions need to be representative of each of the parts. It is important to note that there needs to be symmetry functions for C-O as well as ones for O-C. Their values are identical but the NNP does distinguish which atom is at the center of the symmetry function. The RDFs have been scaled to better coincide with the symmetry functions.

5.3.4 nnp-prune

In section 5.3.2 it was already mentioned that the nnp-prune tool is important to select suitable symmetry functions. Especially when considering angular symmetry functions it is hard to select them according to the angular distribution function. Therefore it is important to use the pruning tool, which requires the scaling.data and input.nn files. The pruning works by simply calculating the range of the symmetry functions that were used in creating scaling.data by subtracting the minimum from the maximum value. The idea is that symmetry functions that have a larger range of values for the whole data set contain more information about the structures that are present in that data set. Therefore the symmetry functions with a range below a threshold that can be set by the user are commented out in the input.nn file by nnp-prune. The symmetry functions that are removed this way are then no longer used.
This means that for a large set of angular symmetry functions the threshold can be increased until only about 20 are left. This can also show simple patterns in how certain parameters influence the range of the symmetry functions for both angular and radial distribution functions.
For this thesis a combination of first nnp-prune and then adding symmetry functions with similar parameters as the ones that were left after the pruning was used for the angular symmetry functions. For the radial symmetry functions a set of parameters was already found using the radial distribution function and then symmetry functions were only removed with nnp-prune if their range was considerably below average.
It should be mentioned that while it does seem like there is a correlation between the range of symmetry functions and how well they describe an atomic configuration, there still needs to be human supervision. For example a set of symmetry functions that are all very similar to each other and all of which have a large range, can yield worse results than a more diverse set with smaller ranges. The goal is a good description of the local environment of the atoms.

5.3.5 nnp-select

In order to test how well the symmetry functions work the nnp-select tool is very useful. This tool should also be used to refine the settings in the input.nn file, for example to optimize the Kalman filter parameters on a smaller training set. All it does is select a sub set of training configurations from input.data and creates a new input.data with a smaller number of configurations. This way quick local tests can be done and different setups for the training compared without having to spend a lot of time before finally committing to training the actual NNP.

5.3.6 nnp-train

Once the symmetry functions and training configurations are chosen the actual training algorithm can be started. The nnp-train program of n2p2 requires the input.nn, scaling.data and input.data files. It starts the optimization process of the NN weights using the chosen method. In this thesis the standard Kalman filter with multi stream enabled for better parallelization was used. Once the training is started the RMSE of energy and forces should on average decrease with each iteration. The evolution of the RMSE for the NNP trained for this thesis can be seen in Chapter 6. It is to be expected that the RMSE does increase in the case of over-fitting or when leaving a local minimum for the training set. A high increase of the RMSE can be a sign of a failed training attempt though. This can especially happen if there are some inconsistencies in the training set. For example there was a mistake made in the transformation of the non orthorhombic phase IV of CO_2 between the VASP output

format and the n2p2 input format. This led to some atomic positions being wrong. The training routine first seemed to correctly optimize the RMSE of energy and forces but after a few epochs the RMSE would increase and then diverge. This behavior can be seen when conflicting information is presented to the NN.
The number of epochs needed for the training depends on the chosen Kalman filter parameters as well as the structure of the atomic NNs and the training data that are used. In figure 6.3 the evolution of the RMSE over multiple epochs can be seen. At the end of that training the RMSE for both energy and forces has converged in a reasonable way although there can still be oscillations seen.
In figure 6.1 and 6.2 the training and test energies and forces are shown for multiple training epochs. The closer the predicted values are to the identity line the better the training of the NNP is. Here small outliers can still be seen although the RMSE of energy and forces is converged. This means that there are still some configurations in the training set where the NNP predictions are not as good as for others.

5.3.7 nnp-comp2

After the NNP is trained with ab-initio data it should be able to make predictions in nearly the same precision in the area of phase space it is trained for.
However it is entirely possible that some areas that the NNP was thought to be trained for are in fact not. In order to find those areas a second NNP is trained identically in every aspect except for different initializations for the pseudo-random number generation. Then both NNPs can be used to predict the forces and energy of the same configuration. Whenever the energy or forces for a configuration differ a lot between the two NNPs the configuration should be included in a new training set, because the NNP seems to be extrapolating, which means it assigns more or less random energies and forces to those configurations.
This process can be easily executed using the n2p2 function nnp-comp2. This tool automatically compares energies, forces and also extrapolation warnings for two NNPs for a given set of configurations. It can also be used on any set of configurations for example an MD trajectory that was calculated by one of the NNPs. In that case every saved configuration from that trajectory is fed to the second NNP and its predictions compared to the first. Using MD simulation data that was created by the first NNP also means that it is easy to get a feeling for which areas of phase space the NNP is trained sufficiently and for which it is not.
This step closes the circle of training an NNP. After choosing configurations for which the energy predictions are not satisfactory they are fed to the DFT software which then calculates the energies and forces with the same ab-initio precision as the rest of the training data. Then the new training configurations can be used in an enhanced training set together with the old training configurations, forming a new and larger input.data. Now the process of training starts again alternating between nnp-train and afterwards nnp-comp2 until the desired results are reached.

6 Validating the Neural Network Potential

The stated goal of this thesis is to train a NNP for CO_2 with ab-initio precision. There are two criteria one has to look at when talking about whether the training of the NNP was successful. First the RMSE (root mean square error, see equation 4.26) for energies and forces needs to converge during the training. Second the MD simulations using the NNP must be able to replicate the DFT calculations.
In this chapter both of these criteria are being investigated and it is concluded whether MD using the NNP and MD using DFT are producing similar results for the static property of the RDF and the dynamic property of the VACF.

6.1 Analysis of the NNP training

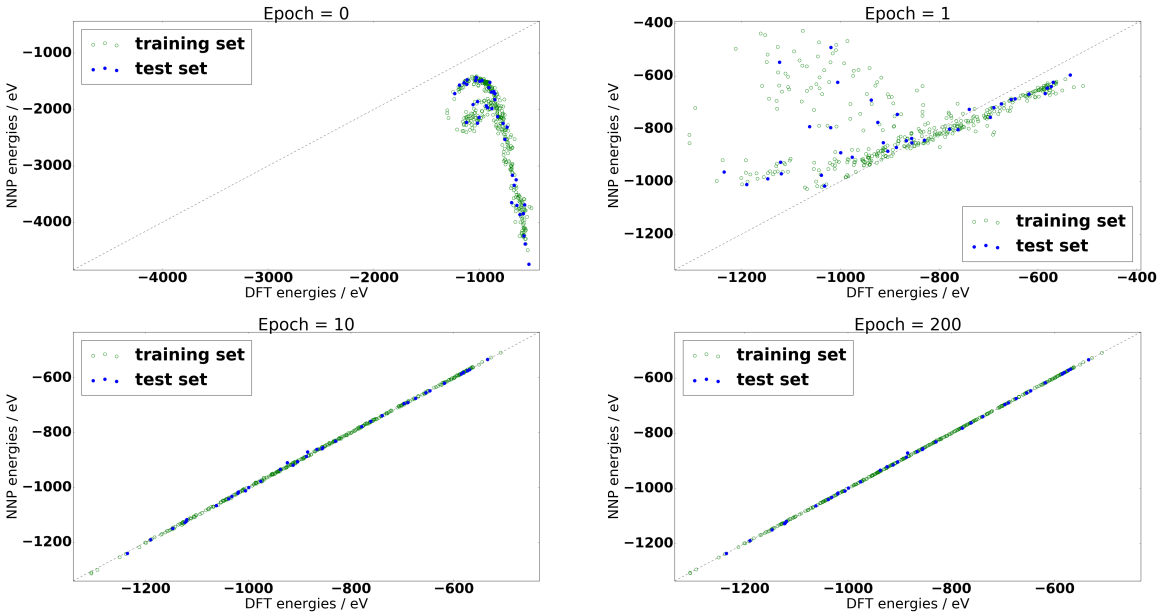

Figure 6.1: The energies predicted by the NNP (y-axis) corresponding to the energies calculated using DFT for each configuration in the training set as well as the test set. While the distribution of the energy value pairs is somewhat random because of the initialization of the NN in epoch zero, with each subsequent epoch the energies converge more towards the identity function $y = x$ that is indicated by the dashed line. For a perfectly trained NNP all dots should lie perfectly on the dashed line. At epoch 200 a reasonable distribution is found for the NNP used in this thesis.

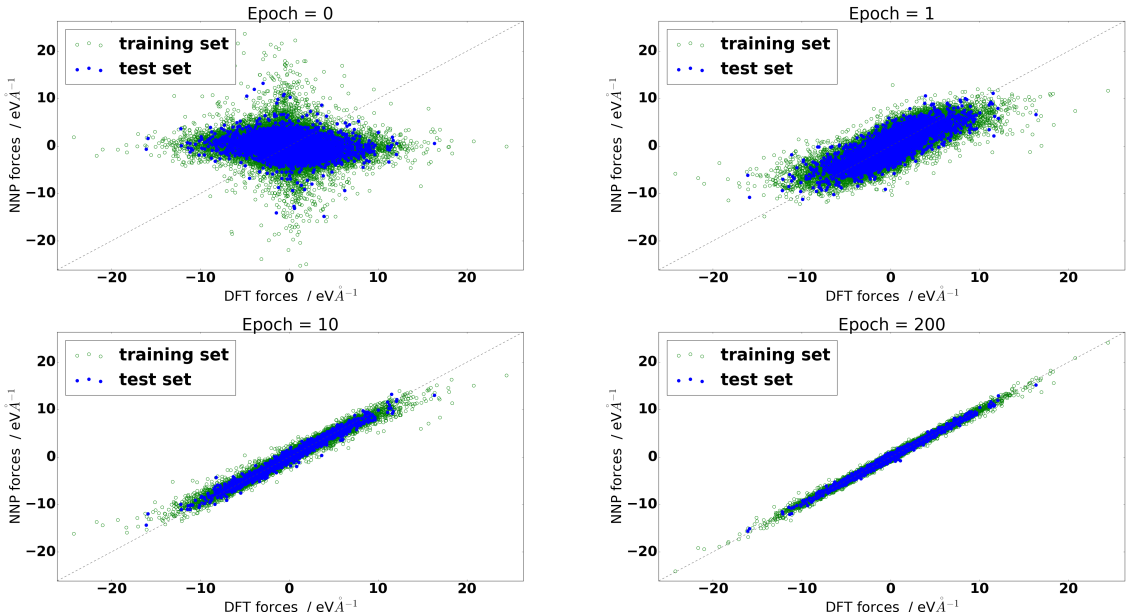

Figure 6.2: Comparison of each force component in the training and test sets as predicted by the NNP with the DFT calculations. The qualitative behavior is the same as described for energies in the previous figure. However there are more force components than energies, because each configuration provides one energy value but $3N$ force components, where N is the number of atoms. Due to the energies being trained more evenly the energies converge towards the dashed identity function less precisely.

In the previous chapter the training process for a NNP was described. In this section the results of the training are investigated by looking at the convergence of the energies and forces of the NNP. For the NNP to reproduce the PES of the training data it is necessary that the RMSE of energies and forces converges. The energies and forces predicted by the NNP should also be corresponding to the energies and forces calculated by the DFT in a linear way. This is shown in figure 6.1 and 6.2. What can be seen in these figures is the prediction for test and training energies (6.1) and forces (6.2) made by the NNP for all configurations on the y-axis and the corresponding values for energies and forces from the DFT calculations on the x-axis.
Each dot in these figures represents one energy (or force component) with the x-value being the DFT calculation's value and the y-value being the NNP's prediction. This is displayed for each configuration in the training and test set, where the training set are the configurations used for the training of the NNP and the test configurations are only used to test how well the NNP predictions coincide with the DFT calculations but are not used for the training. The number of energies displayed is the same as the number of configurations in the test plus training set because per configuration there is one energy value for the system. The number of force components is a lot higher because there are three force components per atom per configuration.
The ideal outcome would be that the DFT/NNP value pairs lie on the identity function $y = x$. This means that there is a one to one transformation between the DFT and NNP values.
For the epoch zero where the NNP weights are first initialized without having seen the training data yet, the corresponding predictions by the NNP are independent of the DFT values, as can be seen in the figures 6.1 and 6.2. For energies and forces alike. This means that a small DFT value can correspond to large NNP values and vice versa. In the first training epoch,

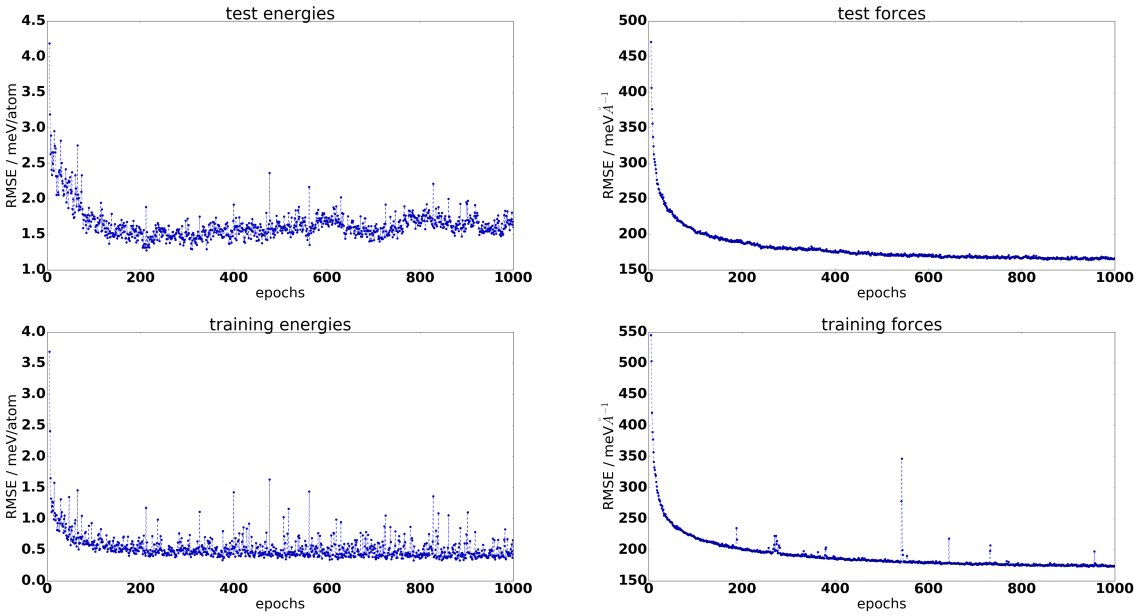

Figure 6.3: Convergence of the RMSE for test and training energies (on the left top to bottom) as well as test and training forces (on the right top to bottom). While the force RMSE for both sets only begins to converge towards epoch 1000 the energies converge much quicker. The test energies even show signs of overfitting after around epoch 200, where the minimum RMSE is found. In all 4 figures the first 5 training epochs are omitted in order to be able to distinguish details in the rest of the data. This is because those first RMSE values are magnitudes larger than the later values and carry relatively little meaning.

after each training configuration was used once to optimize the NNP weights it can already be seen that the energy and force value pairs start to move along the identity that is indicated by the dotted line. This trend then continues for further training epochs until for epoch 200 the energy predictions are all lying close to the identity function. The same holds for the forces with the difference that for the forces there is a larger spread around the identity line. This happens despite an equal split between training energies and forces each epoch.
The other important information about the training is how the RMSE of energy and force predictions evolves with increasing epochs. This contains information about how well on average the energy and force predictions of the NNP coincide with the DFT calculations. The RMSE for energies and forces over a time span of 1000 epochs can be seen in figure 6.3.
In this figure it is shown that it needs the full 1000 epochs for the training and test forces to converge, while the training energies already converge after a couple of hundred epochs. The most interesting behavior is shown for the test energies though. The RMSE for test energies shows a systematic increase after around 200 epochs. The test energies are a small subset of all available configurations that is never used for weight optimization but only to see how good the NNP predictions are for configurations it has never encountered before. An increase in the RMSE for test energies means that the NNP starts to be in the domain of overfitting for its energy predictions.
Overfitting in this case can be seen as the NNP losing flexibility in its predictions and starting to memorize the specific configurations in the training set. Therefore it still decreases the RMSE for the training energies but its ability to make predictions about new configurations that are not in the training set decreases.
Before using the NNP for MD simulations and comparing the results to DFT calculations

it should be made sure that the RMSE for the test set are converging as well as that the DFT/NNP value pairs for energies and forces are converging towards the identity function. Therefore the weights for the NNP MD in the next section are taken from epoch 200 where the NNP is still flexible and makes good predictions for the test energies.

6.2 Comparison of DFT-MD and NNP-MD

After confirming that the training of the NNP was successful and the RMSE for energies and forces converges, it is time to investigate weather the NNP does reproduce the PES of the DFT simulations that were used to create the training data. For this purpose three simulations were done on CO_2 phase III, which was chosen due to its relatively large area in the phase diagram and being an orthorombic structure unlike the triclinic phase IV, which makes a lot of secondary calculations simpler. In order to increase the capabilities of the NNP in simulating this phase of CO_2 an additional 60 configurations of CO_2-III were created according to the procedure described in Chapter 5.3.7 and added to the original training set of 400 configurations.
All three simulations were NVT simulations. A Nose-Hoover thermostat with a damping factor of $100 \cdot \delta t$, with δt =1 fs was used to keep the temperature in the system constant on average (see Chapter 3.6.1). Also the number of particles and volume of the cells were kept constant. The NVT ensemble was chosen instead of the NpT ensemble because the DFT-MD software used that is part of VASP is only able to do NVT simulatons due to limitations of the DFT algorithm itself.
The first simulation was a 1750 fs long simulation with 1 fs time steps and only 16 CO_2 molecules. The temperature was set to be T=200 K and the cell volume was V=449.35 Å^3 This simulation was made using VASP's MD capabilities with the same settings previously used to create the training data (see Chapter 5.1.2). The reason for the small number of molecules and short duration of this simulation lies in the computational cost of VASP and DFT-MD simulations in general. For this simulation the Vienna Scientific Cluster (VSC3) was used with one node (= 16 cores) for 72 hours in order to compute this short and small simulation.
The second simulation is identical to the DFT simulation in starting configuration, temperature, number of molecules, duration and step size, with the difference being that it uses the NNP to calculate the energy and forces for each time step. When using the VSC3 for this simulation with the NNP-MD instead of DFT-MD, the calculation only takes 23 seconds, using a laptop with four cores instead of the VSC3. This shows that the NNP does indeed scale much more like an empirical potential than a DFT calculation and therefore longer and larger simulations can be done. The NNP-MD is done using the LAMMPS plugin that is part of the n2p2 software package [47].
In order to showcase the strengths of MD simulations with the NNP a third MD run was done over 25 000 fs with 108 molecules that used a symmetrically expanded version of the starting configuration of the other two simulations with a volume V=3246.55 Å^3 as well as the same temperature. Despite this simulation being much larger than the one done with VASP it could be computed on a laptop with four cores in about 2 hours. Once the simulations were finished the dynamic and static properties of the systems could be compared using the RDF and VACF that were calculated according to the description in Chapter 3.5.1 and 3.5.2.

6.2.1 Radial distribution function (RDF)

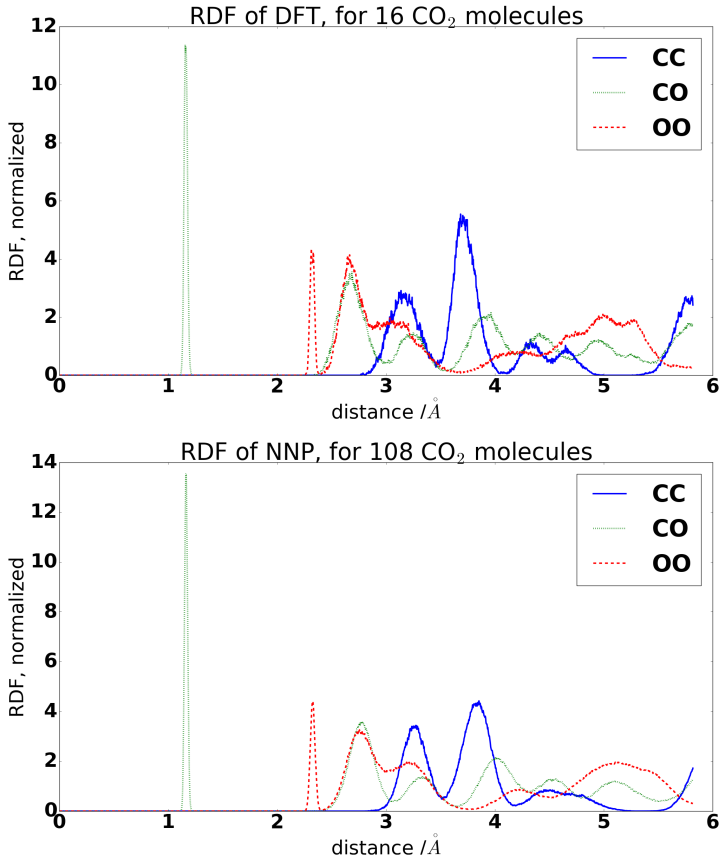

Figure 6.4: The top plot shows the RDF of a system with 16 molecules simulated with DFT. Due to the small simulation box the RDF only goes up to 6 Å and due to the small number of molecules it is not as smooth as the second RDF in the bottom plot. The bottom picture shows the RDF of a simulation using a NNP and 108 molecules. Due to the higher number of molecules the second plot is much smoother. When compared both RDFs show nearly identical peaks up to 4 Å, when they start to slightly diverge because of the aforementioned size restrictions of the DFT simulation. The RDF has been normalized, so that it converges to a value of one for an infinite distance. The height of the first peak (C-O bond distance) was scaled down to $1/4$ in order to better see other structures on the plot.

An important static property of any atomic or molecular structure is the radial distribution function (RDF). It describes how likely it is to find an atom of one type at distance r to a different atom either of the same or of a different type. In figure 6.4 the RDF for two simulations of CO_2-III is compared. The upper plot shows the RDF for the 1750 fs long MD simulation done with DFT that only contains 16 molecules. The lower plot is from a simulation done with the NNP that was longer and contains 108 molecules.
The first noticeable difference when comparing both RDFs in figure 6.4 is that the NNP-RDF is a lot smoother. The reason is that when calculating the RDF averages are considered and with 108 molecules little fluctuations of the individual atomic positions are less influential. On the other hand with only 16 molecules such little fluctuations can be seen in the RDF. The second interesting aspect of the two RDFs is that especially when considering a molecular crystal like CO_2-III the RDF could be calculated for longer distances than just under 6 Å, which is about 5 times the bond length of CO_2 of 1.16 Å and about twice the nearest neighbor distance of

the crystal that was simulated. The nearest neighbor distance is the first peak for the C-C RDF. The reason why the RDF was not calculated for larger distances is that the simulation box for the DFT with only 16 molecules was so small that after a distance of 6 Å the effects of the periodic boundaries could be noticed and the previous peaks started to repeat themselves. For the sake of being more easily comparable the RDF for the NNP simulation was also only calculated up to the same distance.
The important feature to compare between the two RDFs is the position and relative height of the peaks. Up until the CO-peak at around 4 Å these two properties are nearly identical between both figures. After the 4 Å the two RDF start to diverge a little more due to the RDF derived from the DFT simulation becoming less precise. Overall the two RDF do show the same crystal structure and that, at least regarding this static property, both methods seem to be leading to nearly identical results.

6.2.2 Velocity auto correlation function (VACF)

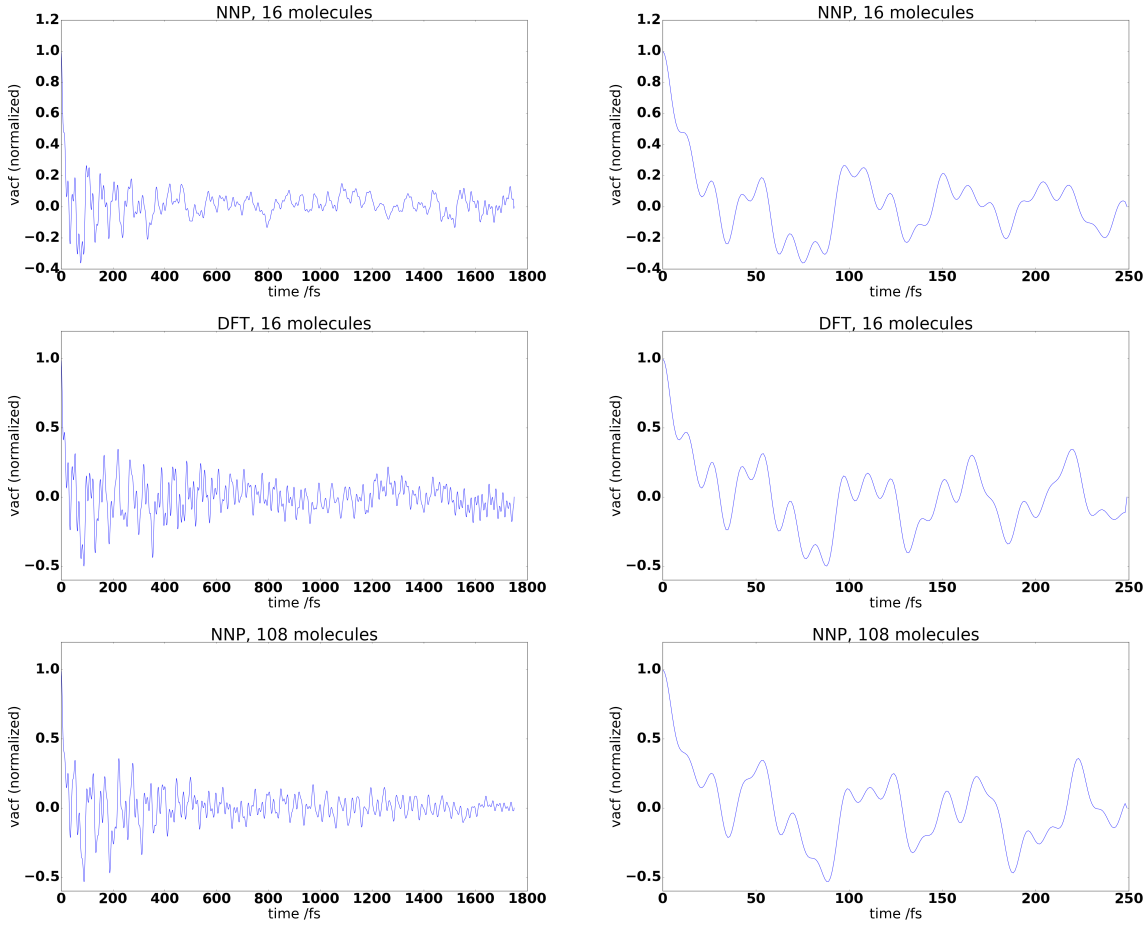

Figure 6.5: The VACF was computed for three simulations over a duration of 1750 steps: (from top to bottom) NNP-MD with 16 molecules, DFT-MD with 16 molecules, NNP-MD with 108 molecules. On the left side the full VACFs are displayed, while on the right only the first 250 steps are shown for better comparison. The structure of all three VACFs is similar over those first 250 steps, although with differences in how pronounced each peak is. For the whole 1750 steps differences can also be detected in the slow overarching oscillations.

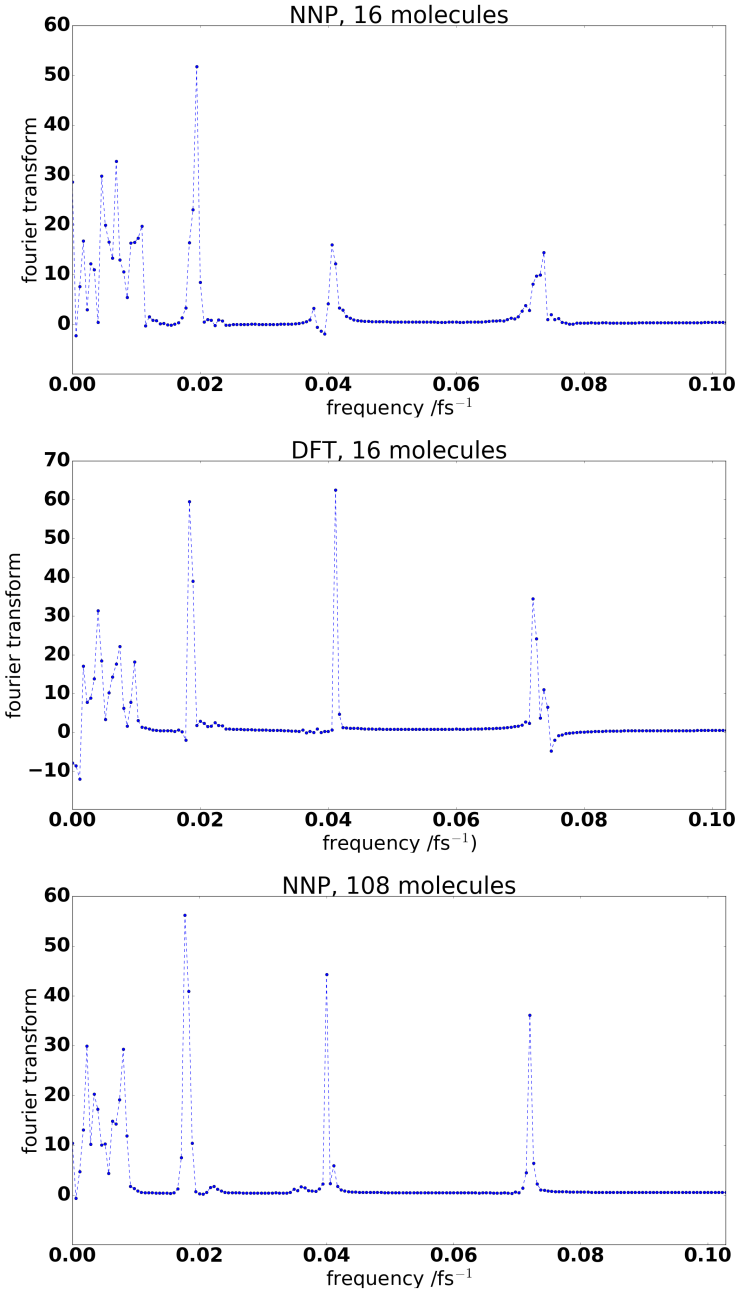

Figure 6.6: The Fourier transforms for all three VACFs over their full lengths are shown for: (top to bottom) NNP-MD with 16 molecules, DFT-MD with 16 molecules, NNP-MD with 108 molecules. The three characteristic peaks of the DFT-VACF are found in both other VACFs, especially the one with 108 molecules. The low frequency area in which four peaks can be seen for the DFT-VACF could not be reproduced as well although the NNP-VACF for 108 molecules shows similar features.

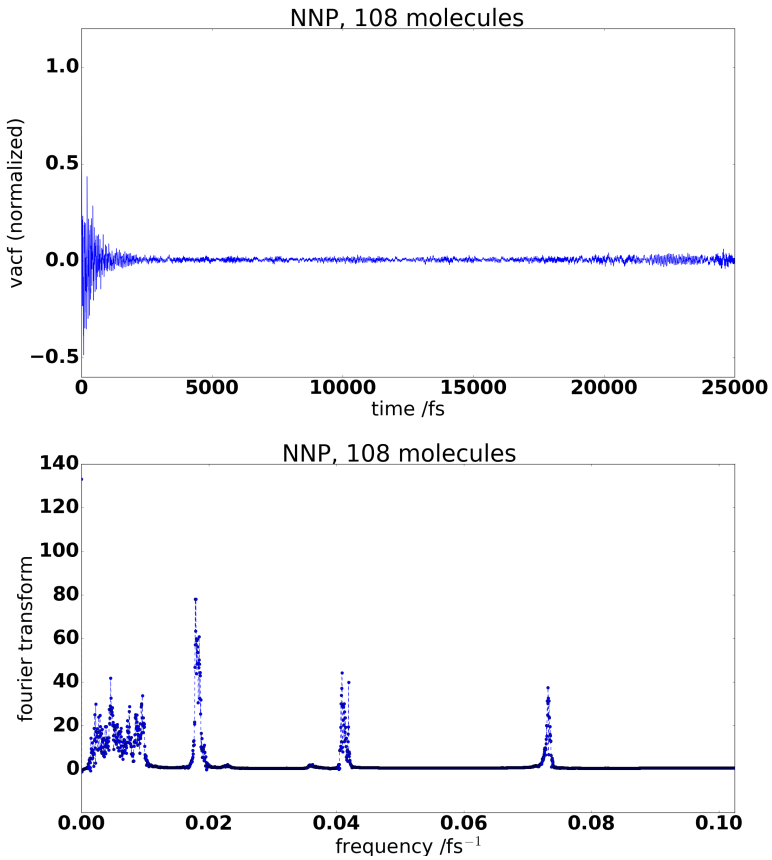

Figure 6.7: VACF (top) and Fourier transform of VACF (bottom) for a 25 000 fs NNP-MD simulation. The three characteristic peaks that can also be seen in the Fourier transform of the DFT-VACF (see figure 6.6, middle) are present. Due to the longer simulation length there are more peaks for low frequencies visible in the Fourier spectrum.

The velocity auto correlation function (VACF) contains information about how the atoms in a simulation move with respect to each other over time. The VACF can for example be used to extract information about phonon frequencies in different crystals.
After already having compared static properties of the different MD simulations using the RDF the VACF is now used to compare dynamic properties. In figure 6.5 the VACFs of three different simulations are displayed.
On the left side of figure 6.5 VACFs are displayed over a time frame of 1750 steps which is the whole duration of the DFT simulation. On the right a zoomed in version of the same VACFs is shown up to 250 steps. The top row of plots shows the VACF for a NNP simulation with 16 molecules, the second row shows the VACF for the DFT simulation, also with 16 molecules and the last row shows the VACF for the larger NNP simulation with 108 molecules.
It can be seen that especially the NNP simulation with 108 molecules shows the same structure as the DFT-VACF over the first 250 steps, although some of the peaks are less pronounced. The NNP simulation with only 16 molecules already differs more from the DFT in the beginning. When looking at the VACF over the whole 1750 steps it can be seen that the NNP-VACFs seem to increasingly differ from the DFT-VACF especially when considering the slow overarching oscillation the DFT-VACF displays.
In order to further decipher the structure of the VACFs of the three simulations their Fourier transforms were calculated and are displayed in figure 6.6 for the 1750 step VACFs in the same order as before. The Fourier transform of the DFT-VACF displays three distinct peaks

at higher frequencies and four peaks close to each other at low frequencies. Here it can be seen that, despite differing in magnitude, both NNP simulations display the same characteristic peaks at higher frequencies as the DFT simulation. However the 108 molecule NNP simulation as was already visible in figure 6.5 matches the structure of the Fourier spectrum for high frequencies much more closely.
For the low frequency peaks both NNP VACFs can not clearly match the four peaks of the DFT-VACF. Although the range of the low frequency peaks is matched by both NNP-VACFs the one with 16 molecules displays at least one additional peak. The one with 108 molecules displays three pronounced peaks similar to the DFT Fourier transform but also fails to match the structure completely.
However despite those discrepancies the NNP-VACFs, especially the larger NNP-VACF, do reproduce the key features of the DFT-VACF. A reason for the larger simulation being a better match could be that the training set always included at least 24 and up to 54 molecules. The NNP therefore seems to be better at predicting energies and forces of systems larger than the systems in the training set and loses precision when confronted with a smaller system size. The dependence of the NNP predictions on the system sizes in the training set could be explored in further studies of the behavior of NNPs.
The VACF of one more NNP simulation was studied with 108 molecules and a simulation length of 25 ps. This simulation length with time steps of 1 fs could not be reproduced with DFT-MD due to the computational cost of it. The VACF of this simulation is displayed in figure 6.7. Again the three characteristic peaks of the DFT-VACF are visible, but as could be expected there are more low frequency contributions. This was expected because due to the longer time over which the VACF was calculated more oscillations with long wavelength can expressed by the VACF.

7 Concluding Remarks

The goal of this thesis was to train a neural network potential to reproduce the potential energy surface of a system of CO_2 molecules with ab-initio precision. The NNP that was trained for this thesis could successfully be used for stable MD simulations of solid CO_2. However there were slight qualitative differences in the investigated properties of the NNP when compared to MD simulations using the reference DFT method. The small deviations in the tail end of the radial distribution function can be attributed to the small system size that had to be used for the DFT-MD simulations due to the computational cost of such simulations. This led to fewer atoms being in the cell and therefore the averages used for the RDF were not as smooth. The NNP for CO_2 was shown to be able to do MD simulations of a nearly 10 times larger system in only 2 hours on a laptop with four cores instead of 72 hours on the Vienna Scientific Cluster (VSC3) with 16 cores. The second property investigated, the velocity autocorrelation function, also showed small differences between NNP- and DFT-MD simulations. Here all the significant peaks in the Fourier spectrum of the VACF could be reproduced but the amplitudes differed by up to a factor of 4. Interestingly the NNP also showed deviations between different system sizes. This could be linked to the system sizes of the training configurations that were used (24-54 molecules) compared to the MD system sizes (16 and 108 molecules).

The correlation between training set system sizes and simulation system sizes for the NNP could be part of a future study of the properties of NNPs. The performance of the NNP created for this thesis could most likely still be improved, however this would have been beyond the limited scope of a master thesis. Possible improvements include a further study of the Kalman filter parameters used, which unfortunately can be very time consuming but better parameters could likely be found. Another improvement would be the use of a larger training set with not only more configurations but also larger system sizes.

Despite all these possible improvements of the NNP it was possible to see the biggest advantage the NNP with ab-initio precision has over pure ab-initio methods: the computational cost. A simulation that took 72 hours for 16 molecules with DFT-MD could be achieved for 108 molecules in 2 hours with the NNP. It should also be noted that the NNP does scale like an empirical potential which can be linearly with the number of atoms in the system, while DFT-methods scale non linearly with the number of electrons in a system. This could eventually lead to the possibility of testing ab-initio methods for larger systems than would ever be possible using DFT-MD alone with the help of NNPs. This approach could show potential flaws in the way ab-initio methods are used today.

Bibliography

[1] T. Morawietz, A. Singraber, C. Dellago, and J. Behler. How van der Waals interactions determine the unique properties of water. *PNAS*, 113(30):8368–8373, 2016. URL https://doi.org/10.1073/pnas.1602375113.

[2] A. Singraber, T. Morawietz, J. Behler, and C. Dellago. Parallel Multistream Training of High-Dimensional Neural Network Potentials. *J. Chem. Theory Comput.*, 15(5):3075–3092, 2019. URL https://doi.org/10.1021/acs.jctc.8b01092.

[3] Sandia National Laboratories. LAMMPS Molecular Dynamics Simulator, 2019. URL https://lammps.sandia.gov/index.html.

[4] J. G. Harris and K. H. Yung. Carbon Dioxide's Liquid-Vapor Coexistence Curve and Critical Properties As Predicted by a Simple Molecular Model. *J. Phys. Chem.*, 99(31):12021–12024, 1995. URL https://doi.org/10.1021/j100031a034.

[5] T. T. Trinh, T. J. H. Vlugt, and S. Kjelstrup. Thermal conductivity of carbon dioxide from non-equilibrium molecular dynamics: A systematic study of several common force fields. *J. Chem. Phys.*, 141(13):134504, 2014. URL https://doi.org/10.1063/1.4896965.

[6] G. Kresse and J. Hafner. Ab initio molecular dynamics for liquid metals. *Phys. Rev. B*, 47(1):558–561, 1993. URL https://doi.org/10.1103/PhysRevB.47.558.

[7] G. Kresse and J. Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Comput. Mat. Sci.*, 6(1):15–50, 1996. URL https://doi.org/10.1016/0927-0256(96)00008-0.

[8] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54(16):11169–11186, 1996. URL https://doi.org/10.1103/PhysRevB.54.11169.

[9] G. Kresse and J. Hafner. Norm-conserving and ultrasoft pseudopotentials for first-row and transition elements. *J. Phys. Condens. Matter*, 6(40):8245–8257, 1994. URL https://doi.org/10.1088/0953-8984/6/40/015.

[10] G. Kresse and D. Joubert. From ultrasoft pseudopotentials to the projector augmented-wave method. *Phys. Rev. B*, 59(3):1758–1775, 1999. URL https://doi.org/10.1103/PhysRevB.59.1758.

[11] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.*, 77:3865–3868, 1996. URL https://doi.org/10.1103/PhysRevLett.77.3865.

[12] M.D. Max. *Natural Gas Hydrate in Oceanic and Permafrost Environments*. Springer Netherlands, 2003. URL https://doi.org/10.1007/978-94-011-4387-5.

[13] H. Lee, Y. Seo, Y. Seo, I. L. Moudrakovski, and J. A. Ripmeester. Recovering Methane from Solid Methane Hydrate with Carbon Dioxide. *Angew. Chem.*, 115(41):5202–5205, 2003. URL https://doi.org/10.1002/ange.200351489.

[14] D. Bai, X. Zhang, G. Chen, and W. Wang. Replacement mechanism of methane hydrate with carbon dioxide from microsecond molecular dynamics simulations. *Energy Environ. Sci.*, 5(5):7033–7041, 2012. URL http://dx.doi.org/10.1039/C2EE21189K.

[15] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000. URL https://doi.org/10.1017/CBO9780511801389.

[16] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL https://mitpress.mit.edu/books/gaussian-processes-machine-learning.

[17] B. G. Sumpter, C. Getino, and D. W. Noid. Theory and Applications of Neural Computing in Chemical Science. *Annu. Rev. Phys. Chem.*, 45(1):439–481, 1994. URL https://doi.org/10.1146/annurev.pc.45.100194.002255.

[18] J. Gasteiger and J. Zupan. Neural Networks in Chemistry . *Angew. Chem.*, 32(4):503–527, 1993. URL https://doi.org/.

[19] S. Lawrence, C.L. Giles, A. C. Tsoi, and A.D. Back. Face recognition: a convolutional neural-network approach. *IEEE Trans. Neural Netw. Learn. Syst.*, 8(1):98–113, 1997. URL https://doi.org/10.1109/72.554195.

[20] W.Sha and K.L.Edwards. The use of artificial neural networks in materials science based research. *Mater. Des.*, 28(6):1747–1752, 2007. URL https://doi.org/10.1016/j.matdes.2007.02.009.

[21] J. Behler and M. Parrinello. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Let.*, 98(14):146401, 2007. URL https://doi.org/10.1103/PhysRevLett.98.146401.

[22] S. Bachu. CO2 storage in geological media: Role and means, status and barriers to deployment. *Prog. Energy Combust. Sci.*, 34(2):254–273, 2008. URL https://doi.org/10.1016/j.pecs.2007.10.001.

[23] M. Santoro, F. A. Gorelli, R. Bini, J. Haines, O. Cambon, C. Levelut, J. A. Montoya, and S. Scandolo. Partially collapsed cristobalite structure in the non molecular phase V in CO2. *Proc. Natl. Acad. Sci. U.S.A.*, 109(14):5176–5179, 2012. URL https://doi.org/10.1073/pnas.1118791109.

[24] V. Iota, C. Yoo, J. Klepeis, Z. Jenei, W. Evans, and H. Cynn. Six-fold coordinated carbon dioxide VI. *Nat. Mater.*, 6:34–38, 2007. URL https://doi.org/10.1038/nmat1800.

[25] M. Santoro, F. A. Gorelli, R. Bini, G. Ruocco, S. Scandolo, and W. A. Crichton. Amorphous silica-like carbon dioxide. *Nature*, 441:857–860, 2006. URL https://doi.org/10.1038/nature04879.

[26] F. Datchi, V.M. Giordano, P. Munsch, and A.M. Saitta. Structure of carbon dioxide phase IV: breakdown of the intermediate bonding state scenario. *Phys. Rev. Let.*, 103 (18):185701, 2009. URL https://doi.org/10.1103/PhysRevLett.103.185701.

[27] J. Li, O. Sode, G. A. Voth, and S. Hirata. A solid–solid phase transition in carbon dioxide at high pressures and intermediate temperatures. *Nat. Commun.*, 4(2647), 2013. URL https://doi.org/10.1038/ncomms3647.

[28] B. Olinger. The compression of solid CO2 at 296 K to 10 GPa. *J. Chem. Phys.*, 77(12): 6255–6258, 1982. URL https://doi.org/10.1063/1.443828.

[29] K. Aoki, H. Yamawaki, M. Sakashita, Y. Gotoh, and K. Takemura. Crystal Structure of the High-Pressure Phase of Solid CO2. *Science*, 263(5145):356–358, 1994. URL https://doi.org/10.1126/science.263.5145.356.

[30] L. Liu. Dry ice II, a new polymorph of CO2. *Nature*, 303:508–509, 1983. URL https://doi.org/10.1038/303508a0.

[31] C. S. Yoo, H. Kohlmann, H. Cynn, M. F. Nicol, V. Iota, and T. LeBihan. Crystal structure of pseudo-six-fold carbon dioxide phase II at high pressures and temperatures. *Phys. Rev. B*, 65(10):104103, 2002. URL https://doi.org/10.1103/PhysRevB.65.104103.

[32] F. Datchi, B. Mallick, A. Salamat, G. Rousse, S. Ninet, G. Garbarino, P. Bouvier, and M. Mezouar. Structure and compressibility of the high-pressure molecular phase II of carbon dioxide. *Phys. Rev. B*, 89(14):144101, 2014. URL https://doi.org/10.1103/PhysRevB.89.144101.

[33] R. C. Hanson. A New High-Pressure Phase of solid CO2. *J. Phys. Chem.*, 89(21): 4499–4501, 1985. URL https://doi.org/10.1021/j100267a019.

[34] C. Yoo, V. Iota, and H. Cynn. Nonlinear Carbon Dioxide at High Pressures and Temperatures. *Phys. Rev. Let.*, 86(3):444–447, 2001. URL https://doi.org/10.1103/PhysRevLett.86.444.

[35] H. Olijnyk and A. P. Jephcoat. Vibrational studies on CO2 up to 40 GPa by Raman spectroscopy at room temperature. *Phys. Rev. B*, 57(2):879–888, 1998. URL https://doi.org/10.1103/PhysRevB.57.879.

[36] V. M. Giordano and F. Datchi. Molecular carbon dioxide at high pressure and high temperature. *EPL*, 77(4):46002, 2007. URL https://doi.org/10.1209/0295-5075/77/46002.

[37] W. Sontising, Y. N. Heit, J. L. McKinley, and G. J. O. Beran. Theoretical predictions suggest carbon dioxide phases III and VII are identical. *Chem. Sci.*, 8(11):7374–7382, 2017. URL https://doi.org/10.1039/c7sc03267f.

[38] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, 2001. URL https://doi.org/10.1016/B978-0-12-267351-1.X5000-7.

[39] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987. URL https://doi.org/10.1093/oso/9780198803195.001.0001.

[40] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2004. URL https://doi.org/10.1017/CBO9780511816581.

[41] M. E. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. Oxford University Press, 2010. URL https://doi.org/10.1002/anie.201105752.

[42] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.*, 76(1):637–649, 1998. URL https://doi.org/10.1063/1.442716.

[43] S. Nose. A unified formulation of the constant temperature molecular dynamics methods. *J. Chem. Phys*, 81(1):511–519, 1984. URL https://doi.org/10.1063/1.447334.

[44] W. G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31(3):1695–1697, 1985. URL https://doi.org/10.1103/PhysRevA.31.1695.

[45] M. Parrinello and A. Rahman. Polymorphic transitions in single crystals: A new molecular dynamics method. *J. App. Phys.*, 52(12):7182–7190, 1981. URL https://doi.org/10.1063/1.328693.

[46] W. Shinoda, M. Shiga, and M. Mikami. Rapid estimation of elastic constants by molecular dynamics simulation under constant stress. *Phys. Rev. B*, 69(13):134103, 2004. URL https://doi.org/10.1103/PhysRevB.69.134103.

[47] A. Singraber, J. Behler, and C. Dellago. Library-Based LAMMPS Implementation of High-Dimensional Neural Network Potentials. *J. Chem. Theory Comput.*, 15(3):1827–1840, 2019. URL https://doi.org/10.1021/acs.jctc.8b00770.

[48] J. E. Jones and C. Sydney. On the determination of molecular fields. —II. From the equation of state of a gas. *Proc. R. Soc. Lond. A*, 106(736):463–477, 1924. URL https://doi.org/10.1098/rspa.1924.0082.

[49] R. Gross and A. Marx. *Festkoerperphysik*. Oldenburg Wissenschaftsverlag GmbH, 2012. URL https://www.degruyter.com/view/product/278639.

[50] V. P. Filippova, S. A. Kunavin, and M. S. Pugachev. Calculation of the Parameters of the Lennard Jones Potential for Pairs of Identical Atoms Based on the Properties of Solid Substances. *Inorg. Mater. Appl. Res*, 6(1):1–4, 2015. URL https://doi.org/10.1134/S2075113315010062.

[51] D. Yin and A. D. Mackerell Jr. Combined Ab Initio / Empirical Approach for Optimization of Lennard-Jones Parameters. *J. Comput. Chem.*, 19(3):334–348, 1998. URL https://doi.org/10.1002/(SICI)1096-987X(199802)19:3<334::AID-JCC7>3.0.CO;2-U.

[52] R. W. Hockney and J. W. Eastwood. *Computer Simulations Using Particles*. IOP Publishing Ltd., 1988. URL https://doi.org/10.1002/phbl.19880441113.

[53] H. A. Lorentz. Über die Anwendung des Satzes vom Virial in der kinetischen Theorie der Gase. *Ann. Phys. (Berl.)*, 248(1):127–136, 1881. URL https://doi.org/10.1002/andp.18812480110.

[54] D. Berthelot. Sur le mélange des gaz. *Compt. Rendus*, 126:1703, 1898.

[55] J. Ryckaert, G. Ciccotti, and H. J. C. Berendsen. Numerical integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of n-Alkanes. *J. Comput. Phys.*, 23(3):327–341, 1977. URL https://doi.org/10.1016/0021-9991(77)90098-5.

[56] J. J. Potoff and J. I. Siepmann. Vapor-Liquid Equilibria of Mixtures Containing Alkanes, Carbon Dioxide, and Nitrogen. *AIChE J*, 47(7):1676–1682, 2001. URL https://doi.org/10.1002/aic.690470719.

[57] C. Nieto-Draghi, T. de Bruin, J. Pérez-Pellitero, J. B. Avalos, and A. D. Mackie. Thermodynamic and transport properties of carbon dioxide from molecular simulation. *J. Chem. Phys.*, 126(6):064509, 2007. URL https://doi.org/10.1063/1.2434960.

[58] T. T. Trinh, T. J. H. Vlugt, and S. Kjelstrup. Thermal conductivity of carbon dioxide from non-equilibrium molecular dynamics: A systematic study of several common force fields. *J. Chem. Phys.*, 141(13):134504, 2014. URL https://doi.org/10.1063/1.4896965.

[59] J. C. Cuevas. Introduction to Density Functional Theory, accessed 28th May 2019. URL http://webs.ftmc.uam.es/juancarlos.cuevas/Talks/JC-Cuevas-DFT.pdf.

[60] J. Behler. Perspective: Machine learning potentials for atomistic simulations. *J. Chem. Phys.*, 145(17):170901, 2016. URL https://doi.org/10.1063/1.4966192.

[61] A. Singraber. n2p2 -A neural network potential package -documentation, 2019. URL https://compphysvienna.github.io/n2p2/.

[62] J. Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.*, 134(7):074106, 2011. URL https://doi.org/10.1063/1.3553717.

[63] J. Behler. Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations. *Phys. Chem. Chem. Phys.*, 13(40):17930–17955, 2011. URL https://doi.org/10.1039/C1CP21668F.

[64] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signals Syst.*, 2(4):303–314, 1989. URL https://doi.org/10.1007/BF02551274.

[65] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, 1989. URL https://doi.org/10.1016/0893-6080(89)90020-8.

[66] J. Behler. Perspective: Machine learning potentials for atomistic simulations. *J. Chem. Phys.*, 145(17):170901, 2016. URL https://doi.org/10.1063/1.4966192.

[67] S. Haykin. *Kalman Filtering and Neural Networks*. JOHN WILEY & SONS, INC., 2001. URL https://doi.org/10.1002/0471221546.