MASTERARBEIT | MASTER'S THESIS

Titel | Title

Nuclear quantum effects at the water/vapor interface from neural network based path integral molecular dynamics simulations

verfasst von | submitted by

Elias Eingang BSc

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

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 876
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Physics
Betreut von | Supervisor:	Univ.-Prof. Mag. Dr. Christoph Dellago

Acknowledgements

First and foremost, I would like to express my special thanks of gratitude to Dr. Marcello Sega for his constant aid and his patience, as well as his technical support, without which this work would not have been possible.

I would also like to thank my family for their kindness and all kinds of support. They lifted me up when I was feeling down and grounded me when my head was in the clouds.

I would also like to extend my gratitude to Andreas Singraber and Oliver Wohlfahrt, who paved the way with their work and took time out of their busy schedules to give me advice when I needed it.

Last but certainly not least, I would like to thank Univ.-Prof. Dr. Christoph Dellago, for giving me the opportunity to do research in this very exciting field and for supervising this work.

Zusammenfassung

Aufgrund seiner Rolle in vielen natürlichen Prozessen als auch seiner anomalen Eigenschaften steht Wasser im Mittelpunkt zahlreicher Forschungen. Der Atomkern von Wasserstoff besteht aus einem einzelnen Proton, dessen geringe Masse dazu führt, dass Wasser bedeutend von Kernquanteneffekten betroffen ist. Diese Effekte können in Ab-initio Simulationen mit Hilfe des quantenmechanischen Pfadintegralformalismus berücksichtigt werden. Bei der Pfadintegral-Molekulardynamik wird jedes Teilchen als ringartiges Polymer dargestellt, was allerdings den ohnehin schon hohen Rechenaufwand weiter steigert. Um dem entgegenzuwirken, können Methoden neuronaler Netzwerke hinzugezogen werden, was die Rechengeschwindigkeit erhöht und bei vergleichbarer Genauigkeit relevante Größen- und Zeitskalen zugänglich macht. In dieser Arbeit wird ein neuronales Netzwerk-Potential anhand von Energie- und Kraftberechnungen trainiert, die von Dichtefunktionaltheorieberechnungen der Potentialhyperfläche von Wasser in der Revised Perdew–Burke–Ernzerhof Näherung mit Dispersionskorrekturen stammen. Der Einfluss von Kernquanteneffekten auf die Grenzfläche zwischen Wasser und Wasserdampf wird untersucht, indem das neuronale Netzwerkpotential sowohl mit als auch ohne Hinzunahme von Pfadintegral-Molekulardynamik ein System von Wasser simuliert. Strukturelle und thermodynamische Eigenschaften werden entlang der Wasser/Dampf-Koexistenzlinie berechnet um Einsicht in Leistung und Verhalten der Kombination von neuronalen Netzwerk- und Pfadintegralmethoden zu erlangen. Die Ergebnisse zeigen unter Anderem die erwartete Verschiebung des kritischen Punktes bei Inklusion der Kernquanteneffekte, als auch Veränderungen in der bevorzugten Struktur der Wasseroberfläche. Die präsentierten Rechenergebnisse wurden zum Teil am Vienna Scientific Cluster (VSC) erzielt.

Abstract

Due to its role in many natural processes as well as its anomalous properties, water remains the center of focus for a number of researches. The nuclei of hydrogen atoms are formed by a single proton, whose low mass leads to water being prominently affected by nuclear quantum effects. To include these effects in first principles simulations, a computational method based on the quantum mechanical Feynman path integral formulation is used. Known as path integral molecular dynamics, each particle is treated as a ring polymer to approximate their quantum nature, which further increases the already high computational cost. However, through the use of neural network methods, the speed of calculations can be improved to make the size and time scale needed for meaningful evaluations feasible while retaining high accuracy. In this work, a neural network potential is trained based on reference energies and forces of the density functional theory potential energy surface of water in the Revised Perdew–Burke–Ernzerhof approximation corrected for dispersion. To gauge the impact of nuclear quantum effects on the water/vapor interface, that neural network potential is used both with and without employing path integral molecular dynamics to simulate an interfacial water system. Structural and thermodynamic properties are calculated along the liquid/vapor coexistence line to garner insight of the performance of the combination of path integral molecular dynamics and neural network potentials. The results show the expected shift of the critical point upon inclusion of the nuclear quantum effects, as well as changes in the preferential structure at the water's surface. The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC).

Contents

1. Introduction

While machine learning applications such as ChatGPT [1] or latent diffusion model-based image generation tools [2] have only recently received wider recognition in the public eye, the usage of artificial neural networks (ANN) in natural sciences has been explored for decades, from event filtering in high energy physics [3] to the solving of systems of ordinary differential equations [4]. Their use can be motivated easily: When knowing the input of a process - for example the mass and acceleration of a body - as well as the corresponding output - for example the force acting on the body - it is of interest to find a function that links these quantities together. In this simple case, we are looking for Newton's second law of motion, which allows to predict the output force for any given input mass and acceleration. For more complex processes however, it may be unfeasible to use the exact function, as the computational cost of predicting the output to a given input is too high. In other cases, the exact function may not be known at all so a prediction is impossible. A more productive approach can be to find an approximation function by tuning the parameters of a model until the known inputs are convincingly mapped to the corresponding outputs. The exact form of the approximation function is of little interest as long as it produces the correct predictions. ANNs can be used to achieve exactly that: Predicting an output to a given input, ideally with faster evaluation times and without sacrificing too much accuracy, by training the ANN beforehand, using either experimental data or data of other models that the ANN should emulate.

1.1. Artificial neural networks in computational physics

One field where the approximation of functions, which would be otherwise costly in terms of computing time, is desirable, is the field of computational physics. When considering a potential energy surface (PES) [5], which describes the relationship between the energy of a system of atoms and their positions, one could solve the Schrödinger equation for each atom to calculate a point on the PES. When performing molecular dynamics (MD) simulations [6], where the calculation of many points on the PES is needed, this approach is usually unfeasible. Approximate ab initio methods where forces are obtained "on the fly" from electronic structure calculations, for instance based on density functional theory (DFT) [7], alleviate these difficulties, but are still practically limited concerning system size and simulation time. At this point, ANNs may be utilized as an approximation function to speed up DFT-based energy and force evaluations while still maintaining a high degree of accuracy. For this, the PES data obtained from DFT calculations is used to train an ANN as a neural network potential (NNP) which in turn can be used in MD simulations to carry out calculations. A popular DFT model is the PBE functional by

Perdew, Burke and Ernzerhof [8], which was further developed into the revised RPBE density functional [9]. As van der Waals forces could still not be described with RPBE, correction schemes were devised to create the RPBE-D3 functional [10]. This was later shown by Morawietz [11], also with the help of NNPs, to be an important step as van der Waals corrections are crucial to accurately model water.

A successful approach to implement NNPs for use in MD was made by Behler and Parrinello in 2007 [12], with applications to many materials having reached accuracy on par with DFT simulations since then [13]. Behler describes the advent of this type of NNPs in 4 generations [14]: The first generation, emerging in the 1990s, was restricted to the global description of low-dimensional systems, while the second generation, employed in this work, is applicable to high-dimensional systems by only describing the short-ranged local atomic energies. The third generation introduced local atomic charges to include long-range electrostatics and dispersion interactions, and the current fourth generation aims to also treat non-local interactions, i.e. global charge (re-)distributions, which are not captured with the local atomic charges approach. Other advancements in the field include the kindred approach of deep potentials, which may incorporate long-range interaction models and for which the powerful DeePMD-kit software package was recently updated [15], and message passing neural networks, which pass information from atom to atom to extend the interaction range and for which the efficient MACE architecture was recently developed [16].

In this work, the NNP approach by Behler and Parrinello is applied to investigate the water/vapor interface of a system of water in slab geometry, expanding on a similar work recently carried out by Oliver Wohlfahrt [17]. For this, the n2p2-package written by Andreas Singraber [18] is used to create an NNP based on RPBE-D3 calculations. Originally, it was planned to employ here the very NNP used by Wohlfahrt in his work, but due to it being created on a very early version of n2p2, it was not possible to make it run on the current version of the software. Instead, Wohlfahrt's data was used for comparison with the results generated in this work.

1.2. Nuclear quantum effects and path integrals

Nuclei of light elements can exhibit significant quantum mechanical behaviour such as delocalization, zero point energy and quantum tunneling. Such effects are known as nuclear quantum effects (NQE). They prominently occur in hydrogen atoms, since their nuclei consist only of a single proton, and consequently play an important role in determining the properties of water. It is thus of great interest to understand NQEs and to that end devise methods to incorporate them in simulations of water or other aqueous solutions. A comprehensive review by Ceriotti from 2016 details recent methods, developments and challenges [19].

Perhaps the most prominent approach to include NQEs is based on the Feynman path-integral formulation of quantum mechanics [20], where the formalism is used to relate equilibrium quantum many body theory to classical statistical mechanics in the form of an isomorphism [21]. This leads to the mathematical description of each atom as a cyclic polymer chain of P points or "beads", where each bead is connected to the adjacent beads via a harmonic potential [22]. Applying this isomorphism to MD gives rise to path integral molecular dynamics (PIMD), with a variety of methods already implemented [23, 24] which are able to sample equilibrium and dynamical properties [25, 26]. Among other findings, PIMD was successfully used to investigate how NQEs affect bond orientations at the water/vapor interface [27], the surface of mixtures of light and heavy water [28] and the influence of NQEs on the vibrational dynamics of surface water molecules [29]. It must be noted that due to the additional interactions that must be evaluated, the computational demands to include NQEs by performing PIMD increase. This may be alleviated by employing the neural network methods described above, which was already done to explore the thermodynamics and phase diagram of water [30, 31].

Normal mode path integral molecular dynamics (NMPIMD) is a type of PIMD which utilizes a normal mode representation that allows for shorter computation times compared to primitive PIMD [23]. In this work, NMPIMD is employed to simulate the aforementioned water system, using the i-PI program developed by Ceriotti et al. [32] in conjunction with force calculations carried out by the LAMMPS Molecular Dynamics Simulator [33], which in turn is aided by an NNP. The goal is to obtain values for the surface tension and critical point, extract mass density and intrinsic profiles of the system, and investigate the microscopic structure of the system, especially near the water/vapor interface. These results are then compared to a simulation additionally performed without PIMD and thus disregarding NQEs.

1.3. Thesis Structure

Chapters 2 and 3 present the theoretical foundations of NNPs and PIMD, respectively. In Chapter 4, the software used in this work is introduced and some of their methods explained. Chapter 5 deals with the training of the NNP later used for simulations as well as a brief treatment of DFT and the density functional the training data is based on. In Chapter 6, the simulations are detailed, from the initial equilibration phase using the empiric SPC/F model over further equilibration using the previously trained NNP up to the productions runs using PIMD methods as well as non-PIMD simulations carried out for comparison. Chapter 7 explains the observables gathered from the simulational data and how to calculate them, while Chapter 8 presents the results of the simulations in regards to these aforementioned observables. Lastly, Chapter 9 provides a discussion on the results and methods used.

2. Artificial Neural Networks

As their name suggests, the underlying idea of ANNs borrows from biology, where the nervous system transmits electrochemical signals through the use of a network of neurons. Neurons receive signals from one or more other neurons and in turn relay signals to again one or more different neurons, creating in a most conceptualizing view a system that turns input data into output data. The notion of an artificial neuron was first described by McCulloch and Pitts in 1943 [34], introducing the idea of neurons that receive mathematical input from other neurons and forwarding the resulting output to different neurons. The further development of the concept of artificial neurons was also aided by contributions from the field of psychology, namely the considerations of Hebb in 1949 of how synapses learn and how the connection between neurons are weighted [35]. This eventually led to the development of perceptrons by Rosenblatt, first published in 1957 [36] and further detailed in 1962 [37].

In this chapter, Section 2.1 explains the basics of perceptrons and their mathematical description, while Section 2.2 deals with the application of this concept to computational physics problems in the form of NNPs in the approach by Behler and Parrinello [12]. Lastly, Section 2.3 takes a look at how ANNs can be trained and introduces the Kalman filter [38] as a training algorithm.

2.1. Perceptrons

Rosenblatt originally described a perceptron as a system consisting of 3 subsystems: The sensory system, containing S-units that receive sensory input; the association system, containing A-units that evoke an output based on the algebraic sum of input pulses they receive from connected S-units and whose output may vary with the system's history; and the response system, containing a small number of R-units that are activated when the signals received from connected A-units exceeds a critical level and which will then provide an output signal.

In a more schematic view, let there be n input nodes, so that each node i provides an input x_i. In addition, let there be a "0th" input node of constant value b called bias node. These nodes shall be collectively called the input layer. Then, an output node is defined by the $n + 1$ connections coming in from the input nodes, the set of weights w_i defining the strength of the connection to each input node i, and the activation function f. Let this output node be called the output layer. The value of the output node is then based on the product of the transferred input x_i of node i with the respective weight value w_i

of each incoming connection, yielding

$$f\left(b \cdot w_0 + \sum_{i=1}^{n} x_i \cdot w_i\right) . \tag{2.1}$$

In Rosenblatt's description, the summation and evaluation of inputs presented as the argument of the function in Eq. 2.1 corresponds to the association system, while the evaluation of the function itself and the output of its result corresponds to the response system. This type of algorithm is known as single-layer perceptron and can be seen as a single neuron fed by a group of preceding neurons. While at first regarded as promising, the single-layer perceptron fails to deal with non-linearly seperable patterns, famously shown by Minsky and Papert in 1969 when they proved that this class of ANN is incapable of learning an XOR function [39].

An advancement of the perceptron concept, the multilayer perceptron (MLP[1]), was proven to be able to solve problems that a single-layer perceptron cannot [40]. MLPs are comprised of a minimum of three layers: One input and one output layer as described above, as well as one or more hidden layers in between them. Each node from each hidden layer receives signals from incoming connections from the previous layer and each incoming connection possesses its own adjustable weight value. The MLP belongs to the class of feedforward neural networks (FFNNs), which are defined by the feature that information only travels in one direction, from each layer to the subsequent one, as opposed to recurrent neural networks which allow connections to previous layers or to nodes in the same layer [41]. If the topology of an FFNN is so that every node of every layer has outgoing connections to every node of the subsequent layer, the FFNN is called fully connected. Such a fully connected MLP is depicted in Figure 2.1. Alternative topologies may also include short cuts, where the outgoing connection of a node skips one or more layers and thus links to a node of a later layer than the subsequent one.

2.1.1. Mathematical description of multilayer perceptrons

To start describing an MLP, some notations need to be introduced. The number of hidden layers L is called the depth of the network. The input layer is treated as the 0th layer and the output layer is defined as the $(L+1)$th layer. The number of nodes of the lth layer, ignoring the aforementioned bias nodes, is called the layer's width and is denoted as $m^{(l)}$. The bias nodes are treated as the 0th node of each layer, except the output layer, which does not have a bias node. The weight w_{ij}^{l-1} is associated with the connection coming from the ith node of layer $l-1$ and pointing to the jth node of layer l. y_k^l denotes the output value of the kth node of the lth hidden layer, with the bias node being $b^l = y_0^l$. x_k is the value of the kth node of the input layer and is synonymous with y_k^0. The graph in Figure 2.1 is an example using these notations. Now one can calculate the value of

[1]Not to be confused with machine learning potentials, which may also be abbreviated as MLP.

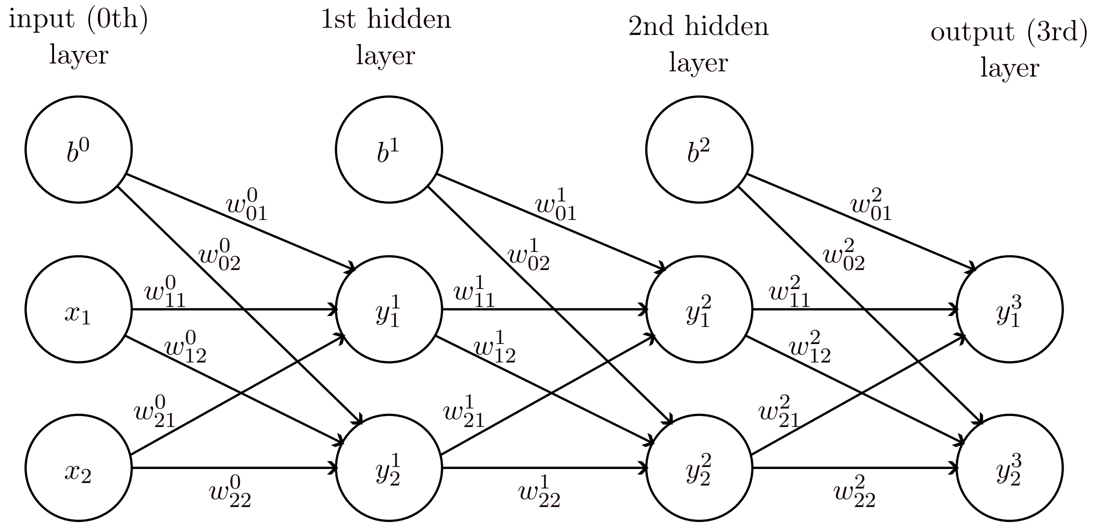


Figure 2.1.: Example of a fully connected multilayer perceptron with a depth of $L = 2$ (2 hidden layers) and a width of $m^{(l)} = 2$ for every layer (2 nodes per layer).

each node, except the input nodes, similarly to Equation 2.1:

$$y_k^l = f_k^l \left(b^{l-1} w_{0k}^{l-1} + \sum_{i=1}^{m^{l-1}} y_i^{l-1} w_{ik}^{l-1} \right) \ . \tag{2.2}$$

Here, f_k^l is the activation function of the kth node of the lth layer. The argument of the activation function is called the activation of the corresponding neuron and is composed of the sum of all weighted inputs. As the bias nodes are of constant value, their contribution to the activation can be absorbed in their corresponding weights if their value is set to $b^l = 1$. Then, the activation u_k^l of the kth neuron of the lth layer can be written as

$$u_k^l = w_{0k}^{l-1} + \sum_{i=1}^{m^{l-1}} y_i^{l-1} w_{ik}^{l-1} \tag{2.3}$$

and the node value becomes $y_k^l = f_k^l(u_k^l)$.

Formally, an MLP can be described analytically as a nested function $\Phi : \mathbb{R}^{m^{(0)}} \to \mathbb{R}^{m^{(L+1)}}$. Let $\mathbf{y}^l$ be the vector of length $m^{(l)}$ that contains the values of the nodes of layer l and let $\mathbf{w}$ be the vector of all weight parameters of the MLP as well as $\mathbf{w}^l$ be the vector containing the weights of all connections going from layer l to layer $l+1$. To set them apart from the other layers' values, the vector of values of the input layer is also denoted as $\mathbf{y}^0 = \mathbf{x}$. If one now defines the functions $\varphi^l : \mathbb{R}^{m^{(l)}} \to \mathbb{R}^{m^{(l+1)}}$ that each map the output values of the nodes of the lth layer $\mathbf{y}^l$ onto the values of the nodes of the $(l+1)$th layer $\mathbf{y}^{l+1}$, one can write:

$$\Phi(\mathbf{x}, \mathbf{w}) = \varphi^L(\mathbf{w}^L, \varphi^{L-1}(\mathbf{w}^{L-1}, \varphi^{L-2}(\ldots, \varphi^0(\mathbf{w}^0, \mathbf{x}) \ldots))) \ . \tag{2.4}$$

Lastly, a distinction between the weight parameters on the one hand and parameters like depth, width, as well as the parameters used to optimize an MLP on the other hand, should be established. To that end, the latter shall be denoted as hyperparameters.

2.1.2. Activation functions

As among the uses of the original perceptron was the mapping of Boolean functions onto a neural network, the most obvious choice for the activation function is the Heaviside step function

$$H(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.5}$$

Considering this choice in Equation 2.1, one can set a negative bias b and interpret it as the threshold that the sum of signals must overcome to activate the neuron, which then yields a binary response.

The choice of activation functions can greatly influence the success of an ANN and some activation functions may be better suited for certain applications than others. The desire to approximate complicated functions through the use of ANNs gave rise to universal approximation theorems. These concern under which circumstances FFNNs have the capabilities to approximate any given function and thus be what is called universal approximators. The relevant result here is the following finding: For every continuous function $g : K \to \mathbb{R}^m$ with compact $K \subseteq \mathbb{R}^n$, there exists an MLP with one or more hidden layers and corresponding activation functions that are continuous, bounded and non-constant, so that $|g^{(MLP)}(x) - g(x)| < \epsilon$, where $g^{(MLP)}$ is the approximation of the function g generated by the MLP, $x \in K$ and $\epsilon > 0$ is an arbitrarily small number. This theorem thus implies that any function may be successfully approximated by an MLP, hence they are universal approximators, and was proven in 1989 by Stinchombe, White and Hornik [42], the latter of whom has also shown that the potential of being a universal approximator stems from the neural network architecture of FFNNs themselves rather than the choice of their activation functions [43]. However, this theorem does not provide a method for constructing such an MLP and its weights, but merely states that the construction is possible. Especially for MLPs with only one hidden layer, such a construction may be impractical, as the theorem holds for a single hidden layer of arbitrary width, which may be unfeasibly large. It also does not state that the activation functions of an MLP must be continuous, bounded and non-constant to successfully approximate a function, but merely that these criteria ensure a successful approximation, taking into account the above caveat. There exist other universal approximation theorems dealing with MLPs of bounded width and arbitrary depth as well as MLPs with both bounded depth and width, though these will not be discussed in this work.

Having established the desired properties of activation functions for MLPs, one can identify two common choices that meet the criteria for the universal approximation theorem: The logistic function

$$f_s(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

and the hyperbolic tangent function

$$f_t(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \,, \tag{2.7}$$

which are depicted in Figure 2.2, though many more functions are possible. To generate a meaningful output, the activation of a neuron should lead to these activation functions being evaluated somewhere in their characteristic nonlinear region. While in the case of the Heaviside step function, the bias shifted the argument to represent a threshold, in the case of these functions, the bias shifts the argument to ensure a meaningful evaluation in the nonlinear region of the function. Usually when constructing an MLP, the same type of

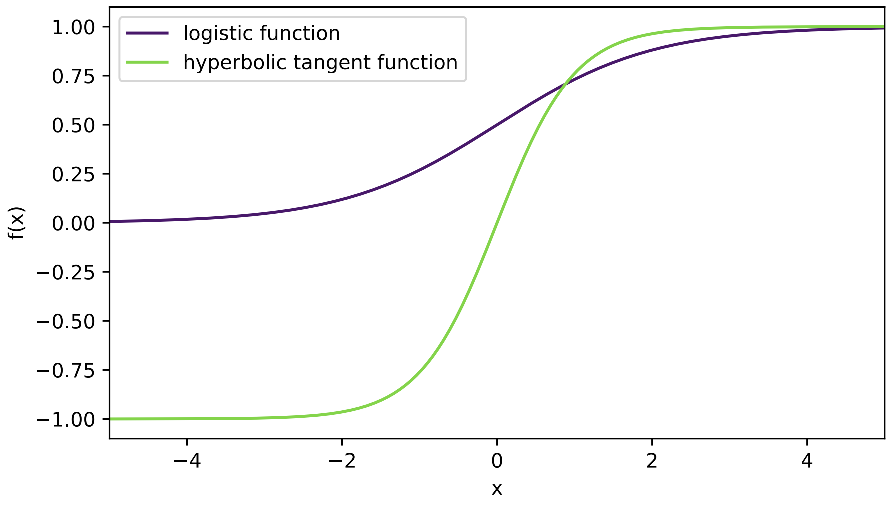


Figure 2.2.: Logistic function and hyperbolic tangent function.

activation function is used for every node of the same layer, while potentially varying the activation function from layer to layer. The output layer often uses the identity function as the activation function of all of its nodes, which does not contradict the universal approximation theorem, as it only concerns the activation functions of the hidden layers. Many other types of activation functions are possible, whether they satisfy the criteria of the universal approximation theorem or not, as there is no general right choice of activation function for any specific problem and the success of a chosen activation function may depend on the optimization algorithm used and its hyperparameters.

2.2. Neural network potentials by Behler and Parrinello

In the framework described above, neural networks may be used in computational chemistry and physics to approximate potential energies of systems of multiple atoms.

Recent achievements, efforts and works have been summarized by Handley and Popelier [44] as well as Behler [45, 13, 14]. Some challenges that occur when trying to approximate the PES through the use of an NNP that must be dealt with are a number of invariances that should hold. In the case where no external forces are acting on the atoms of a system, an NNP should be invariant with respect to translation of the whole system, as well as invariant to rotation of the system. These properties arise from the fact that a PES is also invariant in these regards. Additionally, an NNP should be invariant with respect to the order of atoms of the same element, once again as this generally also holds for PESs. Lastly, while not an invariance, an NNP's approximation capabilities should not depend on the number of atoms in the system. If for example, an NNP is structured as an MLP in a naive fashion, where the input layer contains $3N$ nodes, corresponding to the Cartesian coordinates of the atoms of a system of N atoms, then this NNP can only be applied to systems with N atoms. Larger systems would have additional atom coordinates that are not accounted for in the structure of the MLP, while smaller systems have too few atom coordinates to populate all input nodes with data. Furthermore, simulations in the grand canonical ensemble, where the number of atoms changes, would be impossible for obvious reasons. This is not to say that such an NNP is unusable, but one would need to train a separate NNP for each desired system size, which makes the approach inflexible and may be deemed unsatisfactory.

This work follows the approach by Behler and Parrinello [12], which is suited for creating NNPs that approximate high-dimensional PESs, also called high-dimensional neural network potentials (HDNNP), and overcomes the aforementioned challenges using so called symmetry functions, as well as their own MLP subnet for each atom type. The main idea is to represent the total potential energy E_{pot} of a system of N atoms as the sum of the energy contributions ϵ_i of each atom i:

$$E_{pot} = \sum_{i=1}^{N} \epsilon_i \; . \tag{2.8}$$

Let $\vec{X}_i$ denote the vector of the Cartesian coordinates of atom i and let $\mathbf{X}$ be the column vector containing all $3N$ coordinates of the systems atoms[2], i.e. $\mathbf{X} = \left(\vec{X}_1^T, \ldots, \vec{X}_N^T\right)^T$. Then there is a vector $\mathbf{G}_i$ for each atom i, whose entries are a set of functions of all $3N$ atom coordinates, so that the μth entry, of the vector corresponding to the ith atom, can be written as the function

$$G_i^\mu = G_i^\mu(\mathbf{X}) \; . \tag{2.9}$$

These functions shall be known as symmetry functions (SF) and will be explained in detail in the following sections. In a first step, $\mathbf{X}$ is plugged into each symmetry function to gather the vector of symmetry function values $\mathbf{G}_i$ for each atom i. These $\mathbf{G}_i$ can be regarded as generalized coordinates that describe the local environment of atom i. For

[2]Regarding vector notation: In this work, three-dimensional vectors containing Cartesian coordinates shall be accented by an arrow, while higher-dimensional or more general vectors (or matrices) are denoted by being printed in boldface.

each atom i there is now an atomic MLP denoted as subnet S_i that yields the energy contribution ϵ_i, which when summed up yields the total potential energy E_{pot}. To ensure the invariance of E_{pot} with respect to the order of atoms, the structure of all subnets as well as their weights must be identical for each S_i. This last constraint actually may be weakened a bit when keeping in mind that the invariance must only hold for atoms of the same element. If one defines the set $\chi = \{\chi_1, \ldots, \chi_c\}$ of the c chemical elements that occur in the system, then there are also c different subnets S_χ. Each atom i, whose element shall be denoted by χ_i, then uses the corresponding subnet S_χ for the calculation of its energy contribution, and only the subnets of the same element are constrained to be identical. The topology for a simple system containing a single water molecule is shown schematically in Figure 2.3.

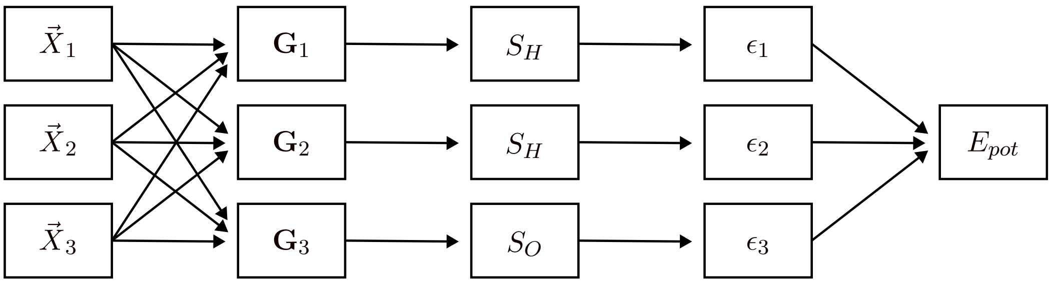


Figure 2.3.: Scheme of an NNP as described by Behler and Parrinello for a system containing a single water molecule. The system consists of two hydrogen atoms (quantities indexed with subscript 1 and 2) and one oxygen atom (quantities indexed with subscript 3). The generalized coordinates $\mathbf{G}_i$ of atom i depend on the Cartesian coordinates $\vec{X}_i$ of all atoms in the environment and enter their respective subnets. As there are two elements, $\chi = \{H, O\}$ and thus there are two subnets, S_H and S_O, i.e. the subnets of the two hydrogen atoms corresponding to the indexes 1 and 2 are identical. The atomic energy contributions ϵ_i yielded by each subnet are then summed to get the total potential energy E_{pot}. Note that arrows do not indicate weighted connections of a neural network, but rather symbolize that the object at the end of each arrow is derived from the object at the start of the arrow.

2.2.1. Symmetry functions

The purpose of SFs is to fulfill the NNP's invariance with respect to translation and rotation of the system and with respect to the permutation of atoms. To motivate the last invariance, consider a single water molecule, whose atoms' positions are described by the vectors $\vec{x}_O$, $\vec{x}_{H_1}$ and $\vec{x}_{H_2}$. Since the hydrogen atoms are identical, it should not matter if they swap positions, as the energy prescribed by the PES, symbolized here as a function E, would stay the same:

$$E_{pot} = E(\vec{x}_O, \vec{x}_{H_1}, \vec{x}_{H_2}) = E(\vec{x}_O, \vec{x}_{H_2}, \vec{x}_{H_1}) \ . \tag{2.10}$$

This is not guaranteed when approximating the PES by means of an NNP and must thus be accounted for by the NNP's structure. Relative quantities like interatomic distances are naturally invariant with respect to translation and rotation of the whole system and can furthermore be used to construct symmetry functions that are invariant with respect to permutation of atoms [46]. Let the interatomic distances be denoted as $r_{OH_1} = |\vec{x}_O - \vec{x}_{H_1}|$ and $r_{OH_2} = |\vec{x}_O - \vec{x}_{H_2}|$. Then one can construct the SFs

$$G^+ = |r_{OH_1} + r_{OH_2}| \tag{2.11}$$
$$G^- = |r_{OH_1} - r_{OH_2}| \tag{2.12}$$

that are invariant to the permutation of the positions of the hydrogen atoms. This is of course a very simplistic example, but the idea can be easily expanded to larger systems of N atoms by constructing SFs that contain the absolute value of sums or differences of other interatomic distances, or the absolute value of the sum or differences of these sums or differences, and so on. In the end, each position vector of the N atoms must enter each SF via at least one interatomic distance to ensure the invariance with respect to all permutations. This is also the disadvantage of this approach, as constructing these SFs becomes increasingly complex as the system size increases. Furthermore, the problem of inflexiblity remains unsolved, as the NNP could only be used for systems with constant N.

The above example is an attempt at creating SFs that describe the whole system. A different approach is to use SFs that describe only the local environment of each single atom, called the central atom of its respective symmetry functions. A system of N atoms hence requires N sets of SFs. This does not yet solve the problem of permutational invariance however, as when gathering all SF values as the input vector of a single MLP, the MLP only yields a certain output for that exact order of SF values in the input vector. Swapping the position of two otherwise identical atoms leads to a different vector of SF values and therefore to a different output. If the vector of SF values of each atom is forwarded to its own atomic MLP is when the invariance with respect to permutation is ensured. The first implementation of this idea was done by Hobday [47], who gathered the SF values from information such as bond lengths and torsional angles from all atom triplets the central atom i could form with its first neighbors. Here, each atom i has its own atomic MLP subnet and the size of the input vector depends on the number of triplets that include atom i, which means that the subnets must be of variable size. A refinement of this idea was developed by Bholoa [48], who used quadruplets of atoms, where each quadruplet consists of atoms that are first neighbors with another atom of the quadruplet.

In contrast to this approach, which utilizes atomic MLPs of variable size, stands the method of Behler and Parrinello [12], where the atomic MLPs are of constant size, even if the local environment of the central atom changes. The information of the environment is similarly gathered from the relative positions of groups of atoms that include the central atom. However, the SFs of Behler and Parrinello sum up values over multiple of these atom groups, where in Hobday's approach each value would constitute its own SF. For instance, if the latter would utilize as an SF the function $f(r_{ij})$, with r_{ij} being the distance between the atoms i and j, then the Behler-Parrinello-approach would yield the sum

$$G_i = \sum_{j \neq i}^{N} f(r_{ij}) \tag{2.13}$$

as the SF.

2.2.2. Cutoff functions

As the sum in Equation 2.13 would theoretically include all atoms of the systems, a cutoff radius r_c is used that limits the local environment from which information is collected. This is realized by multiplying each term of the sum by a continuous cutoff function f_c, so that only neighbors within the cutoff radius to the central atom are considered when calculating the value of the SF. Another reason for the use of continuous cutoff functions is the calculation of forces, which requires the calculation of the spatial gradient of each SF. If a hard cutoff were to be used, i.e. using the Heaviside step function as the cutoff function, the discontinuity could give rise to inaccuracies when performing MD [49]. Using an appropriate cutoff function ensures a smooth decay of the SFs value and slope to zero at the cutoff radius. The cutoff function first proposed by Behler and Parrinello was [12]

$$f_{c,1}(r_{ij}) = \begin{cases} \frac{1}{2}\left[\cos\left(\pi \frac{r_{ij}}{r_c}\right) + 1\right], & 0 \leq r_{ij} \leq r_c \\ 0, & r_{ij} \geq r_c \end{cases} , \tag{2.14}$$

where r_{ij} is once again the distance between the central atom i and some other atom j. This function's second derivative however still has a discontinuity at the cutoff. Among many others, the following function has also been proposed, which possesses continuous first and second order derivatives [50]:

$$f_{c,2}(r_{ij}) = \begin{cases} \tanh^3\left(1 - \frac{r_{ij}}{r_c}\right), & 0 \leq r_{ij} \leq r_c \\ 0, & r_{ij} \geq r_c \end{cases} . \tag{2.15}$$

Both functions and their derivatives are shown in Figure 2.4

2.2.3. Radial symmetry functions

The first type of SF introduced by Behler and Parrinello [12] uses the interatomic distance r_{ij} to collect radial information about the central atom's environment and consists of Gaussians multiplied by the cutoff function

$$g_{ij} = e^{-\eta(r_{ij}-r_s)^2} f_c(r_{ij}) , \tag{2.16}$$

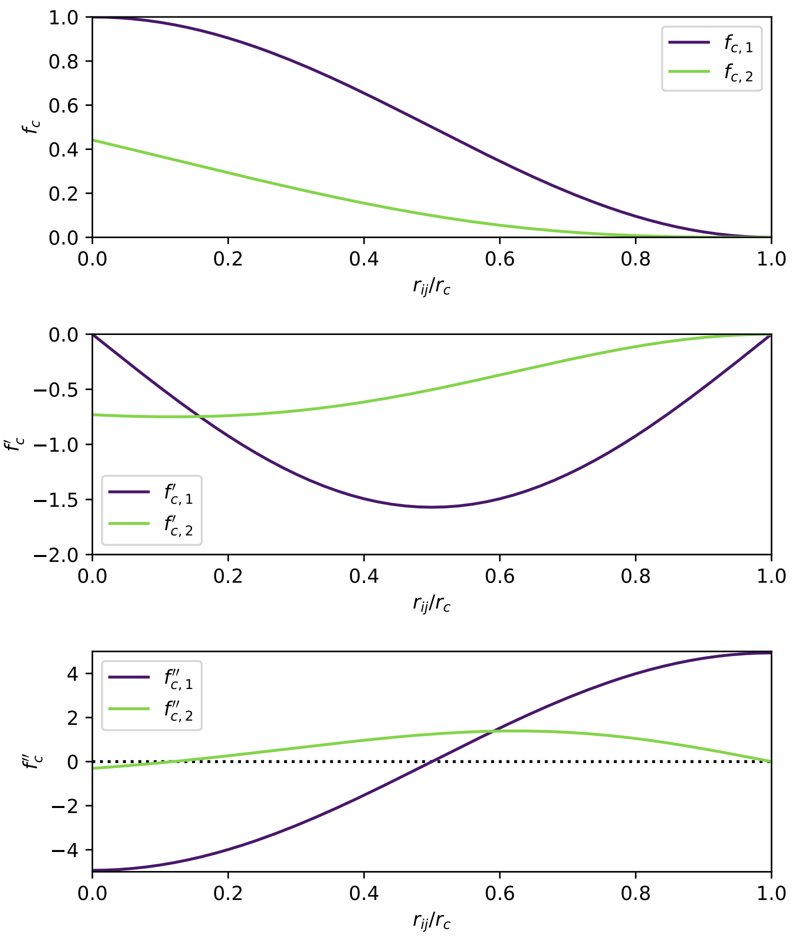


Figure 2.4.: Cutoff functions and their first and second order derivatives. The second derivative of $f_{c,1}$ does not decay to zero at the cutoff radius.

so that the SF is

$$G_i^2 = \sum_{j \neq i}^{N} g_{ij} = \sum_{j \neq i}^{N} e^{-\eta (r_{ij} - r_s)^2} f_c(r_{ij}) \ . \tag{2.17}$$

The parameters η and r_s control the width of the Gaussians and how much the Gaussians' centers are shifted, respectively (see Figure 2.5). Tuning these two parameters allows to sample different regions of the environment around the central atom. As mentioned in the previous section, the cutoff function ensures a smooth decay of the SF to zero. Additionally, since the cutoff functions decrease monotonically, neighbors closer to the central atom have a stronger influence on the value of the SF.

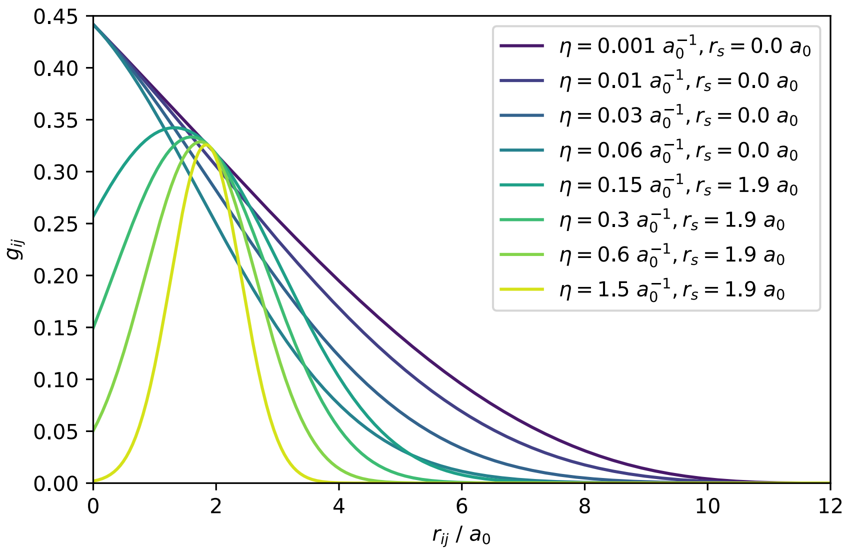


Figure 2.5.: Function graph of a single summand g_{ij} of the radial SF G_i^2 for different values of η and r_s. Values are given in terms of the Bohr radius $a_0 = 5.29 \cdot 10^{-11}$ m. The cutoff function $f_{c,2}$ (Equation 2.15) was used with a cutoff radius of $r_c = 12\,a_0$. The parameters chosen here constitute a set of 8 SFs as they will be used for hydrogen-hydrogen interactions later in this work.

2.2.4. Angular symmetry functions

The second type of SF introduced by Behler and Parrinello [12] describes the local environment through the use of atom triplets (i, j, k), with $i \neq j \neq k$, in addition to the interatomic distances of each atom pair present in this triplet, r_{ij}, r_{ik} and r_{jk}. This represents the many-body terms of all atomic positions inside the cutoff radius. The contribution of each triplet of the SF is

$$g_{ijk} = 2^{1-\zeta} \cdot \left[1 + \lambda \cos\left(\theta_{ijk}\right)\right]^{\zeta} \cdot e^{-\eta\left[(r_{ij}-r_s)^2+(r_{ik}-r_s)^2+(r_{jk}-r_s)^2\right]} \\ \cdot f_c(r_{ij}) f_c(r_{ik}) f_c(r_{jk})\ , \quad (2.18)$$

where θ_{ijk} is the angle between the three atoms of the triplet, centered on the central atom i. η and r_s are analogous parameters as for the radial SF G_i^2, and $\lambda = \pm 1$ and ζ are new parameters (see Figure 2.6). To again ensure a smooth decay to zero in the case of large interatomic separations, each summand is multiplied with a cutoff function for each atom pair. Summing over all atom triplets yields the SF

$$G_i^3 = \sum_{j\neq i}^{N} \sum_{\substack{k\neq i\\ k>j}}^{N} g_{ijk} = \sum_{j\neq i}^{N} \sum_{\substack{k\neq i\\ k>j}}^{N} 2^{1-\zeta} \cdot \left[1 + \lambda \cos\left(\theta_{ijk}\right)\right]^{\zeta} \cdot e^{-\eta\left[(r_{ij}-r_s)^2+(r_{ik}-r_s)^2+(r_{jk}-r_s)^2\right]} \cdot f_c(r_{ij}) f_c(r_{ik}) f_c(r_{jk}) \ . \quad (2.19)$$

To gather all angular information about the local environment, SFs processing even larger

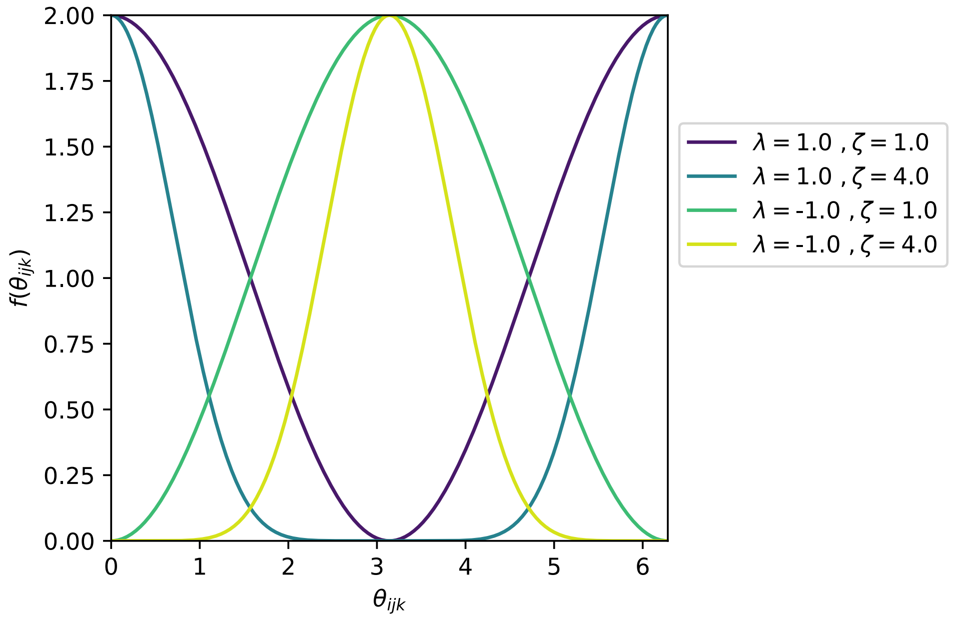


Figure 2.6.: Function graph of the angular contribution $f(\theta_{ijk}) = 2^{1-\zeta} \cdot \left[1 + \lambda \cos\left(\theta_{ijk}\right)\right]^{\zeta}$ of the angular SF for different values of λ and ζ. The parameters chosen here correspond to another set of SFs that will be used in this work.

atom groups — i.e. quadruplets, quintuplets, and so on — would be needed. However, the usage of only group sizes up to triplets is still justified to find a balance between accuracy and computational cost.

2.2.5. Element-dependent symmetry functions

It remains to be discussed how SFs distinguish elements. This distinction is important for systems containing atoms of more than just one element, as SFs should provide different values depending on the element of the central atom's neighbours, even if their relative positions would otherwise be the same. This can be formally done by modifying the SFs with the Kronecker delta

$$\delta_i^j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \ . \tag{2.20}$$

If the element of the central atom i is again denoted by χ_i, then an SF can be restricted to only gather information if the central atom is of some specific element α by writing

$$G_i\{\alpha\} = G_i \cdot \delta_{\chi_i}^{\alpha} \ . \tag{2.21}$$

Here, the curly brackets signify a version of the function that only collects information about central atoms of the element in the brackets. For SFs that use the interatomic distances of central atom i and its neighbouring atoms j, this formalism can be extended to require each atom j to be of a certain element β. In the case of G_i^2, this leads to

$$G_i^2\{\alpha,\beta\} = \delta_{\chi_i}^{\alpha} \sum_{j\neq i}^{N} g_{ij} \cdot \delta_{\chi_j}^{\beta} \ . \tag{2.22}$$

Note that α and β may or may not be the same element. The same approach can be used for SFs that acquire their value from atom triplets (i,j,k) by requiring that the two atoms j and k are of the elements β and γ. Additional care must be taken however, since the SF should not distinguish the order of the elements of the atoms j and k, but only check if one of them is of element β and the other of element γ. In the case of G_i^3, this leads to

$$G_i^3\{\alpha,\beta,\gamma\} = \delta_{\chi_i}^{\alpha} \sum_{j\neq i}^{N} \sum_{\substack{k\neq i \\ k>j}}^{N} g_{ijk} \cdot \left(\delta_{\chi_j}^{\beta}\delta_{\chi_k}^{\gamma} + \delta_{\chi_j}^{\gamma}\delta_{\chi_k}^{\beta} \left[1-\delta_{\beta}^{\gamma}\right] \right) \ . \tag{2.23}$$

The term in the square brackets ensures the correct behaviour of the function for the case where $\chi_j = \chi_k = \beta$ and $\beta = \gamma$.

To construct an efficient set of SFs, one would therefore choose multiple variations of G_i^2 and G_i^3 for different element combinations. For an NNP that describes a system of molecular water containing only the elements H and O, one could create a set of radial SFs $G_i^2\{H,H\}$, $G_i^2\{H,O\}$, $G_i^2\{O,H\}$ and $G_i^2\{O,O\}$, as well as angular SFs $G_i^3\{H,O,H\}$, $G_i^3\{H,O,O\}$, $G_i^3\{O,H,H\}$, $G_i^3\{O,O,H\}$ and $G_i^3\{O,O,O\}$. Each of these SF would then still have multiple variants using different function parameters.

2.3. Training a neural network

Keeping in mind that the width of the output and input layer of an MLP with L hidden layers is denoted as $m^{(0)}$ and $m^{(L+1)}$ respectively, one can introduce training data in the form of patterns. Each training pattern a consists of a vector of input data $\mathbf{X}_a \in \mathbb{R}^{m^{(0)}}$ and the corresponding output vector $\mathbf{Y}_a \in \mathbb{R}^{m^{(L+1)}}$. The training itself is then the process of feeding the MLP these training patterns and adjusting the weights of the MLP to reproduce the expected output $\mathbf{Y}_a$ as closely as possible. The measure of how well the

MLP approximates the data is given by a cost function Γ, commonly chosen as the root-mean-square error

$$\Gamma = \sqrt{\frac{1}{M}\sum_{a=1}^{M}\left(\mathbf{Y}_a - \mathbf{y}_a^{L+1}\right)^2} , \tag{2.24}$$

where M is the number of training patterns and $\mathbf{y}_a^{L+1}$ is a vector whose entries are the values of the output nodes of the MLP when receiving the input $\mathbf{X}_a$. There are different optimization algorithms that achieve the minimization of the cost function by successively adjusting the weight parameters, many of which utilize the derivatives of Γ with respect to the weight parameters. To calculate these derivatives one can use a method called backward propagation of errors, or backpropagation for short, which was shown to be applicable to the training of neural networks by Werbos in 1974 [51]:

Remembering the formulation of the neural network as a function Φ of nested functions and weights (see Equation 2.4), it is clear that the cost function can also be described as a function of Φ, and thus of the network's weight parameters $\mathbf{w}$:

$$\Gamma = \Gamma(\Phi(\mathbf{x}, \mathbf{w})) = \Gamma(\varphi^L(\mathbf{w}^L, \varphi^{L-1}(\mathbf{w}^{L-1}, \varphi^{L-2}(\ldots, \varphi^0(\mathbf{w}^0, \mathbf{x})\ldots)))) . \tag{2.25}$$

Now, let γ_{ij}^l denote the derivative of the cost function Γ with respect to the weight w_{ij}^l:

$$\gamma_{ij}^l = \frac{d\Gamma}{dw_{ij}^l} . \tag{2.26}$$

Then it can be shown through use of the chain rule that γ_{ij}^l can be calculated for each layer iteratively by using the result of the gradient of the next layer, until one arrives at the output layer. This gives rise to the formula

$$\gamma_{ij}^{l-1} = \sum_{k=0}^{m^{(l-1)}} \gamma_{0k}^l w_{jk}^l \frac{df_j^l}{du_j^l} y_i^{l-1} , \tag{2.27}$$

which holds for all $l \leq L-1$.

2.3.1. Kalman filter training

The original Kalman filter algorithm [38] optimizes the estimate of linear dynamical systems' unknown states by using their complete history of possibly noisy measurements instead of just a single data point of an observable. This allows to minimize the root-mean-square error with respect to an entire available data set. The algorithm is also designed recursively, storing only the last state and an error covariance matrix, which are updated with each new data set. The soon after developed extended Kalman filter is also able to deal with non-linear systems and was later found to be applicable to neural network training [52].

Let the M available training patterns again be described as a vector of input data $\mathbf{X}_a$ and the vector corresponding to the values of the m output nodes $\mathbf{Y}_a = [Y_1(\mathbf{X}_a), \ldots, Y_m(\mathbf{X}_a)]^T$, where $a = 1, \ldots, M$ signifies the different training patterns. The set of training patterns can be seen as a time series of measurements, where for each measurement, the error between reference data and neural network prediction is calculated. The Kalman filter iteratively optimizes the weight parameters for each measurement to yield a minimum error for all training patterns in the time series. Let the current state vector be the vector containing all n of the neural network's weight parameters and be denoted as $\mathbf{w}(t) = [w_1(t), \ldots, w_n(t)]^T$, where t shall be the iteration time step where training pattern a is used to update the state vector. Further, let $\mathbf{y}_a(t) = [y_1(\mathbf{w}(t), \mathbf{X}_a), \ldots, y_m(\mathbf{w}(t), \mathbf{X}_a)]^T$ be the predicted output values of the neural network to the input vector $\mathbf{X}_a$ with the current weights. A single Kalman filter iteration then goes as follows:

The error vector is calculated as

$$\boldsymbol{\xi}(t) = \mathbf{Y}_a - \mathbf{y}_a(t) \; . \tag{2.28}$$

The derivative matrix $\mathbf{H}$, a $n \times m$ matrix containing the derivatives of the error vector with respect to all weight parameters, whose elements are written as

$$H_{ij}(t) = \frac{\partial}{\partial w_i} \xi_j(t) \; , \tag{2.29}$$

is calculated. The $m \times m$ scaling matrix $\mathbf{A}$ is introduced as

$$\mathbf{A}(t) = \left(\frac{1}{\eta(t)} \mathbf{I} + \mathbf{H}^T(t)\mathbf{P}(t)\mathbf{H}(t) \right)^{-1} \; , \tag{2.30}$$

where the learning rate $\eta(t)$ is a free hyperparameter that may be varied over time, $\mathbf{I}$ is the identity matrix, and $\mathbf{P}$ is the error covariance matrix with dimension $n \times n$. With this, the Kalman gain matrix $\mathbf{K}$ can be calculated:

$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}(t)\mathbf{A}(t) \; . \tag{2.31}$$

The iteration step is then finally concluded by updating the state vector $\mathbf{w}$ and the error covariance matrix $\mathbf{P}$ according to

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mathbf{K}(t)\boldsymbol{\xi}(t) \tag{2.32}$$

$$\mathbf{P}(t+1) = \mathbf{P}(t) - \mathbf{K}(t)\mathbf{H}^T(t)\mathbf{P}(t) + \mathbf{Q}(t) \; , \tag{2.33}$$

where $\mathbf{Q}$ is the process noise covariance matrix, which introduces artificial noise to improve the fit quality and prevents trapping in poor local minima [53]. Iterating the steps of equations 2.28 through 2.33 for all M training patterns is referred to as one neural network training epoch. Over the course of one epoch, the cost function

$$\Gamma = \sum_t \boldsymbol{\xi}^T(t)\boldsymbol{\xi}(t) \tag{2.34}$$

is minimized, where the sum runs over the time steps covering all training patterns used in the epoch. It should be noted that a fixed order of training patterns used in every epoch should be avoided due to the recency effect [54], where groups of reference data being treated in the same order over and over leads to an overemphasis of their characteristics, reducing the overall quality of the training results.

The remaining problem, influencing the quality of the training results, is the initialization of η, $\mathbf{P}$ and $\mathbf{Q}$. Recommendations in literature [54] suggest a learning rate η between 10^{-3} and 1. The error covariance matrix can be initialized as

$$\mathbf{P}(0) = \varepsilon^{-1}\mathbf{I} \, , \tag{2.35}$$

where ε lies between 10^{-3} and 10^{-2}. The artificial process noise is commonly added via

$$\mathbf{Q}(t) = q(t)\mathbf{I} \, , \tag{2.36}$$

where $q(t)$ is fixed somewhere between 10^{-6} and 10^{-2}. Alternatively, it can be added with

$$q(t) = \max\left(q_0 e^{-t/\tau_q}, q_{min}\right) \, , \tag{2.37}$$

choosing the hyperparameters in a way so that the noise reaches a minimum value after a few training epochs.

2.3.2. Application of Kalman filter training to HDNNPs

To apply the Kalman filter method to HDNNPs in the Behler/Parrinello-approach, the formalism is extended to include both total potential energies and atomic forces [18]. In this case, instead of patterns, the training data set comprises atomic structures, their total potential energies and their atomic forces, calculated from some reference method. The reference potential energy of structure a shall be denoted as E_a^{ref} and the forces of the N_a atoms in structure a as $\mathbf{F}_{a1}^{ref}, \ldots, \mathbf{F}_{aN_a}^{ref}$. In the same vein as Equation 2.8, the total predicted energy is then written as a sum of atomic energy contributions

$$E_a = \sum_{i=1}^{N_a} \epsilon^{\chi_{ai}}(\mathbf{w}^{\chi_{ai}}, \mathbf{G}_i(\mathbf{X}_a)) \, , \tag{2.38}$$

where χ_{ai} denotes the chemical element of atom i in the reference structure a and $\epsilon^{\chi_{ai}}$ is the value of the output neuron of the atomic subnet for that element. When S^{χ} is the number of symmetry functions for element χ, then $\mathbf{G}_i(\mathbf{X}_a) = \{G_{i,1}^{\chi_{ai}}(\mathbf{X}_a), \ldots, G_{i,S^{\chi_{ai}}}^{\chi_{ai}}(\mathbf{X}_a)\}$ is the set of symmetry functions of atom i, which are fed the input data in the form of the atomic coordinates $\mathbf{X}_a = \left(\vec{X}_{a1}^T, \ldots, \vec{X}_{aN_a}^T\right)^T$. The vector $\mathbf{w}^{\chi} = \left[w_1^{\chi}, \ldots, w_{n_\chi}^{\chi}\right]^T$ contains all n_χ weight parameters of the subnet for the atomic species χ. When again having c such chemical elements, the weight vectors for each can be gathered in a combined weight vector that fully describes the state of the system:

$$\mathbf{w} = \left(\mathbf{w}^{\chi_1 T}, \ldots, \mathbf{w}^{\chi_c T}\right)^T \, . \tag{2.39}$$

Since there are now $3N_a$ reference forces per structure, but only one reference energy, it is randomly chosen whether a Kalman filter weight update is based on an energy or a force component. This way an overemphasis on reference forces is prevented and the energies and forces are treated as independent information that form the artificial time series of training patterns. Depending on whether the weight vector update is based on an energy or a reference force in α-direction, the error vector takes the form

$$\boldsymbol{\xi}(t) = \left[E_a^{ref} - E_a(\mathbf{w}(t))\right] \tag{2.40}$$

or

$$\boldsymbol{\xi}(t) = \left[\beta(F_{ai,\alpha}^{ref} - F_{ai,\alpha}(\mathbf{w}(t)))\right] , \tag{2.41}$$

where the hyperparameter β controls the importance of force updates relative to energy updates. The force is computed by the HDNNP analytically by

$$\begin{aligned} F_{ai,\alpha} &= -\frac{\partial E_a(\mathbf{w}(t))}{\partial x_{ai,\alpha}} \\ &= -\sum_{j=1}^{N_a} \frac{\partial}{\partial x_{ai,\alpha}} \epsilon^{\chi_{aj}}(\mathbf{w}^{\chi_{aj}}(t), \mathbf{G}_j(\mathbf{X}_a)) \\ &= -\sum_{j=1}^{N_a} \sum_{k=1}^{S^{\chi_{aj}}} \frac{\partial \epsilon^{\chi_{aj}}}{\partial G_{j,k}^{\chi_{aj}}}(\mathbf{w}^{\chi_{aj}}(t), \mathbf{X}_a) \frac{\partial G_{j,k}^{\chi_{aj}}}{\partial x_{ai,\alpha}}(\mathbf{X}_a) . \end{aligned} \tag{2.42}$$

Using these adjusted equations with the Kalman filter from Equations 2.28 through 2.33, the weight vector $\mathbf{w}$ is optimized to minimize the cost function

$$\Gamma = \sum_{a=1}^{M} (E_a^{ref} - E_a)^2 + \beta^2 \sum_{a=1}^{M} \sum_{i=1}^{N_a} (\vec{F}_{ai}^{ref} - \vec{F}_{ai})^2 \tag{2.43}$$

within one epoch. It remains to clarify how the derivative matrix $\mathbf{H}$ from Equation 2.29 is calculated, which in this case is now a derivative vector. For energies, the lth vector entry, assuming the weight l belongs to the subnet of element χ, is

$$H_l(t) = -\sum_{i=1}^{N_a} \delta_{\chi}^{\chi_{ai}} \frac{\partial}{\partial w_l^{\chi}} \epsilon^{\chi}(\mathbf{w}^{\chi}, \mathbf{G}_i(\mathbf{X}_a)) , \tag{2.44}$$

which can be calculated via backpropagation (see Equation 2.27). In the case of forces, one finds the more involved derivative

$$\begin{aligned} H_l(t) &= -\beta \frac{\partial}{\partial w_l^{\chi}} F_{ai,\alpha} = \beta \frac{\partial E_a(\mathbf{w}(t))}{\partial w_l^{\chi} \partial x_{ai,\alpha}} \\ &= \beta \sum_{j=1}^{N_a} \sum_{k=1}^{S^{\chi}} \delta_{\chi}^{\chi_{ai}} \frac{\partial^2 \epsilon^{\chi}}{\partial w_l^{\chi} \partial G_{j,k}^{\chi}}(\mathbf{w}^{\chi}(t), \mathbf{X}_a) \frac{\partial G_{j,k}^{\chi}}{\partial x_{ai,\alpha}}(\mathbf{X}_a) . \end{aligned} \tag{2.45}$$

Since the subnets of other elements do not depend on w_l^{χ}, the sums only contain terms of element χ, facilitated here by the Kronecker delta. This shows that subnets of different species are very loosely coupled during Kalman filter training, which can be further exploited to reduce computational efforts. Every term of Equations 2.28 through 2.33 may be indexed with a χ to create independent per-element Kalman filters. The per-element definitions of the error vectors for energy updates

$$\boldsymbol{\xi}^{\chi}(t) = \left[\frac{N_a^{\chi}}{N_a} (E_a^{ref} - E_a(\mathbf{w}(t))) \right] , \tag{2.46}$$

where N_a^{χ} is the number of atoms of element χ in the training structure a, and for force updates

$$\boldsymbol{\xi}^{\chi}(t) = \left[\beta \frac{N_{ai}^{\chi}}{N_{ai}} (F_{ai,\alpha}^{ref} - F_{ai,\alpha}(\mathbf{w}(t))) \right] , \tag{2.47}$$

where N_{ai} is the number of neighbouring atoms within the cutoff radius r_c and N_{ai}^{χ} is the number of neighbours with element χ, were found to give suitable results comparable to using a global Kalman filter [18].

3. Path Integral Molecular Dynamics

Although equivalent to the Heisenberg and Schrödinger pictures of quantum mechanics, the formulation of Feynman provides an easier connection to statistical mechanics with his use of path integrals [20, 55]. To give a brief introduction, as well as a more detailed explanation of the application to molecular dynamics in the following sections, this chapter roughly follows the work of Tuckerman [22], which provides an excellent explanation of the topic.

First, consider a particle localized at point x that evolves unobserved to point x'. So, the exact path the particle takes before arriving at x' is impossible to know, not only because it is not observed, but also because of its quantum mechanical nature. In Feynman's picture, it is not unknown which path the particle takes, but rather the particle follows all infinitely many possible paths simultaneously. To find the complete amplitude of the process, the amplitude of each path must be calculated, which are then summed up. If each path has an amplitude A_i, then the total amplitude is $A = A_1 + A_2 + A_3 + \ldots$ and the corresponding probability becomes

$$P = |A_1 + A_2 + A_3 + \ldots|^2 \, . \tag{3.1}$$

The cross terms resulting from the square of this sum of these complex amplitudes are then the interference terms between the paths. In other words, the infinitely many paths are interfering alternatives, whose interference gives rise to the interference patterns we know from processes such as the double-slit experiment.

A helpful heuristic visualization by Feynman [55] is to imagine a large number of gratings between points x and x'. For each grating, the particle may pass through any of a grating's slits before reaching the next grating, where it can again move through an arbitrarily chosen slit, and so on. The number of possible paths the particle can take increases with the number of gratings and the number of slits in the gratings. If one now takes the limit of infinitely many gratings, each with infinitely many slits, then there are infinitely many possible paths and the picture essentially reverts to a particle in empty space. This visualization is shown in Figure 3.1.

The relevant theoretical foundation of path integrals is laid out in Section 3.1 and applied to molecular dynamics in Section 3.2. Section 3.3 discusses NQEs as the motivation of using PIMD, as well as their occurance in water.

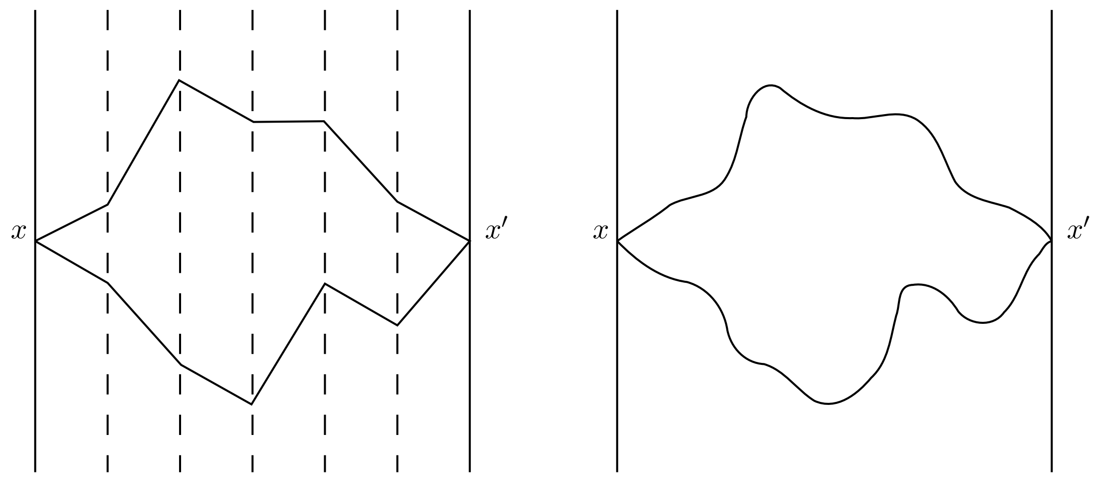


Figure 3.1.: Two possible paths from x to x' through multiple intermediate gratings (left) and in free space (right).

3.1. Mathematical Description of Feynman Path Integrals

To formally discuss the mathematics of path integrals, it is easiest to start with a single one-dimensional particle. The Hamiltonian of this particle is

$$\hat{H} = \frac{\hat{p}^2}{2m} + U(\hat{x}) \equiv \hat{K} + \hat{U} \tag{3.2}$$

with the mass of the particle m, the potential U and the momentum- and position operators $\hat{p}$ and $\hat{x}$. As in the example above, the particle of interest is initially prepared at a point x and the question is what the amplitude for detecting the particle at point x' will be. Now, let the coordinate-space matrix elements of the canonical density matrix $\hat{\rho}\,(\beta)$ be

$$\rho(x, x'; \beta) \equiv \langle x'|e^{-\beta\hat{H}}|x\rangle \ , \tag{3.3}$$

where $|x\rangle$ and $|x'\rangle$ are eigenstates of the position operator, and $\beta = 1/k_BT$ is the inverse temperature with the Boltzmann constant k_B and temperature T. Note that through formal manipulation, one can find that the density matrix

$$\hat{\rho}\,(\beta) = e^{-\beta\hat{H}} = e^{-\frac{i}{\hbar}(-i\beta\hbar)\hat{H}} \tag{3.4}$$

is the quantum-mechanical propagator evaluated at the imaginary time $t = -i\beta\hbar$. Conversely, one can say that the density matrix yields the propagator when evaluated at the imaginary inverse temperature $\beta = it/\hbar$:

$$\hat{\rho}\,(it/\hbar) = e^{-\frac{i}{\hbar}t\hat{H}} \tag{3.5}$$

This paints the picture of the density matrix being a propagator in imaginary time.

Since $\hat{H}$ is a sum of two non-commuting operators, $\hat{K}$ and $\hat{U}$, the operator $e^{-\beta\hat{H}}$ can not be easily evaluated. Here, the Trotter theorem [56] can be used, which states for two operators $\hat{A}$ and $\hat{B}$ for which the commutator $\left[\hat{A}\,,\hat{B}\right] \neq 0$,

$$e^{\hat{A}+\hat{B}} = \lim_{P\to\infty}\left(e^{\hat{B}/2P}e^{\hat{A}/P}e^{\hat{B}/2P}\right)^P \ , \tag{3.6}$$

where P is an integer. This allows to write

$$e^{-\beta(\hat{K}+\hat{U})} = \lim_{P\to\infty}\left(e^{-\beta\hat{U}/2P}e^{-\beta\hat{K}/P}e^{-\beta\hat{U}/2P}\right)^P \ . \tag{3.7}$$

Defining now an operator $\hat{\Omega}$ by

$$\hat{\Omega} = e^{-\beta\hat{U}/2P}e^{-\beta\hat{K}/P}e^{-\beta\hat{U}/2P} \tag{3.8}$$

and substituting Equations 3.7 and 3.8 into Equation 3.3 yields:

$$\begin{aligned}\rho(x,x';\beta) = \langle x'|e^{-\beta(\hat{K}+\hat{U})}|x\rangle &= \lim_{P\to\infty}\langle x'|\left(e^{-\beta\hat{U}/2P}e^{-\beta\hat{K}/P}e^{-\beta\hat{U}/2P}\right)^P|x\rangle \\ &= \lim_{P\to\infty}\langle x'|\hat{\Omega}^P|x\rangle\end{aligned} \tag{3.9}$$

One can now insert an identity operator

$$\hat{I} = \int dx\,|x\rangle\langle x| \tag{3.10}$$

between each factor of $\hat{\Omega}^P$, introducing a total of $P-1$ integrations and resulting in the expression for the density matrix

$$\rho(x,x',\beta) = \lim_{P\to\infty}\int dx_2\cdots dx_P\ \langle x'|\hat{\Omega}|x_P\rangle\langle x_P|\hat{\Omega}|x_{P-1}\rangle\langle x_{P-1}|\cdots|x_2\rangle\langle x_2|\hat{\Omega}|x\rangle\ . \tag{3.11}$$

This step is analogous to the insertion of $P-1$ gratings, as described in the heuristic picture above and depicted in Figure 3.1. The integration over each variable x_i represents the summation over all possible paths a particle can take through each grating. As P goes to infinity, more gratings are inserted, until one once again arrives at the picture of free space.

Considering a general matrix element as it appears in Equation 3.11

$$\langle x_{k+1}|\hat{\Omega}|x_k\rangle = \langle x_{k+1}|e^{-\beta\hat{U}/2P}e^{-\beta\hat{K}/P}e^{-\beta\hat{U}/2P}|x_k\rangle\ , \tag{3.12}$$

one can now evaluate this element in closed form. Since $\hat{U} = U(\hat{x})$ is a function of the position operator, and $|x_{k+1}\rangle$ and $|x_k\rangle$ are position eigenvectors, they are also eigenvectors of $\exp\left(-\beta U(\hat{x})/2P\right)$ with eigenvalues $\exp\left(-\beta U(x_{k+1})/2P\right)$ and $\exp\left(-\beta U(x_k)/2P\right)$, respectively. Thus, Equation 3.12 can be written as

$$\langle x_{k+1}|\hat{\Omega}|x_k\rangle = e^{-\beta U(x_{k+1})/2P}\langle x_{k+1}|e^{-\beta\hat{K}/P}|x_k\rangle e^{-\beta U(x_k)/2P}\ . \tag{3.13}$$

To evaluate $\exp(-\beta\hat{K}/P) = \exp(-\beta\hat{p}^2/2mP)$, one can again insert a identity operator of the form

$$\hat{I} = \int dp\, |p\rangle\langle p| \,, \tag{3.14}$$

which leads to the operator acting on one of its eigenvectors, yielding

$$\begin{aligned}\langle x_{k+1}|e^{-\beta\hat{K}/P}|x_k\rangle &= \int dp\, \langle x_{k+1}|e^{-\beta\hat{K}/P}|p\rangle\langle p|x_k\rangle \\ &= \int dp\, \langle x_{k+1}|p\rangle\langle p|x_k\rangle e^{-\beta p^2/2mP} \,.\end{aligned} \tag{3.15}$$

For position and momentum eigenstates, it holds that

$$\langle x|p\rangle = \frac{1}{\sqrt{2\pi\hbar}} e^{ipx/\hbar} \tag{3.16}$$

and applying this formula to Equation 3.15 leads to

$$\langle x_{k+1}|e^{-\beta\hat{K}/P}|x_k\rangle = \frac{1}{2\pi\hbar}\int dp\, e^{ipx_{k+1}/\hbar} e^{-ipx_k/\hbar} e^{-\beta p^2/2mP} \,. \tag{3.17}$$

Completing the square in the exponents found in Equation 3.17

$$\begin{aligned}-\frac{\beta}{2mP}p^2 + \frac{i(x_{k+1}-x_k)}{\hbar}p &= -\frac{\beta}{2mP}\left[p^2 - \frac{2imP(x_{k+1}-x_k)}{\beta\hbar}p\right] \\ &= -\frac{\beta}{2mP}\left\{\left[p - \frac{imP(x_{k+1}-x_k)}{\beta\hbar}\right]^2 + \frac{m^2P^2(x_{k+1}-x_k)^2}{\beta^2\hbar^2}\right\} \\ &= -\frac{\beta}{2mP}\left[p - \frac{imP(x_{k+1}-x_k)}{\beta\hbar}\right]^2 - \frac{mP(x_{k+1}-x_k)^2}{2\beta\hbar^2}\end{aligned} \tag{3.18}$$

and substituting

$$\tilde{p} = p - \frac{imP(x_{k+1}-x_k)}{\beta\hbar} \tag{3.19}$$

allows to carry out the momentum integration as a Gaussian integral:

$$\begin{aligned}\langle x_{k+1}|e^{-\beta\hat{K}/P}|x_k\rangle &= \frac{1}{2\pi\hbar}e^{-mP(x_{k+1}-x_k)^2/2\beta\hbar^2}\int_{-\infty}^{\infty} d\tilde{p}\, e^{-\beta\tilde{p}^2/2mP} \\ &= \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{1/2}\exp\left[-\frac{mP}{2\beta\hbar^2}(x_{k+1}-x_k)^2\right]\end{aligned} \tag{3.20}$$

Combining this with the findings from Equation 3.13 leads to the general matrix element

$$\begin{aligned}\langle x_{k+1}|\hat{\Omega}|x_k\rangle = \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} &\exp\left[-\frac{\beta}{2P}(U(x_{k+1}) + U(x_K))\right] \\ &\cdot \exp\left[-\frac{mP}{2\beta\hbar^2}(x_{k+1}-x_k)^2\right] \,.\end{aligned} \tag{3.21}$$

Going back to Equation 3.11 and evaluating all P matrix elements according to Equation 3.21 yields the density matrix

$$\rho(x,x',\beta) = \lim_{P\to\infty} \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx_2 \cdots dx_P \\ \cdot \exp\left\{-\frac{1}{\hbar}\sum_{k=1}^{P}\left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2 + \frac{\beta\hbar}{2P}(U(x_{k+1})+U(x_k))\right]\right\}\Bigg|_{x_1=x}^{x_{P+1}=x'} , \quad (3.22)$$

where x_1 and x_{P+1} are fixed as the original starting and end points of the path, x and x', respectively. The integrations over x_2 to x_P are the summations over all possible paths in the imaginary time $-i\beta\hbar$, or, in the case of finite P, linear paths between discrete imaginary time points.

3.1.1. The Canonical Partition Function

To approach the application of path integrals to molecular dynamics, the canonical partition function, which is important for statistical mechanics, can be derived. Considering the canonical partition function $Q(L,T)$ for a system at temperature T confined to $x \in [0,L]$ and using $Q(L,T) = Tr[\exp(-\beta\hat{H})]$, where Tr is the trace, one can evaluate the trace in the coordinate basis:

$$Q(L,T) = \int_0^L dx\, \langle x|e^{-\beta\hat{H}}|x\rangle = \int_0^L dx\, \rho(x,x;\beta)\ . \quad (3.23)$$

The diagonal elements of the density matrix correspond to Equation 3.22 when setting $x_1 = x_{P+1} = x$. If instead naming the integration variable x_1, the partition function becomes

$$Q(L,T) = \lim_{P\to\infty} \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx_1 \cdots dx_P \\ \cdot \exp\left\{-\frac{1}{\hbar}\sum_{k=1}^{P}\left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2 + \frac{\beta\hbar}{2P}(U(x_{k+1})+U(x_k))\right]\right\}\Bigg|_{x_{P+1}=x_1} . \quad (3.24)$$

Using that

$$\frac{1}{2}\sum_{k=1}^{P}[U(x_k)+U(x_{k+1})] = \frac{1}{2}[U(x_1)+U(x_2)+U(x_2)+U(x_3)+\ldots \\ \ldots + U(x_{P-1})+U(x_P)+U(x_P)+U(x_1)] = \sum_{k=1}^{P} U(x_k)\ , \quad (3.25)$$

the canonical partition function can finally be written as

$$Q(L,T) = \lim_{P\to\infty} \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx_1 \cdots dx_P \\ \cdot \exp\left\{ -\frac{1}{\hbar} \sum_{k=1}^{P} \left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2 + \frac{\beta\hbar}{P} U(x_k) \right] \right\}\Bigg|_{x_{P+1}=x_1} . \quad (3.26)$$

The condition $x_{P+1} = x_1$ means that the paths are restricted to begin and end at the same point, i.e. they are cyclic paths.

3.1.2. Expectation Values and estimators

To receive the expectation value of a Hermitian operator $\hat{A}$, it is first noted that

$$\langle \hat{A} \rangle = \frac{1}{Q(L,T)} Tr\left[\hat{A}\, e^{-\beta \hat{H}}\right] = \frac{1}{Q(L,T)} \int dx \langle x|\hat{A}\, e^{-\beta\hat{H}}|x\rangle \ , \quad (3.27)$$

with the trace being expanded in the coordinate basis. If $\hat{A}$ is only a function of $\hat{x}$, then $|x\rangle$ is an eigenvector of $\hat{A}\,(\hat{x})$ with eigenvalue $a(x)$. Thus one finds

$$\langle \hat{A} \rangle = \frac{1}{Q(L,T)} \int dx\, a(x) \langle x|e^{-\beta\hat{H}}|x\rangle \ . \quad (3.28)$$

Using Equation 3.22 with $x = x'$ leads to

$$\langle \hat{A} \rangle = \frac{1}{Q(L,T)} \lim_{P\to\infty} \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx_1 \cdots dx_P\, a(x_1) \\ \cdot \exp\left\{ -\frac{1}{\hbar} \sum_{k=1}^{P} \left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2 + \frac{\beta\hbar}{P} U(x_k) \right] \right\}\Bigg|_{x_{P+1}=x_1} . \quad (3.29)$$

This path integral expression for the expectation value of $\hat{A}\,(\hat{x})$ unfortunately only evaluates $a(x)$ at that point, which is problematic for practical uses. All points $x_1, \dots, x_P$ are however equal for the cyclic path, which can be shown by the invariance of the exponential's argument under a cyclic relabeling of the coordinate variables:

$$x'_2 = x_1, \quad x'_3 = x_2, \quad \dots \quad x'_P = x_{P-1}, \quad x'_1 = x_P \ . \quad (3.30)$$

Applying this relabeling to Equation 3.29 yields the equivalent expression

$$\langle \hat{A} \rangle = \frac{1}{Q(L,T)} \lim_{P\to\infty} \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx'_1 \cdots dx'_P\, a(x'_2) \\ \cdot \exp\left\{ -\frac{1}{\hbar} \sum_{k=1}^{P} \left[\frac{mP}{2\beta\hbar}(x'_{k+1}-x'_k)^2 + \frac{\beta\hbar}{2P} (U(x'_{k+1}) + U(x'_k)) \right] \right\}\Bigg|_{x'_{P+1}=x'_1} . \quad (3.31)$$

This type of relabeling can be performed P times, yielding an equivalent expression with $a(x_3'')$, $a(x_4''')$ and so on. Adding all these expressions together and dividing them by P then leads to

$$\langle \hat{A}\,\rangle = \frac{1}{Q(L,T)} \lim_{P\to\infty} \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx_1 \cdots dx_P \, \left[\frac{1}{P}\sum_{k=1}^{P} a(x_k)\right] \cdot \exp\left\{-\frac{1}{\hbar}\sum_{k=1}^{P}\left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2 + \frac{\beta\hbar}{P}U(x_k)\right]\right\}\Bigg|_{x_{P+1}=x_1} . \tag{3.32}$$

This expression treats all P coordinates $x_1, \ldots, x_P$ as equal.

The expectation value from Equation 3.32 can be approximated for finite P through the use of an appropriate estimator depending on the P coordinates $x_1, \ldots, x_P$. Firstly, dropping the limit from Equation 3.26 leaves a discrete partition function

$$Q_P(L,T) = \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx_1 \cdots dx_P \cdot \exp\left\{-\frac{1}{\hbar}\sum_{k=1}^{P}\left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2 + \frac{\beta\hbar}{P}U(x_k)\right]\right\}\Bigg|_{x_{P+1}=x_1} , \tag{3.33}$$

which becomes the partition function $Q(L,T)$ again in the limit $P \to \infty$. With Equation 3.33, the probability distribution function

$$f(x_1,\ldots,x_P) = \frac{1}{Q_P(L,T)} \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \cdot \exp\left\{-\frac{1}{\hbar}\sum_{k=1}^{P}\left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2 + \frac{\beta\hbar}{P}U(x_k)\right]\right\}\Bigg|_{x_{P+1}=x_1} \tag{3.34}$$

can be defined. An appropriate estimator for $\langle \hat{A}\,\rangle$ from Equation 3.32 is obviously

$$a_P(x_1,\ldots,x_P) = \frac{1}{P}\sum_{k=1}^{P} a(x_k) \tag{3.35}$$

that yields the approximate expectation value by averaging it with respect to the probability distribution function $f(x_1,\ldots,x_P)$. This shall be written as

$$\langle \hat{A}\,\rangle_P = \langle a_P(x_1,\ldots,x_P)\rangle_f \ , \tag{3.36}$$

where the true expectation value $\langle \hat{A}\,\rangle$ is recovered in the limit $P \to \infty$ as

$$\langle \hat{A}\,\rangle = \lim_{P\to\infty} \langle \hat{A}\,\rangle_P \ . \tag{3.37}$$

This scheme can be directly applied to yield the estimator for the potential energy of the system, given by

$$U_P(x_1,\dots,x_P) = \frac{1}{P}\sum_{k=1}^{P} U(x_k)\ . \tag{3.38}$$

Other expectation values and their estimators may not be as straightforward to derive and it must also be noted, that when $\hat{A}$ is a function of the momentum operator, the expectation value is no longer a cyclic path, which leads to slow convergence and increasingly worse performance with increasing P.

Some expectation values can be evaluated via thermodynamic relations. The average energy

$$E = \langle \frac{\hat{p}^2}{2m} + U(\hat{x})\rangle \tag{3.39}$$

is a function of both the momentum and position operator, which would otherwise be cumbersome. Here it can be used that

$$E = -\frac{\partial}{\partial\beta}\ln Q(L,T) = -\frac{1}{Q(L,T)}\frac{\partial Q(L,T)}{\partial\beta}\ , \tag{3.40}$$

which yields

$$\begin{aligned} E =& \frac{1}{Q(L,T)}\lim_{P\to\infty}\left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2}\int dx_1\cdots dx_P\,\varepsilon_P(x_1,\dots,x_P) \\ &\cdot\exp\left\{-\frac{1}{\hbar}\sum_{k=1}^{P}\left[\frac{mP}{2\beta\hbar}(x_{k+1}-x_k)^2+\frac{\beta\hbar}{P}U(x_k)\right]\right\}\Bigg|_{x_{P+1}=x_1} \\ =& \lim_{P\to\infty}\langle\varepsilon_P(x_1,\dots,x_P)\rangle_f \end{aligned} \tag{3.41}$$

Here,

$$\varepsilon_P(x_1,\dots,x_P) = \frac{P}{2\beta} - \sum_{k=1}^{P}\left[\frac{mP}{2\beta^2\hbar^2}(x_{k+1}-x_k)^2+\frac{1}{P}U(x_k)\right] \tag{3.42}$$

is the estimator for the energy that converges to the true thermodynamic energy E when taking $\langle\varepsilon_P(x_1,\dots,x_P)\rangle_f$ in the limit $P\to\infty$. Similarly, the thermodynamic relations of the canonical ensemble may be used to derive estimators for other quantities. Due to the convergence problems mentioned above, other energy estimators were devised, like the virial energy estimator based on the virial theorem [57]

$$\varepsilon_{vir}(x_1,\dots,x_P) = \frac{1}{P}\sum_{k=1}^{P}\left[\frac{1}{2}x_k\frac{\partial U}{\partial x_k}+U(x_k)\right]\ , \tag{3.43}$$

which handles fluctuations much better than its primitive counterpart from Equation 3.42 and is insensitive to P.

3.2. Application of Path Integrals to Molecular Dynamics

Knowing the framework of path integrals, their application to molecular dynamics may be explored. The discrete partition function from Equation 3.33 can be manipulated to resemble the classical canonical partition function of a cyclic polymer chain. By defining $m' = mP/(2\pi\hbar)^2$, the prefactor in Equation 3.33 can be expressed as the Gaussian integrals over the variables $p_1, \ldots, p_P$

$$\int dp_1 \cdots dp_P \, e^{-\beta \sum_{k=1}^{P} \frac{p_k^2}{2m'}} = \left(\frac{2\pi m'}{\beta}\right)^{P/2} = \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} . \tag{3.44}$$

When additionally introducing the chain frequency

$$\omega_P = \frac{\sqrt{P}}{\beta\hbar} , \tag{3.45}$$

Equation 3.33 can be written as

$$Q_P(L,T) = \int dp_1 \cdots dp_P \int dx_1 \cdots dx_P \cdot \exp\left\{-\beta \sum_{k=1}^{P} \left[\frac{p_k^2}{2m'} + \frac{1}{2} m\omega_P^2 (x_{k+1} - x_k)^2 + \frac{1}{P} U(x_k)\right]\right\}\Bigg|_{x_{P+1}=x_1} . \tag{3.46}$$

This was coined as classical isomorphism [21], as it represents a classical cyclic polymer chain of P points, or "beads" (see Figure 3.2), with a nearest-neighbor coupling constant $m\omega_P^2$ moving in a classical potential $U(x)/P$. Due to its origins in the density operator or imaginary time propagator, the P beads are also indexed by their "imaginary time index". Sampling the partition function of a classical cyclic polymer chain thus allows to sample the approximate quantum canonical distribution and obtain approximate quantum properties, which become more exact as P increases. Note that the "mass" m' appearing in the kinetic energy term can be chosen freely, as it stems from a prefactor that does not affect equilibrium averages.

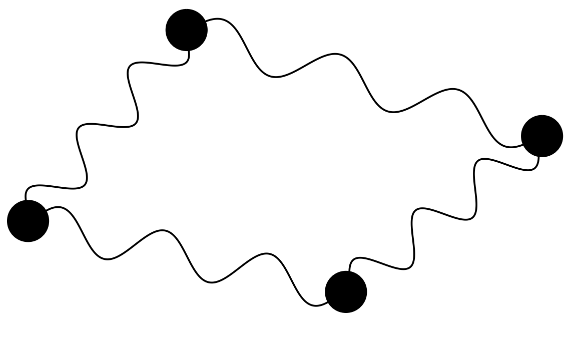


Figure 3.2.: Cyclic polymer chain with $P = 4$.

3.2.1. Normal Mode PIMD

The equations of motion that can be derived from Equation 3.46 suffer from slow convergence due to a broad spectrum of normal mode frequencies. That is, since the time step chosen for any MD simulation must be small enough to accommodate the highest frequency, the large-scale changes of the cyclic polymer chain associated with the low normal mode frequencies are only adequately sampled for very long simulation times. To alleviate this, one can try to uncouple the harmonic term in Equation 3.46 and replace the masses in a way so that only a single harmonic frequency remains. One way to achieve this is to use what is called a normal mode transformation [23] that can be constructed in the following way:

First, generate the matrix

$$A_{i,j} = 2\delta_{i,j} - \delta_{i,j-1} - \delta_{i,j+1} \tag{3.47}$$

with $i, j = 1, \dots, P$. Then, diagonalize the matrix, yielding the eigenvectors as well as the eigenvalues

$$\lambda_{2k-2} = \lambda_{2k-1} = 2\left[1 - \cos\left(\frac{2\pi(k-1)}{P}\right)\right] . \tag{3.48}$$

From the eigenvectors, construct the matrix $O_{i,j}$ that diagonalizes A. The set of normal mode variables $u_1, \dots, u_P$ are then constructed according to

$$u_k = \frac{1}{\sqrt{P}} \sum_{l=1}^{P} O_{k,l} x_l , \tag{3.49}$$

with the inverse transformation

$$x_k = \sqrt{P} \sum_{l=1}^{P} O^T_{k,l} u_l . \tag{3.50}$$

This allows to write

$$\sum_{k=1}^{P} (x_{k+1} - x_k)^2 = \sum_{k=2}^{P} \lambda_k u_k^2 . \tag{3.51}$$

Thus, the harmonic term is now separable. Defining lastly the masses

$$\begin{aligned} m_k &= m\lambda_k \\ m'_1 &= m \\ m'_k &= m_k \end{aligned} \tag{3.52}$$

the transformed partition function becomes

$$Q_P(L,T) = \int dp_1 \cdots dp_P \int du_1 \cdots du_P \cdot \exp\left\{ -\beta \sum_{k=1}^{P} \left[\frac{p_k^2}{2m'_k} + \frac{1}{2} m_k \omega_P^2 u_k^2 + \frac{1}{P} U(x_k(u)) \right] \right\} , \tag{3.53}$$

where $x_k(u)$ represents the coordinates transformed via Equation 3.50. Equation 3.53 can be evaluated using the classical Hamiltonian

$$H = \sum_{k=1}^{P} \left[\frac{p_k^2}{2m_k'} + \frac{1}{2} m_k \omega_P^2 u_k^2 + \frac{1}{P} U(x_k(u)) \right] \tag{3.54}$$

with the equations of motion

$$\begin{aligned} \dot{u}_k &= \frac{p_k}{m_k'} \\ \dot{p}_k &= -m_k \omega_P^2 u_k - \frac{1}{P} \frac{\partial U}{\partial u_k} \,. \end{aligned} \tag{3.55}$$

The forces on the normal mode variables are obtained by applying the chain rule

$$\begin{aligned} \frac{1}{P} \frac{\partial U}{\partial u_1} &= \frac{1}{P} \sum_{l=1}^{P} \frac{\partial U}{\partial x_l} \\ \frac{1}{P} \frac{\partial U}{\partial u_k} &= \frac{1}{\sqrt{P}} \sum_{l=1}^{P} \frac{\partial U}{\partial x_l} O_{l,k}^T \,. \end{aligned} \tag{3.56}$$

If coupled to a thermostat, these equations of motion can be integrated by an MD algorithm of choice to sample the quantum canonical distribution. Furthermore, a convenient property of the normal mode transformation is that the variable u with imaginary time index 1 is the average over all path variables and corresponds to the center of mass of the cyclic polymer chain, also known as the "path centroid", and the force acting on u_1 is the average force, as can be seen in the first line of Equation 3.56.

The scheme presented in this section can easily be extended to N Boltzmann particles in d dimensions. Let $\mathbf{r}_i^{(1)}, \ldots, \mathbf{r}_i^{(P)}$ be the d-dimensional coordinates of the P beads of the path that represents particle i. Noting that the subscript i denotes the particle and the superscript (k) denotes the imaginary time index of the beads, the discrete partition functions takes the form

$$\begin{aligned} Q_P(N,V,T) \sim & \int \prod_{i=1}^{N} d\mathbf{r}_i^{(1)} \cdots d\mathbf{r}_i^{(P)} d\mathbf{p}_i^{(1)} \cdots d\mathbf{p}_i^{(P)} \\ & \cdot \exp\left\{ -\beta \sum_{k=1}^{P} \left[\sum_{i=1}^{N} \frac{(\mathbf{p}_i^{(k)})^2}{2m_i^{(k)\prime}} + \sum_{i=1}^{N} \frac{1}{2} m_i^{(k)} \omega_P^2 (\mathbf{r}_i^{(k+1)} - \mathbf{r}_i^{(k)})^2 + \frac{1}{P} U(\mathbf{r}_1^{(k)}, \ldots, \mathbf{r}_N^{(k)}) \right] \right\} \end{aligned} \tag{3.57}$$

with $\mathbf{r}_i^{(P+1)} = \mathbf{r}_i^{(1)}$. The potential $U(\mathbf{r}_i^{(k)}, \ldots, \mathbf{r}_N^{(k)})$ only acts between beads with the same imaginary time index, as is illustrated in Figure 3.3. Also, in this case the virial estimator from Equation 3.43 can be used to yield an estimator for the kinetic energy as [57, 58]

$$T_P(\mathbf{r}_1, \ldots, \mathbf{r}_N) = \frac{N}{2\beta} + \frac{1}{2P} \sum_{i=1}^{N} \sum_{k=1}^{P} (\mathbf{r}_i^{(k)} - \mathbf{r}_i^c) \cdot \frac{\partial U(\mathbf{r}_1^{(k)}, \ldots, \mathbf{r}_N^{(k)})}{\partial \mathbf{r}_i^{(k)}} \,, \tag{3.58}$$

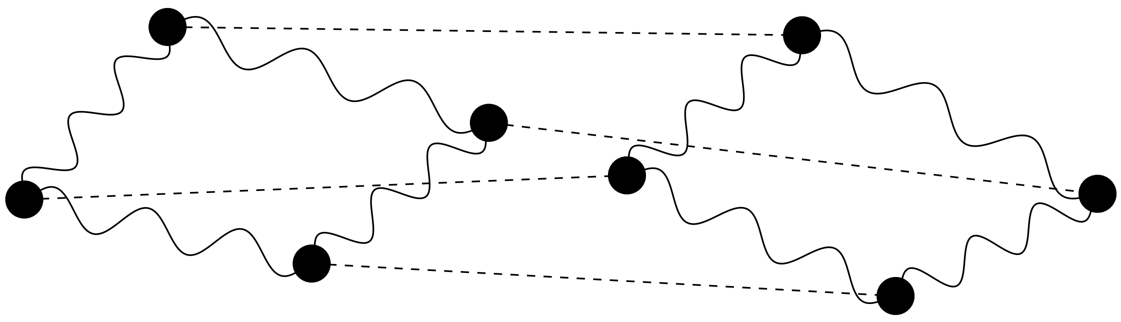


Figure 3.3.: Two quantum particles represented as cyclic polymer chains with $P = 4$. Only beads with the same imaginary time index interact with each other, as indicated by the dashed lines.

where

$$\mathbf{r}_i^c = \frac{1}{P}\sum_{k=1}^{P}\mathbf{r}_i^{(k)} \tag{3.59}$$

is the centroid coordinate of the polymer ring representing the particle i.

A classical Hamiltonian can again be constructed from Equation 3.57 by carrying out a normal mode transformation for each particle:

$$H = \sum_{k=1}^{P}\left[\sum_{i=1}^{N}\frac{(\mathbf{p}_i^{(k)})^2}{2m_i^{(k)\prime}} + \sum_{i=1}^{N}\frac{1}{2}m_i^{(k)}\omega_P^2(\mathbf{u}_i^{(k)})^2 + \frac{1}{P}U(\mathbf{r}_1^{(k)}(\mathbf{u}_1),\ldots,\mathbf{r}_N^{(k)}(\mathbf{u}_N))\right] . \tag{3.60}$$

The equations of motion and forces are acquired analogous to Equations 3.55 and 3.56.

3.3. Motivation: Nuclear Quantum Effects

Finally it would be appropriate to address, why path integral methods are needed and when they are used. Thanks to the path integral formalism, simulations can be carried out that treat atoms, or more precisely their nuclei, quantum mechanically. This allows to capture NQEs, such as quantum tunneling and zero-point motion. These effects are most likely to be exhibited by light nuclei, such as hydrogen, for which NQEs occur even at room temperature [59, 60]. So, whenever NQEs are expected to be important, PIMD may be used to include them, while for simulations of heavier elements or when NQEs can be neglected, classical molecular dynamics can be safely applied instead to not further the computational cost.

It is also important to consider the type of potential-energy model used when deciding whether or not to use PIMD for a given application. When using an empirical potential-energy model with parameters that are derived from fits to experimental data, using PIMD would "double count" NQEs, since the experiments the model is based is naturally quantum mechanical and NQEs thus implicitly included. However, empirical potentials may be adapted for use with path integral methods by parameterizing them to reproduce certain quantities in quantum instead of classical simulations. This has been done for example with the popular TIP4P water model, creating the q-TIP4P/F model [61]. On the other hand, when potential energies and forces are calculated from the electronic structure via ab initio methods [62], NQEs are explicitly not included. Such models are thus only completely correct when using them with PIMD or other methods incorporating NQEs.

3.3.1. Nuclear Quantum Effects in Water

As suggested before, hydrogen is the prime candidate for the study of NQEs. The low mass of the single proton nucleus allows for phenomena such as proton tunneling, zero-point energy and delocalization, which in turn affect the static and dynamic properties of water. Understanding these NQEs is vital to understanding these properties and the anomalous behaviour of water. Interestingly, quantum effects can affect water in seemingly contradicting ways: One the one hand, NQEs allows protons to be more strongly delocalized between hydrogen-bonded pairs of water molecules, enabling a lengthening of the covalent bond and effectively strengthening the bond. On the other hand, the delocalization of the proton also allows for a greater angular deviation from and thus distortion of the bond, weakening it [63, 64]. Which of these effects — stretching or bending — dominates depends on the distance between the oxygen (O) atoms of the pair of hydrogen-bonded water molecules, with short bonds getting stronger through the inclusion of NQEs and long bonds getting weaker. This interplay has been dubbed "competing quantum effects".

The effect of NQEs in water may be gauged by exploring isotope effects. By substituting the hydrogen (H) in water with deuterium (D) or tritium (T), the comparison of heavy water (D_2O) and tritiated water (T_2O) with water reveals the differences in properties caused by NQEs. Note however that while D and T are heavier than H, they are still not free from NQEs, as they are light enough to exhibit them themselves, albeit less than H. Thus, it is more a study of "strong NQEs vs. weak NQEs", instead of "NQEs vs. no NQEs". Experimental results have shown that the water dimer dissociation energy of the O-D bond in the gas phase (14.88 kJ/mol) is 12.7% higher than that of the O-H bond (13.2 kJ/mol) [65, 66], suggesting a significantly stronger hydrogen bond as NQEs get weaker. A number of observations support this view and even extend it lower temperatures, where for example the melting point of water increases upon deuteration and tritiation by 3.82 K and 4.49 K, respectively, indicating that stronger hydrogen bonds lead to a higher stability of ice [67, 68, 69]. X-ray scattering experiments provide further evidence that at room temperature, the hydrogen bonds of D_2O are stronger than those of H_2O [70].

While the above mentions are consistent with the idea that NQEs weaken the hydrogen bond, care must be taken, as competing quantum effects may impact thermodynamic properties in counter-intuitive ways. Molar densities show a slight increase in liquid H_2O compared to D_2O and T_2O [68], which could be interpreted as the presence of stronger hydrogen bonding in H_2O. However, stronger hydrogen bonding can actually lead to a decrease in density due to the directional nature of those bonds, which was also observed in DFT simulations [71]. Ab initio PIMD simulations performed for various H-bonded systems indicate the trend that weak hydrogen bonds are weakened by NQEs while strong hydrogen bonds become stronger upon their inclusion [63]. The critical temperature of H_2O (647.10 K) is higher than the ones of D_2O (643.85 K) and T_2O (641.66 K) [67, 68], which suggests that NQEs strengthen hydrogen bonds at higher temperatures. The liquid/vapor surface tension also shows seemingly anomalous behaviour, with values decreasing slightly from 71.98 mN/m to 71.87 mN/m upon deuteration at 25 °C [68], which could pay tribute to the modulation of the local hydrogen-bonded structure in different phases or at interfaces [27]. It must also be noted that competing quantum effects may almost completely cancel each other out, making their study a delicate matter [72].

Seeing how some of the above results either stem from simulations or were experiments augmented by simulational data, the usefulness of PIMD in the study of water is obvious. Still, PIMD is computationally quite expensive, which is why the field is very active in developing new algorithms and methods to accelerate simulations, and progress is further facilitated through new hardware and the successful inclusion of machine learning and ANNs [30, 31, 73]. As always, great care must be taken to prevent mispredictions of numerical methods, as was shown that this can happen with DFT calculations, though correction schemes can alleviate such problems [74, 75, 76].

4. Software

To perform NMPIMD simulations using HDNNPs, multiple software has to work in tandem. The universal force engine i-PI [32] acts as a server that deals with the nuclear motion of the system in a given thermodynamic ensemble and applies the PIMD scheme. The calculation of the forces used to propagate these nuclear motions are outsourced to other software, that connects multiple instances of itself to i-PI akin to clients of the server. In this case, LAMMPS [33] will be used to handle the force calculations. The n2p2 package [18] contains the tools to train an HDNNP and provides an interface to let LAMMPS use the HDNNP for its force calculations. Afterwards, the Pytim package [77] provides the means to investigate the configurations resulting from the simulations in regards to a systems different phases. In this chapter, these four software packages and some of their methods will be briefly discussed.

4.1. LAMMPS

LAMMPS [33] stands for Large-scale Atomic/Molecular Massively Parallel Simulator and, as the name suggests, is a code designed for classical molecular dynamics simulations using multiple CPUs parallelly. With a focus on materials modeling, it contains a plethora of options to model different atomic and molecular interactions and potentials and is usable for both soft matter and solid-state materials, each on the atomic, meso or continuum scale. The code's strength lies in its parallel algorithm that decomposes the simulation domain and assigns each processor a fixed spatial region, allowing for efficient parallelization and to make good use of the computational power of high-performance computing (HPC) clusters [78]. LAMMPS offers an abundance of functions to accommodate the simulation of a wide variety of problems, which is further extended by the ability to add custom code due to being open source under the GNU General Public License v2.0.

4.1.1. NVT ensemble in LAMMPS

In LAMMPS, commands that apply operations to the simulated system during time stepping or minimization are known as "fixes". Such operations could be the updating of atom positions and velocities during time integration or the controlling of the temperature through the use of a thermostat. As this work deals with systems with constant particle number, volume and temperature, the fix used to sample the NVT ensemble is of interest.

The NVT fix propagates atoms using the equations of motions by Shinoda et al. [79], which in turn are integrated using either the velocity-Verlet or the r-RESPA algorithm [80]. The fix also applies a thermostat in the form of a Nosé-Hoover chain, or, if the chain's length is set to 1, a standard Nosé-Hoover thermostat.

4.2. n2p2

n2p2 [81, 18] is a C++ based package containing libraries and applications used for training HDNNPs in the Behler-Parrinello-approach as well as applying these HDNNPs to MD calculations. Before training, a data set, containing atomic configurations together with forces and total potential energies, is needed as reference data for the PES one wants to approximate. For these configurations to cover the possible atomic environments and resulting high-dimensional PES densely enough to safely interpolate between reference points, a number of reference structures of the order of $10^3 - 10^4$ should be aimed for. Also needed is an input file that details the topology of the HDNNP to be trained, i.e. the width and depth of the MLPs and the types of activation- and symmetry functions used, and sets the hyperparameters used for the training. As a mandatory step, the n2p2 component `nnp-scaling` computes and normalizes the SFs set in the input file. Optionally, the data set can also be normalized with the component `nnp-norm`. The component `nnp-train` performs the actual training of the NNP, using the given reference data to optimize the weight parameters based on the Kalman filter method.

Various additional components are also included in the n2p2 package, some to further optimize reference data and hyperparameters, others to directly apply the resulting NNP to a problem. Most notably, the package includes a library to interface the NNP created with `nnp-train` with other applications, such as LAMMPS. That way, the option to perform force calculations with the newly trained NNP becomes available in LAMMPS [82].

4.2.1. Multistream Extended Kalman Filter

By and large, the `nnp-train` routine uses the Kalman filter method as it was explained in Section 2.3.2. To better utilize multi core processor architecture however, the approach was refined to an algorithm called the multistream extended Kalman filter (MS-EKF). As the time series of sequential training patterns is fictitious and only pieces of the entire reference data are used for updating the state vector $\mathbf{w}$ at any one time, the opportunity for parallelization seems obvious. While for the regular Kalman filter training of an HDNNP, one input and one output vector of a single "measurement" optimize the unknown weights, multistream training uses multiple input-output pairs at the same time with the simultaneous measurements sharing the same weights. Formally, the error vector is extended to contain data from s measurements or "streams"

$$\boldsymbol{\xi}(t) = \left[\boldsymbol{\xi}_1^T(t), \ldots, \boldsymbol{\xi}_s^T(t)\right]^T \tag{4.1}$$

and the derivative matrix is appropriately enlarged to include the derivatives from the respective streams

$$\mathbf{H}(t) = \left[\frac{\partial}{\partial w_1}, \ldots, \frac{\partial}{\partial w_n}\right]^T \boldsymbol{\xi}^T(t) = \left[\mathbf{H}_1(t), \ldots, \mathbf{H}_s(t)\right] \ . \tag{4.2}$$

All other expressions from Equations 2.30 through 2.33 are unchanged save for the dimensions of matrices $\mathbf{A}$ and $\mathbf{K}$, which are extended accordingly. The training performance can be significantly improved this way [83, 84], as multiple data sets enter for a coordinated weight update accommodating all involved training patterns. The application of the MS-EKF to HDNNP training is as simple as letting data from multiple reference structures enter into the error and derivative vectors.

4.2.2. Warnings

As detailed before, the vectors $\mathbf{G}_i$ contain the general coordinates, i.e. the SF set values, of each central atom i. Each SF vector describes the central atom's chemical environment and can be seen as a point in a multidimensional space spanned by the SFs. Since the number of SF vectors appearing in a training set is limited, NNP simulations are practically guaranteed to run into new SF vectors that did not occur during training. The larger the distance between these points in multidimensional space, representing the SF vectors from training and the new SF vectors during simulation, is, the higher the chance that the prediction of the NNP is poor. While n2p2 could theoretically warn users when that happens, there is no clear rule how large the distance may be before warranting a warning. Additionally, the calculation of the distance between one point to a plethora of others in multidimensional space is computationally expensive.

Another problem which may occur concerns extrapolation. During an NNP simulation, whenever at least one component of an SF vector is smaller than the minimum value or greater than the maximum value of the same component obtained during training, the NNP is extrapolating when predicting its output. This is undesirable, as NNPs show poor performance when extrapolating. n2p2 constantly checks if SF vector components lie below the minimum value or above the maximum value of the corresponding components appearing during training and issues a warning if that happens during an NNP simulation [85]. It should be noted, that an NNP simulation without extrapolation warnings is not automatically correct and, vice versa, NNP simulations showing extrapolation warnings are not automatically incorrect. However, a high number of extrapolation warnings justifies enlarging the training set to eliminate the extrapolation errors or otherwise investigating the cause of the warnings.

4.3. i-PI

i-PI [86, 32] is a Python package used for path integral molecular dynamics according to the methods explained in Chapter 3, created by Ceriotti as well as many collaborators. It uses a socket interface, allowing external driver codes (such as LAMMPS) to perform the calculations of inter-atomic forces and potential energies and sharing them with i-PI. The clients can communicate with the sockets via internet or locally using UNIX-domain sockets. i-PI itself then performs the time integration by means of molecular dynamics or Monte Carlo as well as the output of properties, before relaying the updated atom positions back to the external code for the next time step's force and energy calculations. Since in the PIMD scheme, each atom is represented by a ring polymer of P beads, the complete atomic system can be seen as a collection of P replica systems, each corresponding to the beads with the same imaginary time index. When P instances of the driver code connect to i-PI, each client can then deal with one of the replicas, before the data from each client is gathered by i-PI again for time integration. This also fulfills one of the self-stated goals of i-PI, which is to provide an easily implementable method of dealing with quantum nuclear effects, independent of the particular client code. Figure 4.1 shows a schematic overview of the client-server-model used.

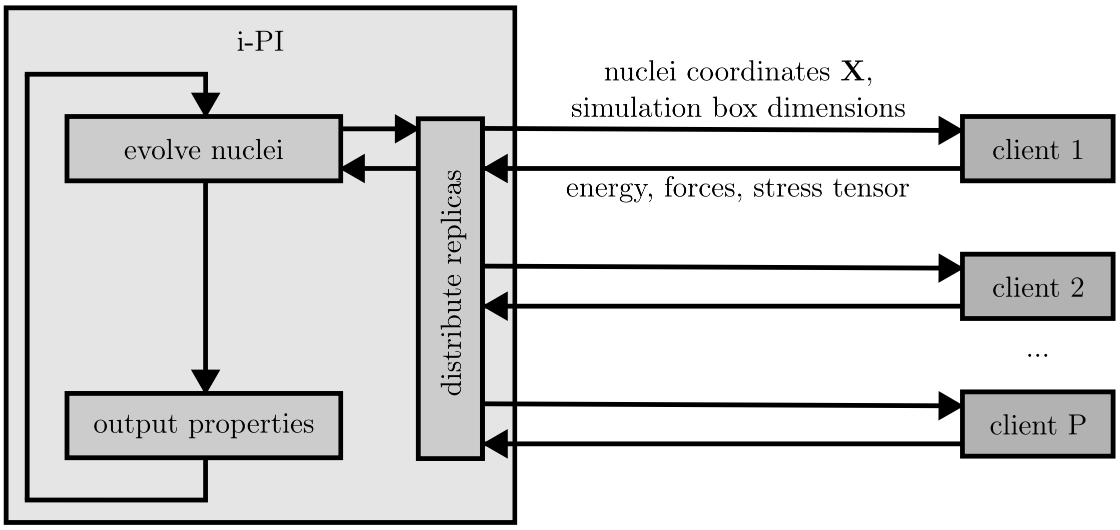

i-PI
evolve nuclei
output properties
distribute replicas
nuclei coordinates **X**, simulation box dimensions
energy, forces, stress tensor
client 1
client 2
...
client P

Figure 4.1.: Schematic of the server-client interaction of i-PI. P clients of the driver code are connected to i-PI. The client code is only sent nuclei coordinates ($\mathbf{X}$) and the dimensions of the simulation box, and computes the energy, forces and stress tensor, which are then used by i-PI to evolve the nuclei.

i-PI incorporates many other features that aid PIMD in some way. Perhaps the most important endeavor is the reduction of computation time: If P replicas of the same system are evolved, one could expect that the computational cost would be P times as high as if treating the system classically. The number of beads required to reach converged results depends on the model considered. For example, 6 beads are typically enough when simulating a rigid-body water model, whereas flexible water models need up to 40 beads. i-PI uses polymer contraction schemes to reduce the effective number of beads without sacrificing accuracy [87, 88].

4.3.1. NVT ensemble in i-PI

For the integration of the equations of motion, i-PI splits the Hamiltonian into the sum of a free ring polymer part

$$H_P = \sum_{k=1}^{P} \left[\sum_{i=1}^{N} \frac{(\mathbf{p}_i^{(k)})^2}{2m_i^{(k)\prime}} + \sum_{i=1}^{N} \frac{1}{2} m_i^{(k)} \omega_P^2 (\mathbf{u}_i^{(k)})^2 \right] \tag{4.3}$$

and the remaining potential energy part (see also Equation 3.60) and uses the exact evolutions of both parts in a symplectic integration scheme [58]. Herein, the free ring polymer part underwent the normal mode transformation detailed in Section 3.2.1. The scheme exactly evolves the system half a time step under the potential energy part, transforms to the normal mode representation for a full time step of exact evolution under the influence of the free ring polymer part H_P, and then performs another half time step under the potential energy part after transforming back from the normal mode representation.

Among many other choices, i-PI implements a white noise Langevin equation thermostat that is applied before and after the integration steps mentioned above. While formally sampling the canonical distribution in classical statistical mechanics, it can be applied to PIMD as it is simply classical statistical mechanics in an extended phase space. As all degrees of freedom and all beads are thermostatted locally, it is referred to as path integral Langevin equation local (PILE-L) thermostat. It however may inhibit the proper sampling of properties sensitive to the slow, collective motion of all atoms of the system. This is why it can be combined with a global velocity rescaling thermostat that replaces the local thermostat of only the centroid mode, creating the global PILE-G thermostat [89, 90]. The full integration scheme including the description of PILE thermostats can be found in a paper by Ceriotti et al. [58], where they have shown that these thermostats perform as well as Nosé-Hoover chain thermostats while being computationally cheaper.

4.4. Pytim

When examining the water/vapor interface, one might be interested in the properties of the surface layers of the liquid phase, and it is thus helpful to be able to identify said surface layer by layer. In this work, the Python-based Pytim package [77] is employed, which utilizes the framework provided by the MDAnalysis package [91] and provides several methods to identify and analyse interfaces, their surface layers, and their properties.

One method to identify which molecules of a given interfacial configuration belong to which surface layer is the Identification of Truly Interfacial Molecules (ITIM) [92] algorithm, implemented by Pytim. For ITIM, a ficticious probe sphere of radius R_{ps} is moved along a number of straight test lines to approach the interface in question. The test lines are perpendicular to the interface and either arranged randomly or in a grid, but ultimately close enough that the probe sphere can cover the whole area of the interface. When the probe sphere moves along a test line coming from one phase, as soon as it hits a molecule of the other phase, the probe sphere is stopped and the molecule that stopped it is regarded as interfacial (see Figure 4.2). A sensible choice for the radii must be made: To determine when an atom is hit, it must have a defined radius, which may be the atom's van der Waals radius or a fraction of its Lennard-Jones distance parameter σ. The probe sphere radius R_{ps} must also be taken into account, as values too small may lead to a rugged surface and values too large may lead to a smoothed out surface. Sensible choices of this free parameter depend on the system, but are usually of the same order of magnitude as the size of the particles making up the phase to be studied. In principle, the process is then repeated for all test lines, though the algorithm may be sped up by going over all molecules instead and check for test lines along which a probe sphere would be stopped by the given molecule. After having identified all molecules in the first surface layer, the process may be repeated ignoring the molecules of the first layer, yielding the second surface layer, and so on.

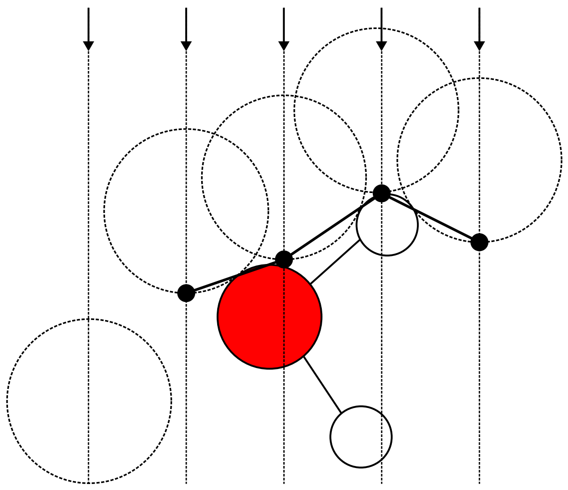


Figure 4.2.: Visualisation of the ITIM method on the example of a water molecule. The dashed lines and circles represent the test lines and probe spheres, respectively. The surface can be estimated by taking the intersection of the test lines and the probe spheres at the position where they were stopped, pictured here as solid black circles connected by bold lines signifying the estimated surface. Note that the leftmost probe sphere is not stopped by the water molecule and would continue on its path.

To aid with the separation of phases, a given configuration may be prefiltered by employing the Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [93]. It uses the notions of "density reachability" and "density connectivity" to differentiate clusters from noise, or in this context, phases of high density (i.e. the liquid phase) from phases of low density (i.e. the vapor phase). Simply put, the algorithm defines molecules as part of the same cluster when a certain number density threshold in the neighbourhood around the molecules is exceeded. In the Pytim implementation, this threshold is determined automatically [94]

5. Training the neural network potential

To use an HDNNP for computations in an MD simulation, it must first be trained. This chapter details the data used for training and the models used to generate this data (Section 5.1), the training process and the structure of the HDNNP created (Section 5.2), and the training results and relevant metrics (Section 5.3).

5.1. Data used

The NNP was trained using data provided by Dr. Marcello Sega and can be found online as a Zenodo repository [95]. The training data set consists of 8007 reference structures of water, including interfacial configurations of the water/vapor interface. Each structure is made up of up to a few hundred atoms and contains the configurational data as well as the forces and total energies used for the training, totalling 1085016 atoms, 3255048 force components and 8007 energies.

The training data set is based on DFT simulations employing the RPBE-D3 functional [9, 10], which were carried out by Morawietz et al. [11] using the Fritz Haber Institute "ab initio molecular simulations" (FHI-aims) code package [96]. Wohlfahrt augmented this data set with 400 configurations of the liquid/vapor interface generated with the Flexible Simple Point-Charge (SPC/F) empirical model [97], which were perturbed for improved sampling of the neighborhoods of the free energy minima. After computing the DFT forces and energies of this augmented data set with FHI-aims, an NNP was trained with the results and used to create a trajectory, from which 500 frames were taken and added to the augmented data set in a self-consistency refinement step, yielding the final training data set [17].

5.1.1. Density Functional Theory

To perform a truly ab-initio MD simulation of a set of atoms, one would need to solve the full Schrödinger equation for the system of electrons and nuclei. This approach quickly becomes unfeasible even for small systems, which makes approximations and numerical methods necessary. DFT is a popular method to calculate the ground state-energy of electrons of many body systems. It makes a number of approximations, the first of which is the Born-Oppenheimer approximation [98]. It states that due to the big mass difference between the electrons and the nuclei, it can be assumed that the electrons adiabatically follow the nuclei. The nuclei act on the electrons only as a static external potential, and the electrons are at each instant in the ground state corresponding to the configuration of the nuclei [99]. Then, the problem left to solve for an n-electron molecule is

$$\hat{H}\Psi = E\Psi \ , \tag{5.1}$$

with the wave function Ψ, energy E and the Hamiltonian, in atomic units ($\hbar = e = m = 1$),

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{n}\nabla_i^2 + \sum_{i=1}^{n} v(\vec{r}_i) + \sum_j \sum_{i>j} \frac{1}{|\vec{r}_i - \vec{r}_j|} \ . \tag{5.2}$$

Here, $\vec{r}_i$ is the position vector of electron i and $v(\vec{r}_i)$ denotes the potential energy

$$v(\vec{r}_i) = -\sum_\alpha \frac{Z_\alpha}{|\vec{r}_i - \vec{R}_\alpha|} \tag{5.3}$$

between electron i and the nuclei, with the position vectors $\vec{R}_\alpha$ and atomic numbers Z_α. This quantity, called the external potential acting on electron i, is only dependent on the position vector $\vec{r}_i$ of the electron as variables, as the nuclei positions are assumed to be fixed. Treating Equation 5.1 as a variational problem to find the ground state-energy is still a hard problem to solve, as the wave function Ψ depends on $3n$ spatial and n spin coordinates. In contrast, DFT utilizes the continuous electron probability density, which is a function of only three spatial variables

$$\rho(\vec{r}) = n \int d\vec{r}_1 d\vec{r}_2 \cdots d\vec{r}_{n-1} |\Psi(\vec{r}, \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_{n-1})|^2 \ . \tag{5.4}$$

Hohenberg and Kohn showed that for molecules with a nondegenerate ground state, the ground state electron probability density ρ_0 uniquely determines the ground state energy E_0 and ground state wave function Ψ_0, as well as other electronic properties [100]. Giving the theory its name, E_0 is thus the functional of ρ_0

$$E_0 = E_0[\rho_0] \tag{5.5}$$

and Equation 5.1 can be recast as a variational problem minimizing E by varying ρ instead of Ψ

$$E_0 = \min_\rho E_v[\rho] \ . \tag{5.6}$$

The subscript v denotes the dependence of E_0 on the external potential $v(\vec{r})$. Furthermore, one can write

$$E_v[\rho] = \int \rho(\vec{r})v(\vec{r})d\vec{r} + \frac{1}{2}\int \frac{\rho(\vec{r})\rho(\vec{r}')}{|\vec{r} - \vec{r}'|} d\vec{r}d\vec{r}' + F[\rho] \ . \tag{5.7}$$

The first and second terms on the right hand side represent the interaction energy of the electrons with the external potential and the classical Coulomb energy, respectively. The remaining functional F, uniquely defined by ρ, is not dependant on the external potential and contains all contributions necessary to make Equation 5.7 equal to the sought after expectation value E_0. However, while the theorem of Hohenberg and Kohn states that it is possible to calculate ground state properties from ρ_0, it does not specify how. For practical application, F must be approximated.

Kohn and Sham introduced the concept of fictitious non-interacting electrons, that have the same density as the true interacting system [101]. This allows to write

$$E_v[\rho] = \int \rho(\vec{r})v(\vec{r})d\vec{r} + \frac{1}{2}\int \frac{\rho(\vec{r})\rho(\vec{r}')}{|\vec{r}-\vec{r}'|}d\vec{r}d\vec{r}' + 2\sum_i \left\langle \psi_i \middle| -\frac{\nabla^2}{2} \middle| \psi_i \right\rangle + E_{XC}[\rho] \; . \quad (5.8)$$

The third term is the kinetic energy of the non-interacting electrons and the functional $E_{XC}[\rho]$ contains the additional energy contributions to make Equations 5.7 and 5.8 equal. $E_{XC}[\rho]$ is called the exchange-correlation energy functional, containing an exchange energy contribution and a correlation contribution to the kinetic and potential energy of the electrons. This functional must again be approximated and the further methods of DFT deal with approaches on how to do that with the best accuracy.

A convenient approach to approximate $E_{XC}[\rho]$ is to define the exchange and correlation energy per electron $\varepsilon_{XC}(\vec{r})$:

$$E_{XC}[\rho] = \int d\vec{r}\, \rho(\vec{r})\varepsilon_{XC}(\vec{r}) \; . \quad (5.9)$$

Perdew, Burke and Ernzerhof used the generalized gradient approximation (GGA) approach for their functional, which is suitable for dealing with rapidly changing electron densities. With GGA, one treats the exchange and correlation energy per electron as a function of the electron density as well as its gradient:

$$E_{XC}^{GGA}[\rho] = \int d\vec{r}\, \rho(\vec{r})\varepsilon_{XC}^{GGA}(\rho(\vec{r}), \nabla\rho(\vec{r})) \; . \quad (5.10)$$

The result of their work is the simple but effective PBE functional [8]. By adjusting empirical coefficients, Zhang and Yang created the revised revPBE functional, which improved the atomic total energies and molecule atomization energies when applied to calculations [102]. Hammer, Hansen and Nørskov found that the revPBE functional also improved the chemisorption energetics of atoms and molecules on transition-metal surfaces and reformulated the PBE functional into the RPBE functional, which yields the same improvements [9]. By adding D3 dispersion corrections to the RPBE functional [10], which are crucial to accurately model water displaying van der Waals forces [11], one finally arrives at the RPBE-D3 functional.

5.1.2. Simple point charge models

The original simple point charge (SPC) model is an empirical effective pair potential devised for liquid water, which was first published in 1981 [103]. In this model, each water molecule consists of three interaction sites coinciding with the centers of the two hydrogen atoms and the oxygen atom. The hydrogen atoms each possess a partial charge q_H and the oxygen atom a partial charge q_O, with $q_O = -2q_H$ to ensure charge neutrality. In addition to Coulomb interactions between all sites of different molecules, a Lennard-Jones potential is applied to oxygen atoms, leading to the effective intermolecular potential between molecules a and b

$$V_{ab} = 4\epsilon \left[\left(\frac{\sigma}{r_{OO}} \right)^{12} - \left(\frac{\sigma}{r_{OO}} \right)^{6} \right] + \sum_{i\,\mathrm{on}\,a} \sum_{j\,\mathrm{on}\,b} k_e \frac{q_i q_j}{r_{ij}} \ . \tag{5.11}$$

Here, ϵ and σ are parameters of the Lennard-Jones potential and r_{OO} is the distance between the oxygen atoms of molecule a and b. In the Coulomb term, $k_e = \frac{1}{4\pi\epsilon_0}$ is the Coulomb constant with ϵ_0 being the electric constant, q_i is the partial charge of atom i of molecule a, q_j is the partial charge of atom j of molecule b and r_{ij} is the distance between these two atoms i and j. The first sum runs over all atoms of molecule a and the second sum over all atoms of molecule b. Furthermore, the SPC model fixes the O-H bond length r_{OH} and the H-O-H bond angle θ_{HOH} as constant values, creating a rigid molecule with no intramolecular potential energy. The geometry of an SPC water molecule is sketched in Figure 5.1.

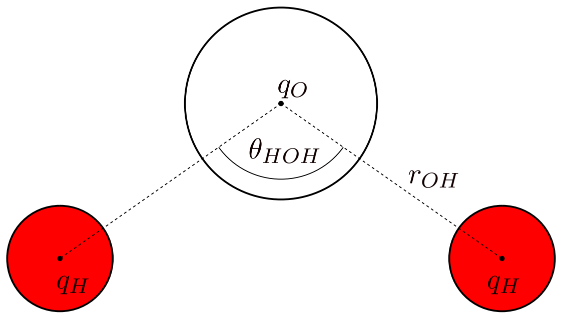


Figure 5.1.: Geometry of an SPC water molecule.

A derivative of the SPC model is the flexible SPC/F model [97]. Here, r_{OH} and θ_{HOH} are not fixed, but rather the equilibrium values of the bond lengths and angles. The now flexible bonds and angles are driven to their equilibrium value via harmonic potentials. This gives rise to the intramolecular potential energy

$$V_{intra} = \sum_{i}^{N} \frac{k_r}{2} \left[\left(r_{OH_{i,1}} - r_{OH} \right)^2 + \left(r_{OH_{i,2}} - r_{OH} \right)^2 \right] + \frac{k_\theta}{2} \left(\theta_{HOH_i} - \theta_{HOH} \right)^2 \ . \tag{5.12}$$

Here, k_r and k_θ are the spring constants of the harmonic potentials, $r_{OH_{i,1}}$ is the distance between the oxygen atom and the first hydrogen atom of molecule i, $r_{OH_{i,2}}$ is the distance between the oxygen atom and the second hydrogen atom of molecule i, and similarly θ_{HOH_i} is the molecular angle of molecule i. The summation runs over all N molecules of the system. Defining now the total intermolecular energy as

$$V_{inter} = \sum_{a<b}^{N} V_{ab} \ , \tag{5.13}$$

the total potential energy of the SPC/F model is given by

$$V_{tot} = V_{inter} + V_{intra} \ . \tag{5.14}$$

Table 5.1 shows the values of all parameters needed for the SPC/F model.

SPC/F parameter	Value
σ [Å]	3.145
ϵ [kcal/mol]	0.1615
q_O [e]	-0.78
q_H [e]	0.39
$\frac{k_r}{2}$ [kcal/mol Å^2]	554.1347992
r_{OH} [Å]	1
$\frac{k_\theta}{2}$ [kcal/mol rad^2]	45.76959847
θ_{HOH} [°]	109.47

Table 5.1.: Parameters of the SPC/F model [104]. Values are given in units corresponding to the LAMMPS `real` units style. Note that LAMMPS uses the thermochemical calorie so that 1 kcal = 4184 J [105].

5.2. Training process

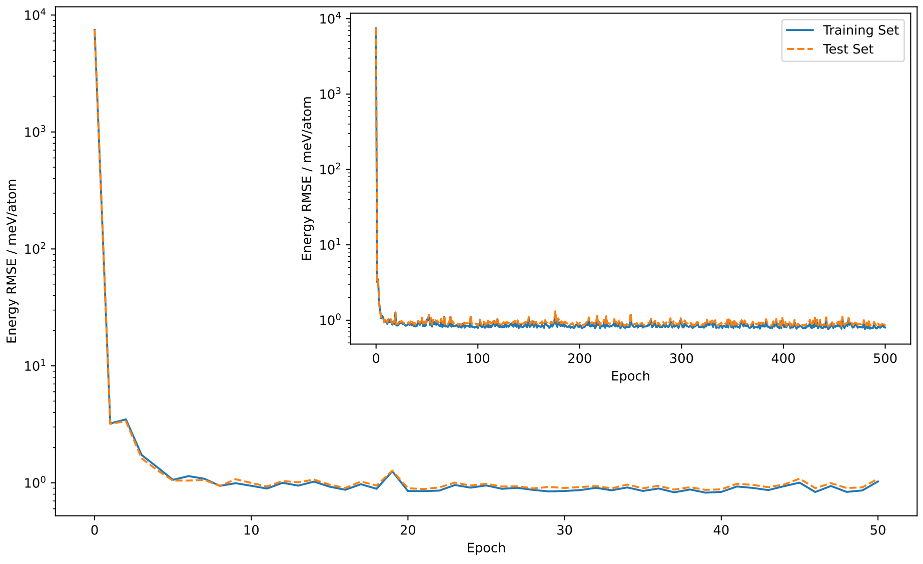
The training of the HDNNP was performed on the Vienna Scientific Cluster (system VSC4) using the n2p2 package and its implementation of the MS-EKF. All SFs use $f_{c,2}$ from Equation 2.15 as their cutoff function, and the radial and angular SFs are of the form G_i^2 from Equation 2.17 and G_i^3 from Equation 2.19, respectively. The symmetry function parameters agree with the ones used in Reference [11] and can be found in full in Appendix A. The NNP consists of 2 hidden layers with 25 nodes each. The activation function for the hidden layers is $f_t(x)$ from Equation 2.7, while the output layer uses the identity function $f_l(x) = x$. The initial weights of the network were chosen randomly from a uniform distribution on the interval $[-1, 1]$.

The training ran for 500 training epochs, though much less were actually needed, as adequate root mean square errors (RMSE) of predicted energies and forces were already reached even before epoch 50. 10% of the reference structures were kept as the testing data set and the hyperparameter β controlling the relative weight of force updates with respect to energy updates was chosen as 10. The update candidates were chosen randomly with all energies and 0.41% of forces being used per update. Energy update candidates were set to always be accepted, while force update candidates had to exceed the RMSE of the previous epoch's RMSE. If they did not, a different candidate was tested, and if no appropriate candidate was found after 3 trials, the candidate with the highest RMSE was chosen. The MS-EKF used 32 streams, as numbers as little as 4 or as high as 48 seem to lose the benefits of the multistream scheme [18]. The hyperparameters for the Kalman filter are $\varepsilon = 0.01$ for the initialization of the error covariance matrix, $q_0 = 0.01$,

$q_{min} = 10^{-6}$ and $\tau_q = 2.302$ for the addition of artificial process noise, and $\eta = 0.01$ for the learning rate.

5.3. Training results

Figure 5.2 shows the learning curves of the training process for both energies and forces. The final NNP, which will be used in the further simulations, yielded an RMSE of 0.76 meV/atom for the training energies and an RMSE of 38.36 meV/Å for the training forces. The force components computed by the NNP during training are further analyzed and compared to the reference forces in Figures 5.3 and 5.4: The former shows around 100000 randomly chosen force components calculated by the NNP as a function of the respective reference forces they were trained with, while the latter shows the distribution of how much the force components deviate from their reference values as a function of these reference values.



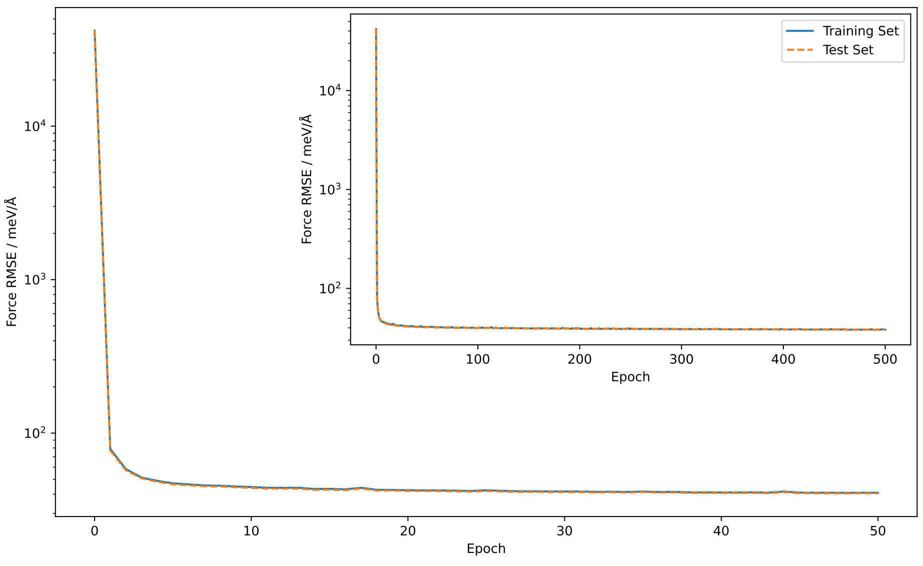


Figure 5.2.: Top: Learning curve in terms of energy RMSEs for the first 50 training epochs. Bottom: Learning curve in terms of forces RMSEs for the first 50 training epochs. The insets show the respective learning curve for the full 500 training epochs.

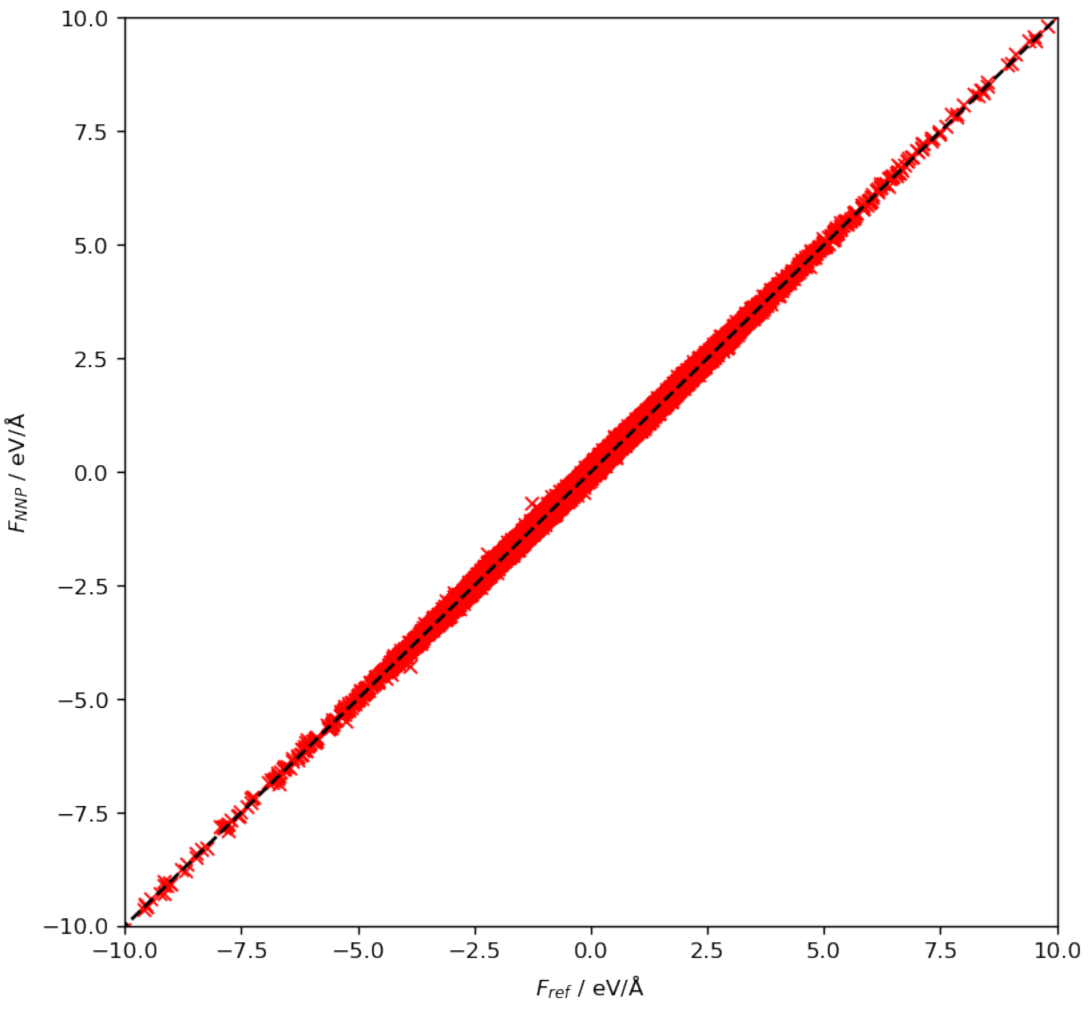


Figure 5.3.: Predictions of force components computed by the NNP compared to the force reference values. The dashed line signifies where the forces are identical, i.e. $F_{ref} = F_{NNP}$.

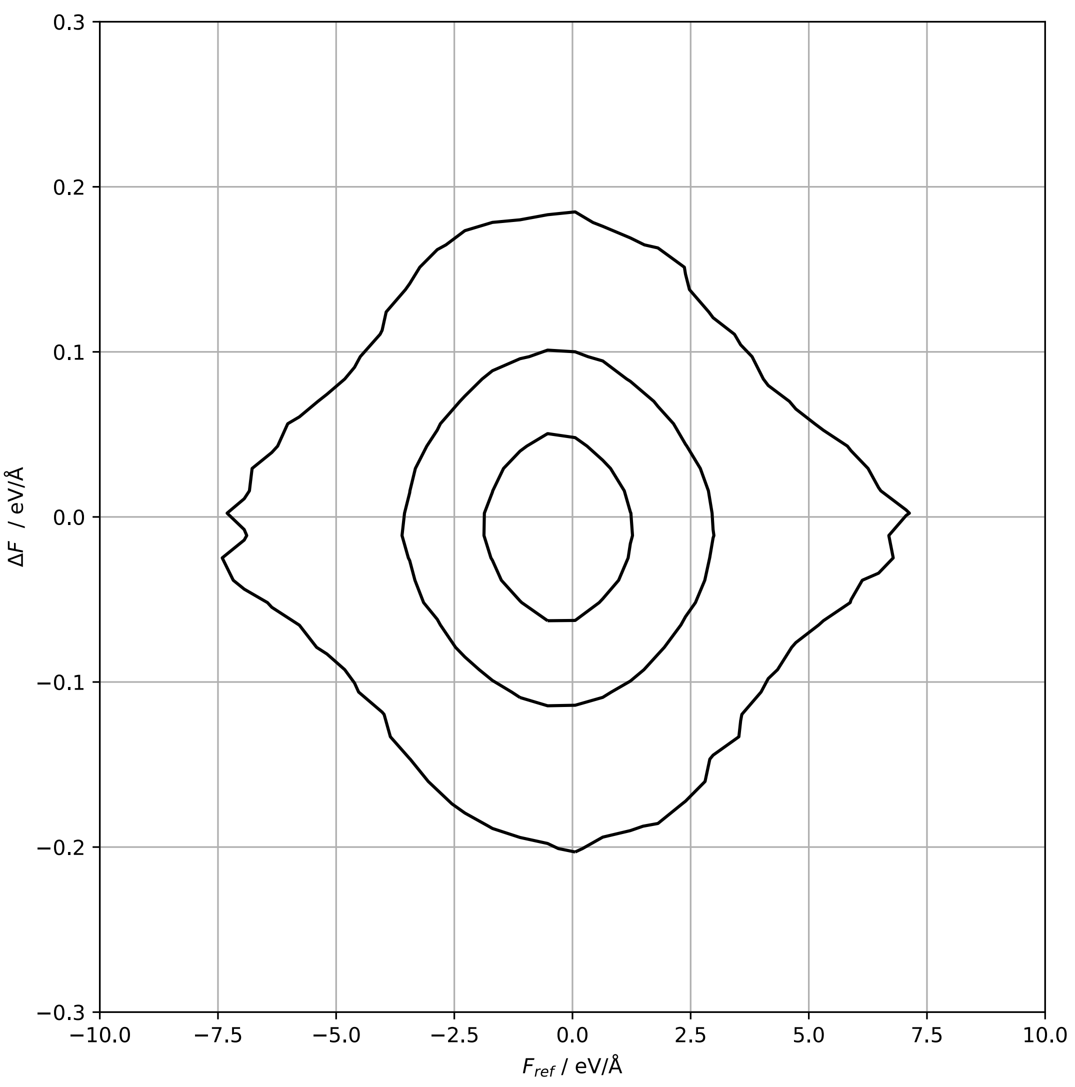


Figure 5.4.: Histogram of the difference ΔF between the force components obtained by the neural network during training and the reference force values, as a function of the reference forces. The innermost contour encloses the area where roughly 68% of the data points lie, the intermediate contour encloses 95% and the outermost contour 99.7%.

6. Simulations

To investigate the water/vapor interface, all simulations were performed using a slab geometry in the NVT ensemble. The simulation box of size $l_x \times l_y \times l_z = 25 \times 25 \times 80\,\text{Å}^3$ used periodic boundary conditions and contained 512 water molecules. Trajectories were created for the 7 different temperatures 300 K, 350 K, 400 K, 450 K, 500 K, 550 K and 600 K, with additional trajectories created later on at 440 K and 620 K. For each temperature, the SPC/F empirical model was used to perform an initial equilibration phase (Section 6.1.1), with a second equilibration phase performed afterwards using the NNP detailed in the previous chapter (Section 6.1.2). The resulting configurations were then used as the respective starting points for the NMPIMD production runs (Section 6.2). Most computations were done using the Vienna Scientific Cluster (systems VSC-4 and VSC-5) with additional test runs or equilibrations being performed on home computers.

6.1. Equilibration

6.1.1. First equilibration phase

In a first equilibration phase, the system started from a randomized slab configuration of water molecules centered in the middle of the simulation box. The starting configuration consists of 512 water molecules where oxygen atoms are placed equidistantly in a simple cubic crystal structure so that if the simulation box were to be reduced to a cube with 25 Å side length, the density of the system roughly corresponds to that of liquid water. The first hydrogen atom of each molecule was placed in a random direction at a distance of 1 Å to the oxygen atom while the second hydrogen atom was placed at the same distance in a random direction that fulfills the condition of creating a molecular angle of 109.47°.

Simulations were performed in the NVT ensemble with LAMMPS, using the velocity-Verlet algorithm for time integration. The system is brought to the target temperature by employing a standard Nosé-Hoover thermostat with a relaxation time of 0.1 ps. All interactions were modelled according to the SPC/F model from Section 5.1.2. The Lennard-Jones potential used a cutoff distance of 9.2 Å while long-range Coulomb forces were evaluated using the particle-particle particle-mesh (PPPM) method [106] with a relative accuracy of 10^{-5}, which means the RMS error of per-atom forces calculated by the long-range solver will be 100000 smaller than a representative reference force determined by LAMMPS [107]. The timestep of integration was chosen to be 0.5 fs. After 0.5 ns of simulation time, the configuration was saved as a starting point for a second equilibration phase. This process was repeated for each target temperature, individually generating an equilibrated configuration each time.

6.1.2. Second equilibration phase: Adding the NNP

The second equilibration phase starts from the equilibrated configuration created using the SPC/F model. The simulation was again run by LAMMPS in the NVT ensemble, using the same thermostat settings and the same integration scheme with a timestep of 0.5 fs. However, the SPC/F model is replaced by the NNP trained in Chapter 5 via the LAMMPS interface provided by n2p2, which now computes the interaction energies and forces between atoms. Note that while the LAMMPS data file for the SPC/F model requires the definitions of atom bonds and angles, the NNP calculations are solely based on the relative positions of the atoms and thus there are formally no bonds and angles defined for the molecules. This also means that in contrast to the SPC/F model, bond breaking and reformation is not prohibited, though it was not observed during the following simulations.

The `nnp` pair style for LAMMPS requires additional keyword values. Firstly, it must be defined how extrapolation warnings are handled. The `maxew` keyword determines how many extrapolation warnings may occur before the simulation is aborted, and was set to a sufficiently high number (10^9) to always continue the run. The `showewsum` and `resetew` keywords were set to 1000 and `yes` respectively, meaning the sum of accumulated extrapolation warnings for each 1000 steps are displayed in the output. Then, the `cflength` and `cfenergy` keywords set the conversion factors if the units used in the NNP training differ from the units used by LAMMPS. Since the training used the Bohr radius and Hartree energy, these were converted to Ångström and kcal/mol used by the LAMMPS real unit style by setting `cflength` to 1.8897261 and `cfenergy` to $1.5936014 \cdot 10^{-3}$. Lastly, the cutoff radius of the pair style is set to 6.4 Å, which is slightly larger than the cutoff radius of the symmetry functions of the NNP (6.35013 Å), as advised by the n2p2 documentation [85] to account for potential rounding errors, as the cutoff of the pair style must not be smaller than the cutoff of the NNP.

This second equilibration phase ran for 0.5 ns, at which point no energy drift could be observed. The process was of course carried out for each temperature and a configuration saved at the end of the 0.5 ns run time as the starting point for the production runs each time.

6.2. Production runs

6.2.1. NMPIMD simulation

To acquire trajectories that include nuclear quantum effects, NMPIMD simulations were performed using the i-PI package. Multiple instances of LAMMPS were connected to i-PI to deal with force and energy evaluations, while i-PI covered time integration. LAMMPS again relied on n2p2 and the NNP trained in Chapter 5, in the same way the calculations for the equilibration in Section 6.1.2 were done. The equilibrated configurations from said section were used as a starting point for the production runs. The number of replicas used was $P = 32$. As the chain frequency

$$\omega_P = \frac{\sqrt{P}}{\beta\hbar} = \frac{\sqrt{P}}{\hbar} k_B T \tag{6.1}$$

scales with the temperature T, the time step was adjusted as higher temperatures were sampled, to accommodate the increased frequencies. The run at 300 K used a time step of 0.4 fs, the ones from 350 K to 500 K used 0.25 fs, and upwards of 550 K used 0.2 fs. Simulations were performed in the NVT ensemble by employing a PILE-G thermostat with friction and scaling coefficients $\tau = 50$ fs and $\lambda = 0.5$, respectively. Each trajectory had a simulation time of 5 ns, during which the configurational data as well as temperature-, energy- and pressure quantities were dumped every 0.5 ps. The dumps from the first 0.5 ns were discarded to allow additional equilibration of the now fully quantum system, and no energy drift was observed after this time window. The number of extrapolation warnings was summed and dumped every 1000 steps and is shown for each temperature in Table 6.1.

T [K]	PIMD	Classical
300	1.3	0
350	2.1	$1.5 \cdot 10^{-6}$
400	3.6	$2.9 \cdot 10^{-5}$
440	5.9	$2.6 \cdot 10^{-4}$
450	6.6	$6.6 \cdot 10^{-4}$
500	11.2	$5.3 \cdot 10^{-3}$
550	16.6	$1.8 \cdot 10^{-2}$
600	22.2	$3.3 \cdot 10^{-2}$
620	23.9	$3.9 \cdot 10^{-2}$

Table 6.1.: Number of extrapolation warnings per time step given by n2p2 during the production runs.

During the runtime of the 450 K simulation, technical problems occurred at the Vienna Scientific Cluster. This led to the crash of the simulation and, due to a change in infrastructure, the need to re-setup in a slightly different way regarding the number of processors, how jobs were distributed among them, versions of packages used etc. While the simulation could be recovered where it stopped and restarted at that point, it may have lead to inconsistencies, which will be discussed later. To bolster the available data, the trajectory at 440 K was created. The trajectory at 620 K was created later on to bolster the data available for the phase diagram and critical temperature calculations.

6.2.2. Non-PIMD simulation

To gather a basis of comparison, trajectories of each temperature were also generated without the use of NMPIMD. These classical[1] trajectories were created in the same way and with the same settings as the second equilibration phase detailed in Section 6.1.2. The simulation was run for 5 ns and the quantities of interest as well as the configurations were acquired with the same frequency as the NMPIMD production runs, dumping data every 0.5 ps. The number of extrapolation warnings can be found in Table 6.1.

[1] "Classical" may be seen as a slight misnomer, as the underlying force calculations are based on the very quantum mechanical density functional theory. By not using PIMD, one only excludes nuclear quantum effects, not all quantum mechanical aspects. However when speaking of "classical" simulations in this context here, only the exclusion of quantum nuclear effects is meant.

7. Observables

After generating the trajectories, the obtained data must be processed to extract the properties of interest. This chapter discusses the observables that can be gleaned from the available data and how they are calculated to yield the structural and thermodynamic quantities presented later on.

7.1. Bond lengths, angles and orientations

When looking at the resulting configurations yielded by a molecular dynamics simulation, the simplest observable one can analyze are the distances and angles between atoms. As in this NNP approach, the oxygen atom and hydrogen atoms that make up a water molecule are in no way assigned to each other in a predetermined way but rather stay as a molecule through the assumptions made by the NNP, the atoms of each configuration must first be gathered as molecules. In this case, this was done by using the k-d-tree-algorithm implemented in the open source Python library Scipy [108], assigning the nearest two hydrogen atoms to each oxygen atom. The OH bond lengths are then trivially given by the distance between the oxygen atom and each of the two associated hydrogen atoms and the HOH molecular angle is given by calculating the angle between these two bonds. From this, distributions of the OH bond lengths and HOH molecular angles can easily be created. As an intermediate step, the surface layers of the liquid slab can be identified before calculating the lengths and angles, so that one can compare their distributions for molecules of a certain surface layer, the bulk or simply the whole system (see Figure 7.1).

7.1.1. Distribution of the angle between bonds and the surface normal

Examining the angle between bonds and the surface normal can shed light on the differences of preferred orientations of the molecules whether they are located in the bulk of the slab or near the interface. To investigate this behaviour, the angle between the OH-bond (i.e. the vector pointing from an oxygen atom to one of its hydrogen atoms) and the z-axis shall be defined as ψ (see Figure 7.2). This assumes that the molecule is located closer to the upper surface of the liquid slab; for molecules closer to the lower surface, the negative z-axis is used instead. Note that for each water molecule, two angles ψ are calculated, one for each OH-bond. The configuration is also prepared for each frame by shifting all atoms along the z-axis by the same amount so that the center of mass of the liquid slab has the same z-coordinate for each frame. The values of each $\cos(\psi)$ and the z-coordinate of the oxygen atom the bond belongs to can then be paired and gathered in a 2-dimensional histogram.

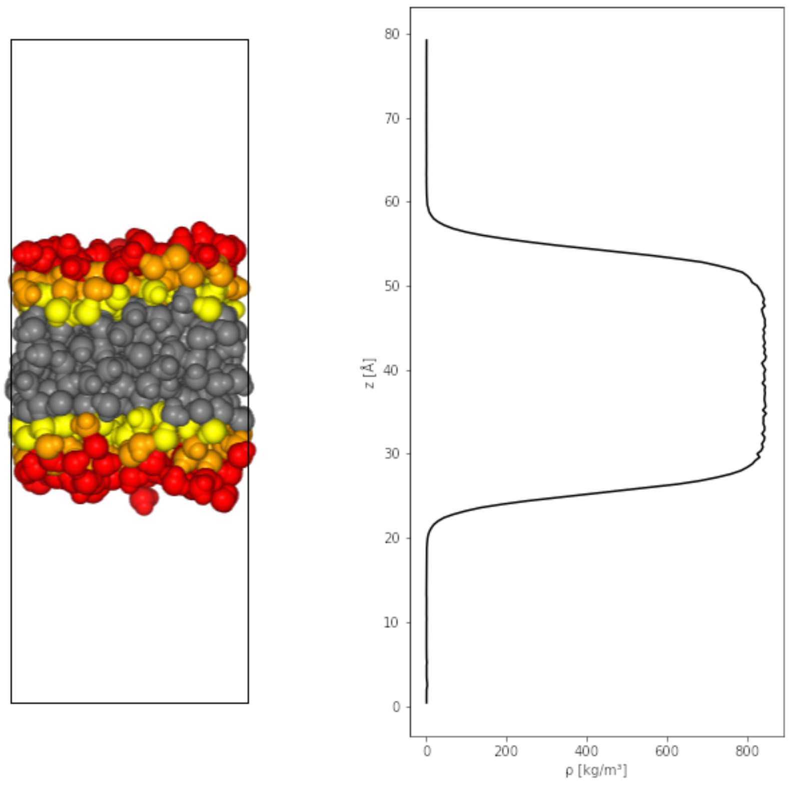


Figure 7.1.: Left: Orthogonal view of the water slab. The black rectangle symbolizes the simulation box. The red, orange and yellow molecules are those which were identified as the first, second and third surface layers, respectively. The grey water molecules are defined as the bulk. Right: Example of a density profile, corresponding to the sketched system on the left.

7.1.2. Bivariate angle distribution

To further investigate the orientation of the surface molecules, the following angles are defined: Let $\vec{N}$ be the unit surface normal vector of the interface between liquid water and vapor (pointing into the vapor phase) and $\vec{d}$ be the unit vector in the direction of the water dipole. Then the angle θ is defined through $\vec{N} \cdot \vec{d} = \cos(\theta)$. Further, the angle ϕ shall be the dihedral angle between the molecular plane and the plane defined by $\vec{N}$ and $\vec{d}$ [109]. In other words, ϕ is the angle which the molecule must be rotated around its dipole moment to have the molecular plane perpendicular to the surface. Figure 7.3 depicts how θ and ϕ are defined. Gathering the tuples of $(\cos(\theta), \phi)$ in a 2-dimensional histogram gives a bivariate distribution of these measures of orientation, which can be further refined by only looking at the angles of the molecules of certain surface layers.

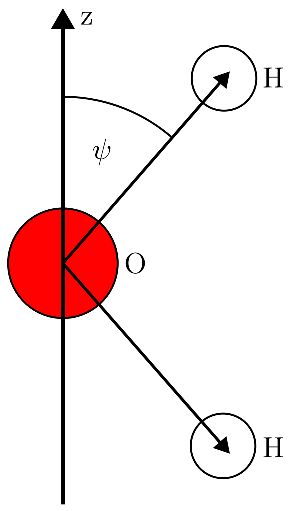


Figure 7.2.: Definition of ψ as the angle between a OH-bond and the z-axis.

7.2. Density profile

The density profile is the mass density of the system along some spatial dimension. As the system is configured to have the water/vapor interface perpendicular to the z-axis, the density will be calculated as a function of the position on the z-axis. For this, the simulation box is divided into k identical slices of thickness Δz so that

$$\Delta z \cdot k = l_z \ . \tag{7.1}$$

Thus, each slice signifies the interval $I_i = [\Delta z \cdot (i-1), \Delta z \cdot i)$ with $i \in \{1, 2, ..., k\}$. Let us now define the number of H- and O-atoms in slice i as $n_{H,i}$ and $n_{O,i}$. Then, the density of each slice can be written as

$$\rho_i = \frac{n_{H,i} \cdot m_H + n_{O,i} \cdot m_O}{l_x l_y \Delta z} \tag{7.2}$$

with m_H and m_O being the atomic masses of hydrogen and oxygen respectively, and l_x and l_y being the size of the simulation box in x- and y-direction. For each frame that these densities are calculated for, the atom positions are once again corrected in a way so that the center of mass is at the same position for each frame. Atoms that move beyond the periodic boundaries reenter the system on the opposite side and are considered in their new position for the purpose of calculating the center of mass. In the end, the density of each slice is averaged over all time frames considered. An example of such a density profile is sketched in Figure 7.1.

The resulting profiles yield the densities of the coexisting liquid and vapor phase, ρ_L and ρ_V, either by taking the average of the densities in the center region of the liquid and the vapor phase, or by fitting the equation

$$\rho(z) = \frac{1}{2}(\rho_L + \rho_V) - \frac{1}{2}(\rho_L - \rho_V) \tanh\left(\frac{z - z_0}{d}\right) \tag{7.3}$$

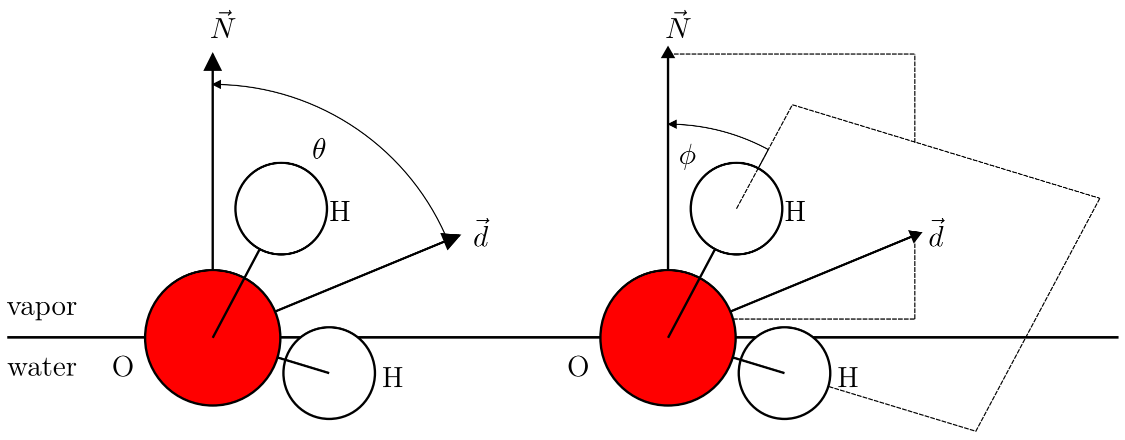


Figure 7.3.: Definition of θ as the angle between surface normal vector $\vec{N}$ and direction of water dipole $\vec{d}$ (left) and ϕ as the dihedral angle between the molecular plane and the plane defined by $\vec{N}$ and $\vec{d}$ (right).

to the profile data, where z_0 is the location of the middle of the interface and d is the thickness of the interface [110]. However, as the real profile does not exactly follow this sigmoidal fit, it may introduce a bias to the estimate of the bulk densities.

7.2.1. Intrinsic density profile

While the above profile shows the density by correlating molecular positions to the center of mass, the intrinsic mass profile [111] correlates molecular positions with the local position of the interface. If the z-coordinate of the rugged interface at each coordinate (x, y) is written as $\zeta(x, y)$, then the intrinsic mass density profile is given by

$$\rho_I(z') = \frac{1}{l_x l_y} \left\langle \sum_i^N m_i \delta(z' - z_i + \zeta(x_i, y_i)) \right\rangle . \tag{7.4}$$

Here, the angular brackets denote again the average over all time frames considered and the sum is taken over all N atoms, with m_i being the mass and x_i, y_i and z_i being the coordinates of the ith atom. By convention, $z' = 0$ signifies the surface layer, with positive values of z' being located in the vapor phase. The delta peak arising from the surface molecules located at $z' = 0$ shall be omitted in graphical representations.

7.3. Critical point

The liquid/vapor critical point of water is the point in the phase diagram where both liquid and vapor phase coexist at the critical temperature T_C and critical pressure p_C, or more relevantly, the same critical density ρ_C [112]. Once the densities ρ_L and ρ_V of the liquid and vapor phase at different temperatures T are known, the location of the critical point can be found by extrapolation using a fit to the expression proposed by Wilding [113]

$$\rho(T) = \rho_C + a|T - T_C| \pm b(T - T_C)^{\beta} \ , \tag{7.5}$$

where a, b and β are fitting parameters. The plus-minus sign implies the usage of the plus sign for the liquid branch and the minus sign for the vapor branch.

7.4. Surface tension

The surface tension γ is the ratio of the change in energy of a liquid W to the change of surface area of the liquid ΔA [112], i.e.

$$W = \gamma \cdot \Delta A \ . \tag{7.6}$$

To calculate γ of the planar liquid/vapor interface the formula

$$\gamma = \frac{l_z}{2} \left\langle P_{zz} - \frac{P_{xx} + P_{yy}}{2} \right\rangle \tag{7.7}$$

is used [114, 115]. Here l_z is the length of the simulation box in the direction of the interface normal and P_{ij} is the ijth element of the pressure tensor. Once acquired, the surface tensions of the system at different temperatures T can be fitted to the interpolation equation proposed by Vargaftik, Volkov and Voljak [116]

$$\gamma = A \left(1 - \frac{T}{T_C}\right)^{\mu} \left[1 + B\left(1 - \frac{T}{T_C}\right)\right] \ , \tag{7.8}$$

where T_C is the critical temperature of the system and A, B and μ are fitting parameters.

8. Results

This chapter presents the results for the structural properties (Section 8.1) and the thermodynamic properties (Section 8.2). By comparing the results from the PIMD simulations including NQEs and the classical simulations where NQEs are absent, the impact of these effects can be directly investigated.

8.1. Structural Properties

Each trajectory, both PIMD and classical, yielded 9000 configurations. In a first step, Pytim was used to identify the surface layers of the liquid phase for each configuration, where ITIM was set to use a probe sphere radius of 2 Å. This created four sets of molecules: The first surface layer, the second surface layer, and the bulk, defined as all molecules of the liquid phase that don't belong to the surface layers, as well as the fourth set containing

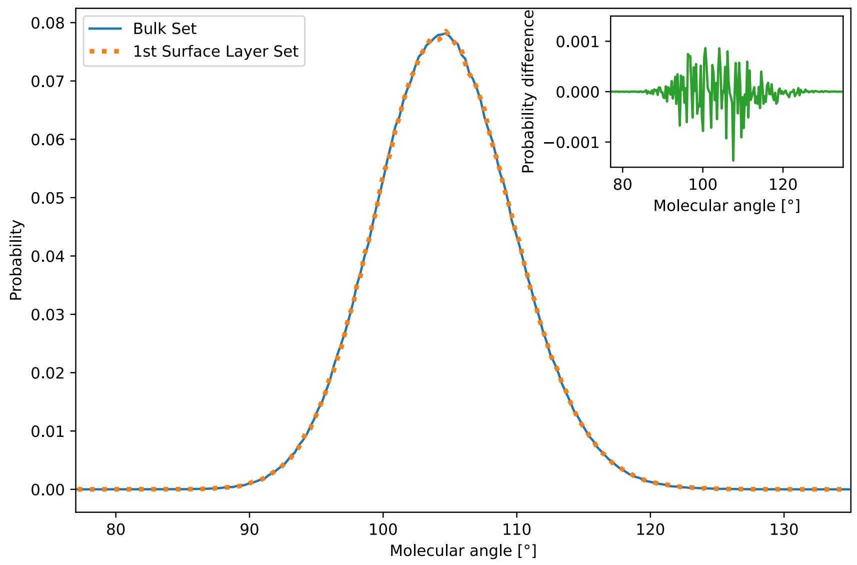


Figure 8.1.: Molecular angle distribution in the first surface layer and the bulk for the PIMD data set at 300 K. The inset shows the difference between the surface and bulk distributions.

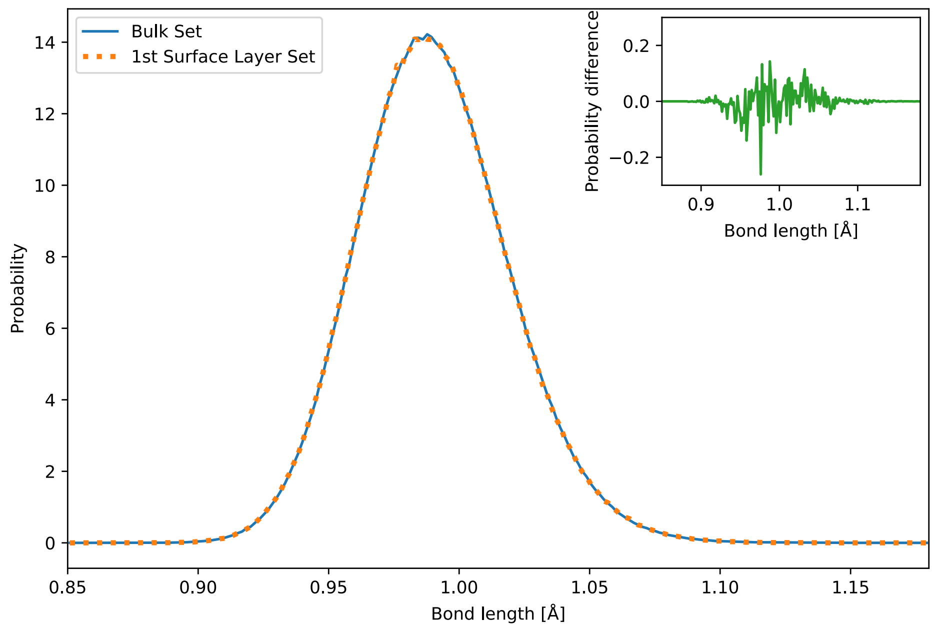


Figure 8.2.: Bond length distribution in the first surface layer and the bulk for the PIMD data set at 300 K. The inset shows the difference between the surface and bulk distributions.

all molecules of the configuration. The length of the O-H bonds and the molecular angles were calculated for each set along all configurations and gathered in histograms, which take the form of slightly right skewed normal distributions. No significant difference can be seen in the distributions of the different sets, apart from statistical noise (see Figures 8.1 and 8.2). Figure 8.3 shows the molecular angle and bond length distributions of the 14 original trajectories, as well as the mean values of those quantities for their respective datasets. All graphs show a similar trend where the classical distributions have a more defined peak while the PIMD distributions are slightly broadened. The bond length distributions experience a clear right shift for the quantum case, which is consistent with the observation of scattering experiments that NQEs make the O-H-bonds longer [117]. For the molecular angles, the mean values of the distributions reveal that while the angles are initially larger for the quantum case, there is a trend reversal at high temperatures, where the molecular angles become smaller for the PIMD data compared to the classical data. This could indicate how NQEs affect properties differently for high and low temperatures.

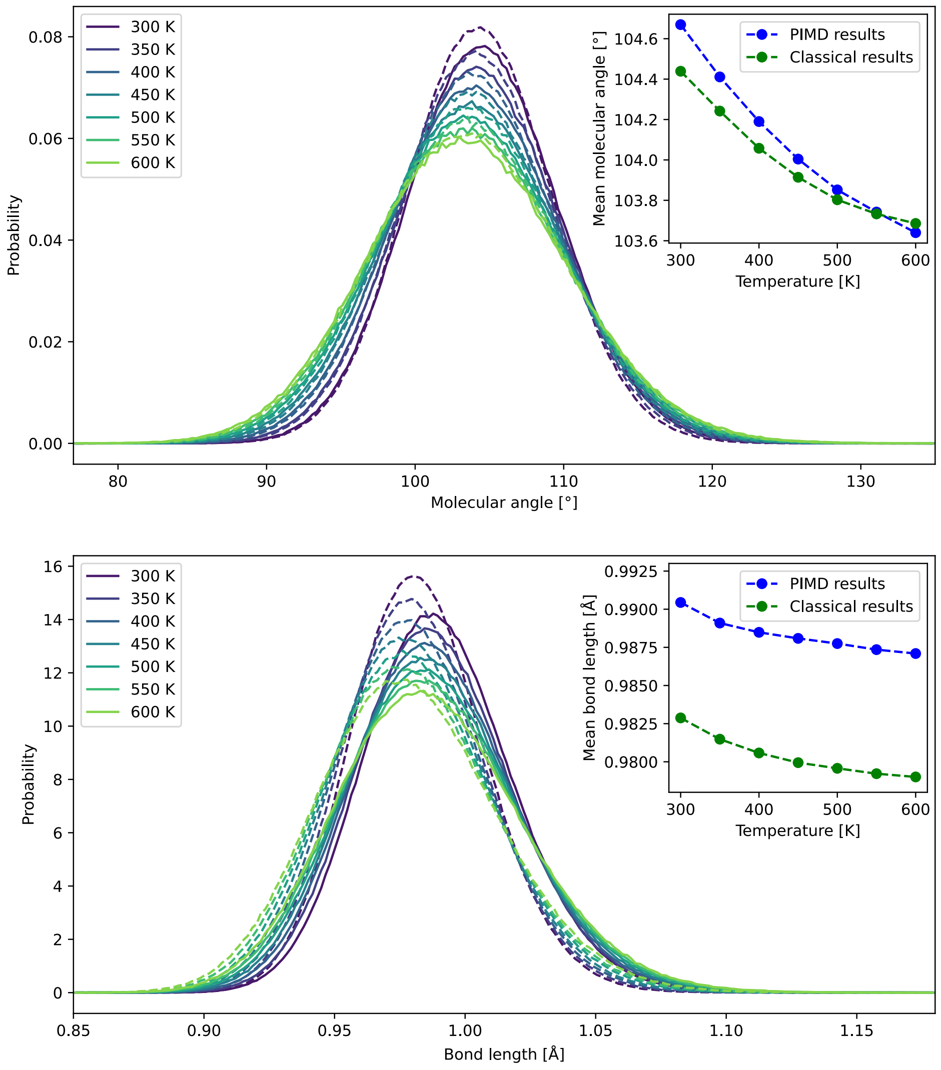


Figure 8.3.: Top: Molecular angle distributions of the bulk data set. Bottom: Bond length distributions of the bulk data set. The solid lines are the distributions gathered from the PIMD trajectories while the dashed lines represent the classically obtained distributions. The insets show the mean values of the distribution for each temperature.

Looking at the orientation of the water molecules, Figure 8.4 shows how the O-H bonds are oriented relative to the z-axis, as defined by the angle ψ in Section 7.1.1, for a few select temperatures. One can see the preferential orientation of water molecules near the surface, where at low temperatures, the O-H bonds tend to point directly inward into the slab or in an acute angle to the water surface outward. In contrast, the probability of finding bonds pointing directly outward are drastically reduced and this trend can also be seen at temperatures as high as 450 K. These trends are present in both the PIMD and the classical case. At even higher temperatures, the structure of the slab is expectedly washed out. The distributions of all original temperatures can be seen in Appendix B. The structure of the first two surface layers is further examined by creating the bivariate distributions of angles θ and ϕ defined in Section 7.1.2. Figure 8.5 shows such distributions for 300 K in terms of the free energy F via the relation $F = -k_B T \ln p$, where p is the probability of finding a molecule with that orientation. In the first layer, one can appreciate the drastically higher free energy for OH-bonds dangling into the vapor phase (i.e. $\cos(\theta) \approx 1$) for the PIMD case compared with the classical results. This means that such spatial orientations are roughly 40% less probable in the PIMD simulations. There is also a center region around $\phi \approx \pi/4$ that is roughly 10% less preferred compared to other values of ϕ or the classical case in general. The second layer also seems to show a slightly higher free energy for low θ values, but this trend quickly disappears for higher temperatures (see Figure 8.6). The behaviour of the first surface layer mentioned above continues to show though, before most of the structure is lost in noise when approaching the critical temperature. Further bivariate distributions can be found in Appendix B.

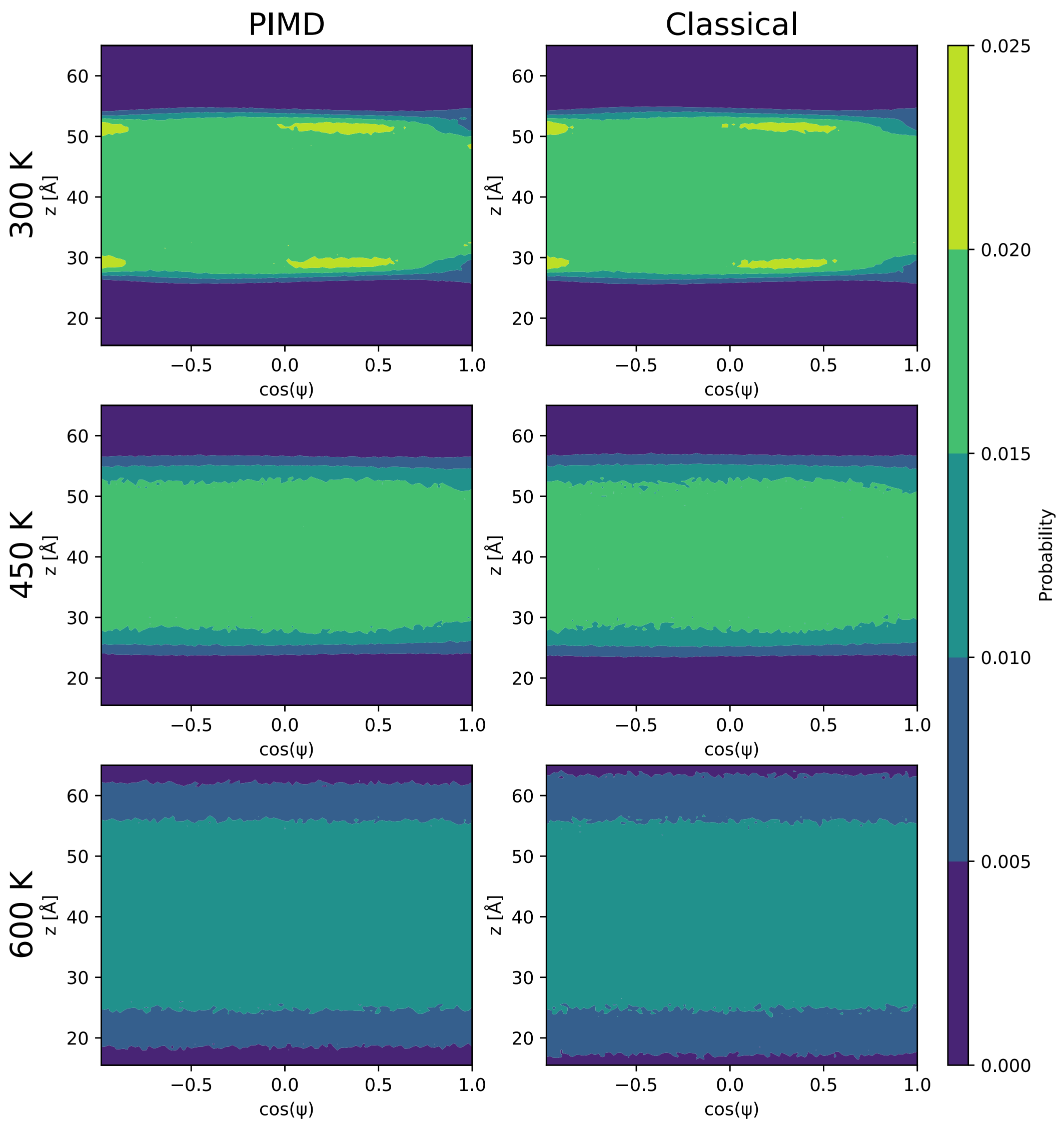


Figure 8.4.: Distribution of the angle ψ between O-H bonds and the z-axis (or negative z-axis for molecules located in the lower half of the slab). The left column shows the distributions gathered from the PIMD data, the right column the classical results. While preferential orientations can be seen at lower temperatures, this trend gradually disappears with increasing temperature.

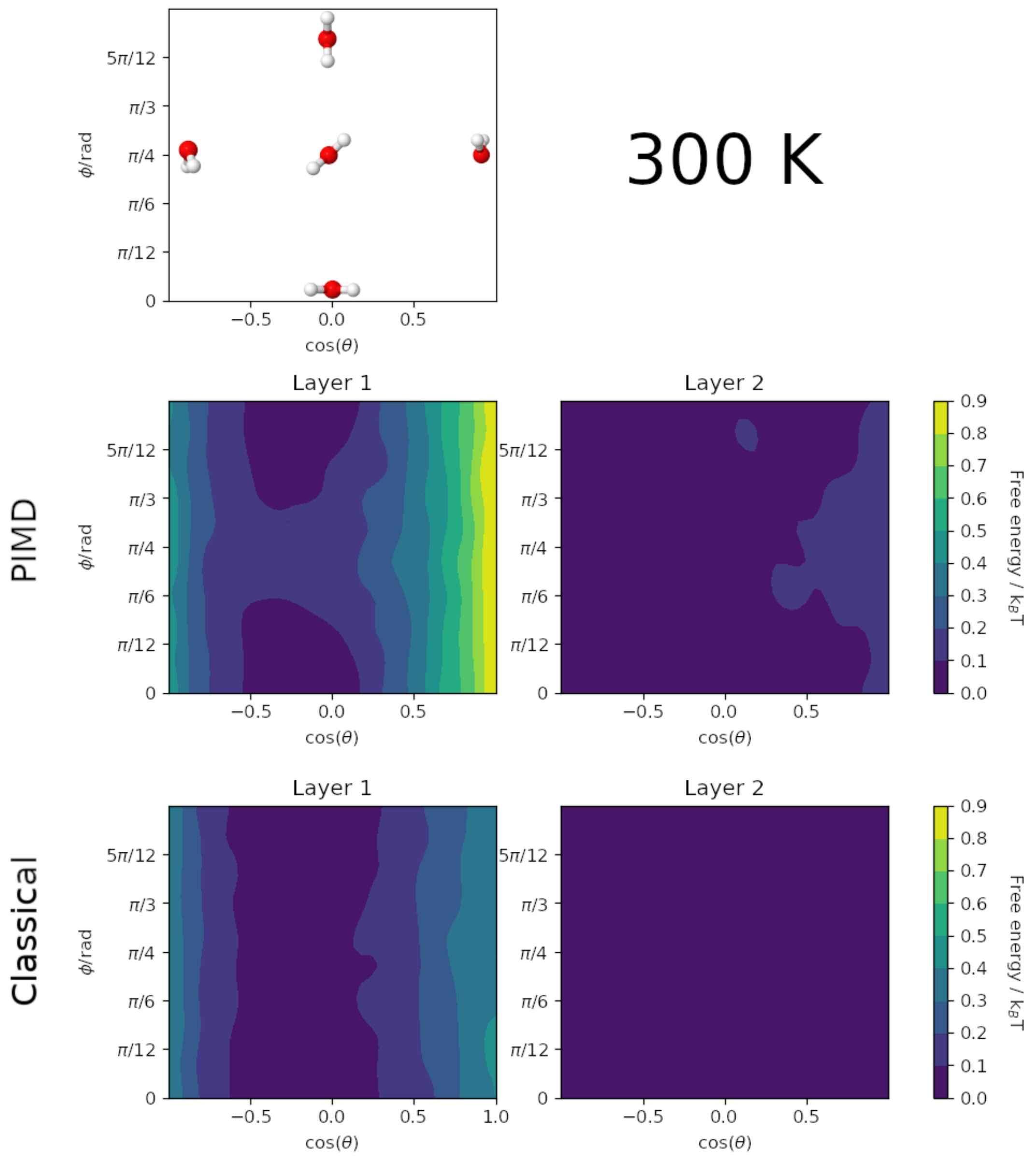


Figure 8.5.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 300 K. The top figure offers a visual guide how the molecule is oriented (with the view direction being parallel to the surface, where the liquid phase is below the molecule and the vapor phase above) in that region of the distribution. The center row shows the distributions gathered from the PIMD data, the bottom row shows the classical case and the first and second column show the first and second surface layer, respectively.

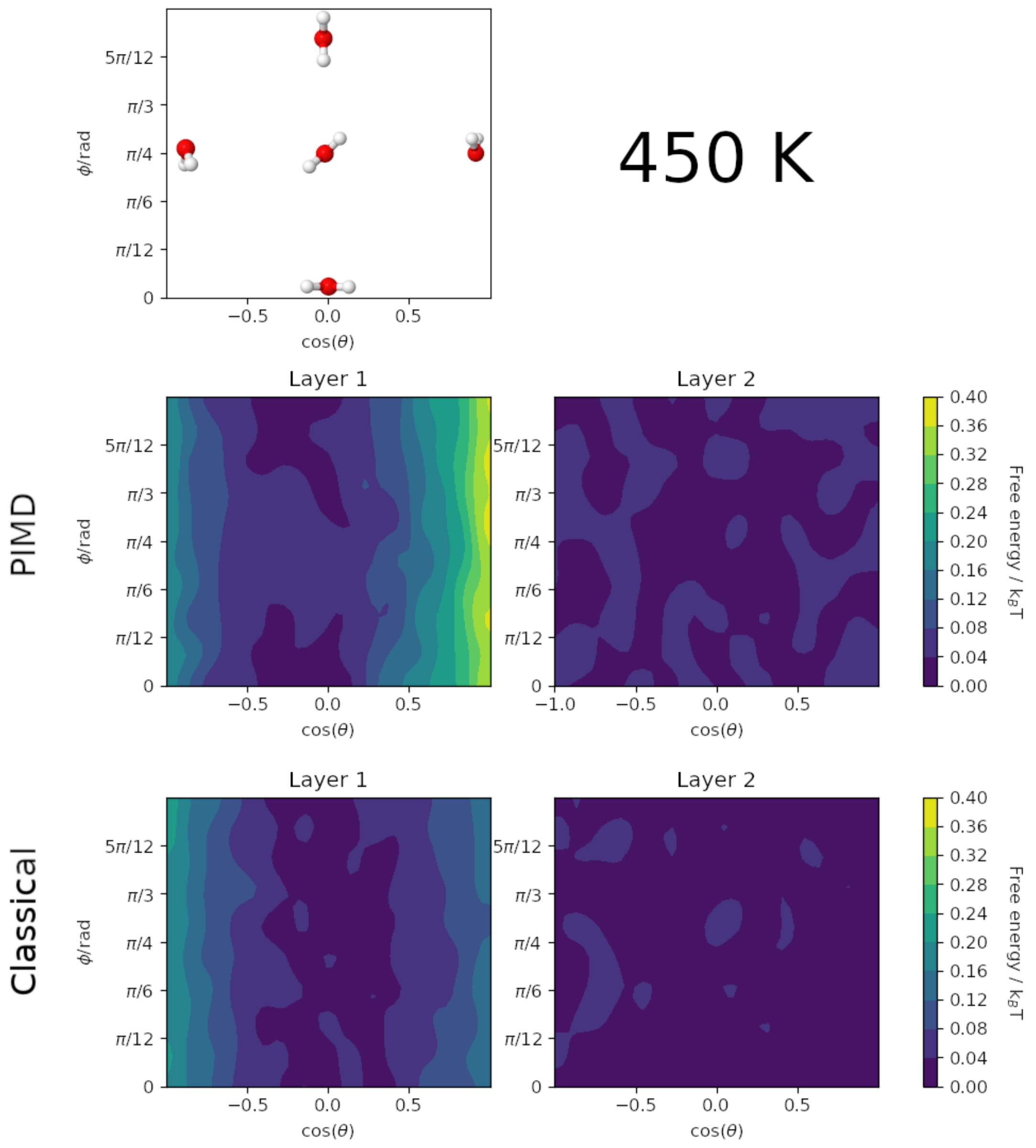


Figure 8.6.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 450 K. The top figure offers a visual guide how the molecule is oriented (with the view direction being parallel to the surface, where the liquid phase is below the molecule and the vapor phase above) in that region of the distribution. The center row shows the distributions gathered from the PIMD data, the bottom row shows the classical case and the first and second column show the first and second surface layer, respectively.

8.2. Thermodynamic Properties

Figure 8.7 reports the mass density profiles for both PIMD and classical trajectories, where the broadening of the water slab with increasing temperature can be seen. Simultaneously, the density of the liquid phase decreases and the density of the vapor phase increases. One can see that the density of the liquid phase is higher when using PIMD compared to without PIMD, which is in accordance with the observations mentioned in Section 3.3.1. The rugged profile at 300 K indicates an over-ordering in the liquid phase that could be the fault of finite size effects, an inconvenience that fortunately disappears for the other higher temperatures. From these profiles, values for the densities of the liquid and vapor phases are extracted by averaging the profiles' values in a 10 Å thick region centered in the middle of the respective phase, except for the trajectories for 600 K and

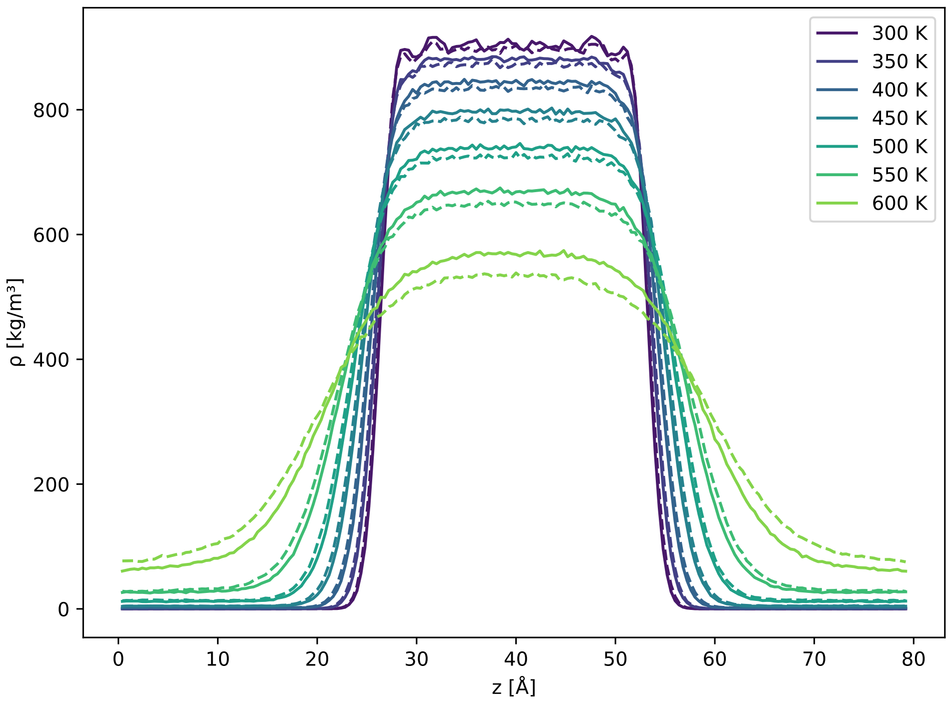


Figure 8.7.: Mass density profiles calculated from the trajectories of different temperatures. For each graph, the center portion of higher density represents the liquid phase while the vapor phase appears as the low density edges. The solid lines are the profiles gathered from the PIMD simulations while the dashed lines represent the classical simulations.

T [K]	$\rho_{L,PIMD}$	$\rho_{L,classical}$	$\rho_{L,OW}$	$\rho_{V,PIMD}$	$\rho_{V,classical}$	$\rho_{V,OW}$
300	904(2)	897(1)	899(1)	0.020(3)	0.031(3)	0.03(1)
350	881(1)	873(1)	876(1)	0.279(8)	0.241(8)	0.20(2)
400	844(1)	834(1)	836(1)	1.47(10)	1.42(3)	1.41(5)
440	807(1)	795(1)	797(1)	3.36(7)	3.68(10)	3.9(2)
450	798(1)	784(1)	788(1)	4.30(7)	4.56(8)	4.7(1)
500	739(1)	724(1)	728(1)	11.8(2)	13.4(2)	12.7(2)
550	669(1)	649(1)	657(1)	26.8(2)	28.8(2)	31.2(3)
600	569(1)	535(1)	550(1)	62.7(4)	77.5(3)	86(1)
620	508(2)	455(1)	509(1)	94(2)	116(3)	132(2)

Table 8.1.: Densities of liquid phase ρ_L and vapor phase ρ_V in kg/m^3, given at different temperatures. The subscripts of the quantities denote whether they stem from the PIMD or classical simulations, and the subscript "OW" denotes the values reported in [17], displayed here for comparison with the classical values.

	T_C[K]	ρ_C[kg/m^3]
PIMD	644(5)	296(5)
classical	629(3)	294(5)
Wohlfahrt 2020	632(2)	310(3)
IAPWS	647.096(100)	322(3)

Table 8.2.: Critical temperatures T_C and critical densities ρ_C resulting from fits. The data from Wohlfahrt [17] is included for comparison with the classical values, as well as the literature value of the critical point by the IAPWS [68].

620 K, in which case the thickness of the center region was chosen as 5 Å. The resulting densities are gathered in Table 8.1, which also shows the higher densities of the PIMD data compared to the classical in the liquid phase, and vice versa in the vapor phase. Also included are the results of the 2020 paper by Oliver Wohlfahrt [17]. In this paper, Wohlfahrt used a similarly trained RPBE3/D3-based NNP to explore the water/vapor interface, however of a larger system with 1024 molecules and without the use of PIMD. Thus the density values reported there lend themselves for comparison with the classical results in this work. While it was attempted to create an NNP comparable to that of Wohlfahrt, the density values drift apart at higher temperatures though. The location of the critical point is extrapolated by fitting Equation 7.5 to the density data of 400 K and higher, creating a coexistence curve. The fit was done by using the `optimize.curve_fit` routine of the open source Python package SciPy [108]. The fit of the classical data yielded the fitting parameters $a = (0.55 \pm 0.03)$ kg/m^{3}K, $b = (86 \pm 7)$ kg/m^{3}K and $\beta = 0.29 \pm 0.02$. The parameters for the PIMD data came out as $a = (0.53 \pm 0.02)$ kg/m^{3}K, $b = (83 \pm 7)$ kg/m^{3}K and $\beta = 0.30 \pm 0.02$. The critical point locations are reported in Table 8.2. The resulting fits are shown in Figure 8.8, along with the IAPWS95 equation of state [68], the data from [17] as well as from the empirical TIP4P/2005 model [110]. One

can see how the PIMD coexistence curve inches upwards towards the IAPWS95 equation of state compared to the classical results. However, while the critical temperature is closer to the literature value, the critical density of both the PIMD and classical data seems to be underestimated. Moreover, the liquid branch of the coexistence curves of all NNP calculations displayed here are much further off from the IAPWS line, whereas empirical models show better agreement.

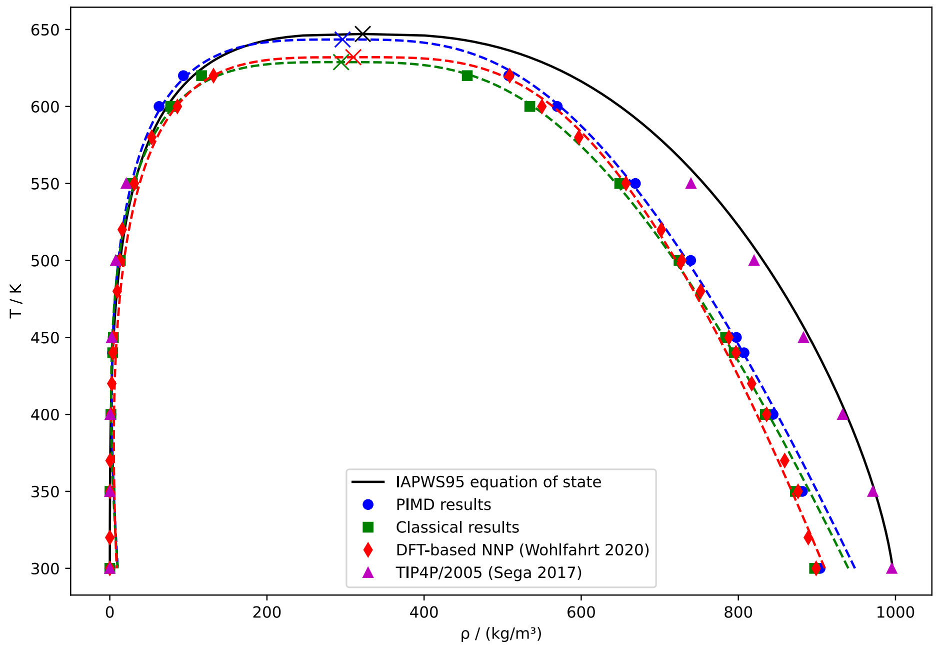


Figure 8.8.: Coexistence curves. The symbols represent the measured densities, the dashed lines are the respective fits and the crosses mark the critical point resulting from the fit. Also included is the IAPWS95 equation of state (solid line) [68], the data from Wohlfahrt [17] as well as from the empirical TIP4P/2005 model [110].

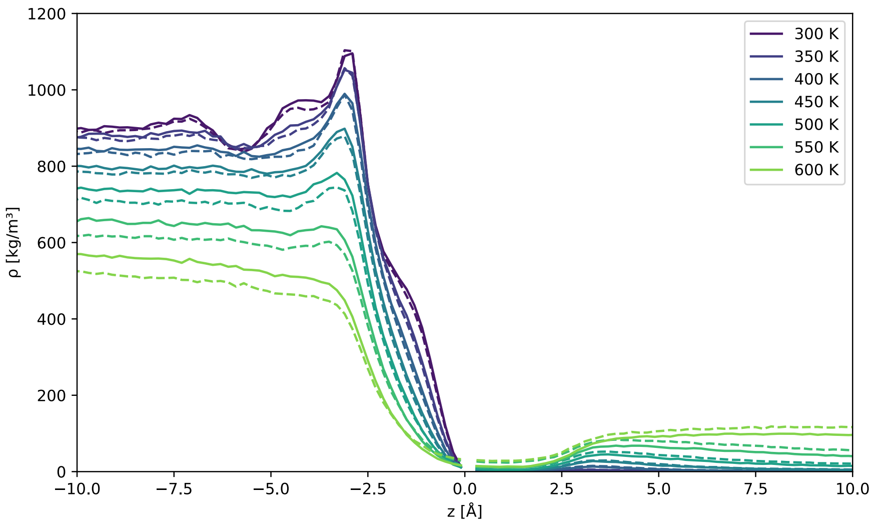
Figure 8.9 shows the intrinsic mass density profiles, which give insight into the local structure of the interface. At low temperatures, the liquid phase shows a prominent peak as well as a clear shoulder. The bottom figure in 8.9 shows the vapor phase in greater detail, where the profile also displays a less prominent peak next to the interface, though all these features start to disappear when approaching the critical point. Using each configurations pressure tensors, the surface tension γ can be calculated for each trajectory via Equation 7.8. The results are gathered in Table 8.3, once again along with Wohlfahrt's

T [K]	γ_{PIMD}	$\gamma_{classical}$	γ_{OW}
300	68(3)	65(3)	68(2)
350	54(3)	53(3)	57(2)
400	47(3)	47(3)	46(2)
440	42(3)	39(4)	40(2)
450	45(3)	37(3)	36(2)
500	27(4)	27(3)	28(2)
550	15(4)	12(4)	15(2)
600	7(4)	4(4)	8(2)
620	6(4)	5(4)	3(2)

Table 8.3.: Surface tension γ in mN/m, given at different temperatures. The subscripts of the quantities denote whether they stem from the PIMD or classical simulations, and the subscript "OW" denotes the values reported in [17], displayed here for comparison with the classical values.

data. The classical data of this work and [17] are in good agreement, while the PIMD results tend to lie slightly higher. With these values, fits are created using Equation 7.8, where $T_C = 647.096$ K was assumed fixed. The resulting fits are displayed in Figure 8.10 and have yielded the parameters $A = 186$ mN/m, $B = -0.47$ and $\mu = 1.19$ for PIMD, and $A = 260$ mN/m, $B = -0.73$ and $\mu = 1.45$ for the classical case. Unfortunately, the errors of the parameters are very large, ranging from 15 to 58% of their best fit value, and thus must be taken with a grain of salt. They can still serve as a visual guide, showing the PIMD curve getting closer to the experimental surface tension values than the classical one. Figure 8.10 includes experimental values for both H_2O and D_2O, however the impact of the isotope effects between these two data sets is almost imperceptible, while the observed difference between the PIMD results and classical results is significant. Whether this discrepancy can be traced back to the difference between isotope effects and the full inclusion of NQEs is hard to say from the available data.

It should also be noted the γ_{PIMD}-value at 450 K: The data point may seem like an outlier and arises from a simulation that had to be halted and resumed due to technical difficulties. Restarting the simulation pointed to the convergence at a comparable value after around one fifth of the original simulation time, so these events may have been coincidental. Furthermore, analysis has shown that the data point still lies just around 2 standard deviations away from the fitting curve, and some other data points also lie quite below the fitting curve when observing the values at 350 K and 550 K, so it stands to reason that the number of measurements were simply not enough to properly fit such a fluctuating quantity like the surface tension. This is also why additional trajectories at 440 K and 620 K were created, but the computational effort needed did not allow to produce more data feasibly.



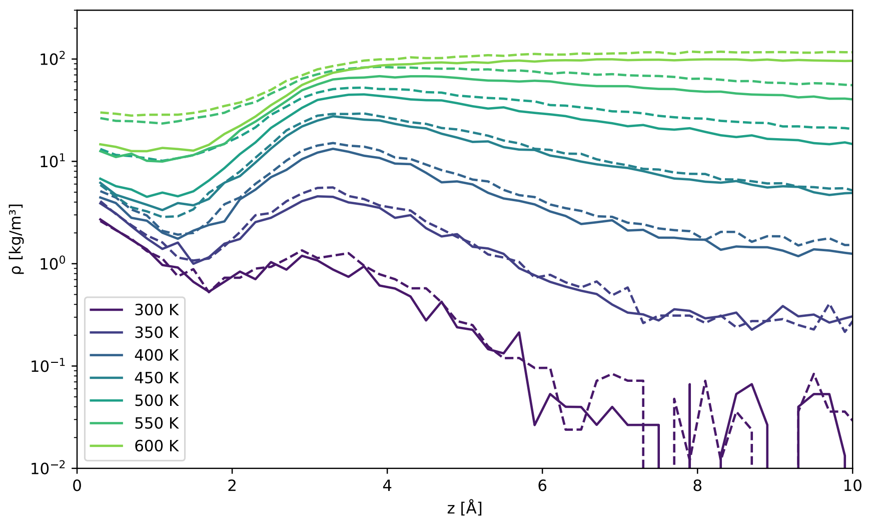


Figure 8.9.: Top: Intrinsic mass density profiles calculated from the trajectories of different temperatures. $z = 0$ is set to be the location of the surface and the resulting delta-like spike there is omitted, explaining the gap in the graph. The left hand side is the liquid phase and the right hand side is the vapor phase. The solid lines are the profiles gathered from the PIMD simulations while the dashed lines represent the classical simulations. Bottom: Intrinsic mass density profiles of the vapor phase in semi-logarithmic scale.

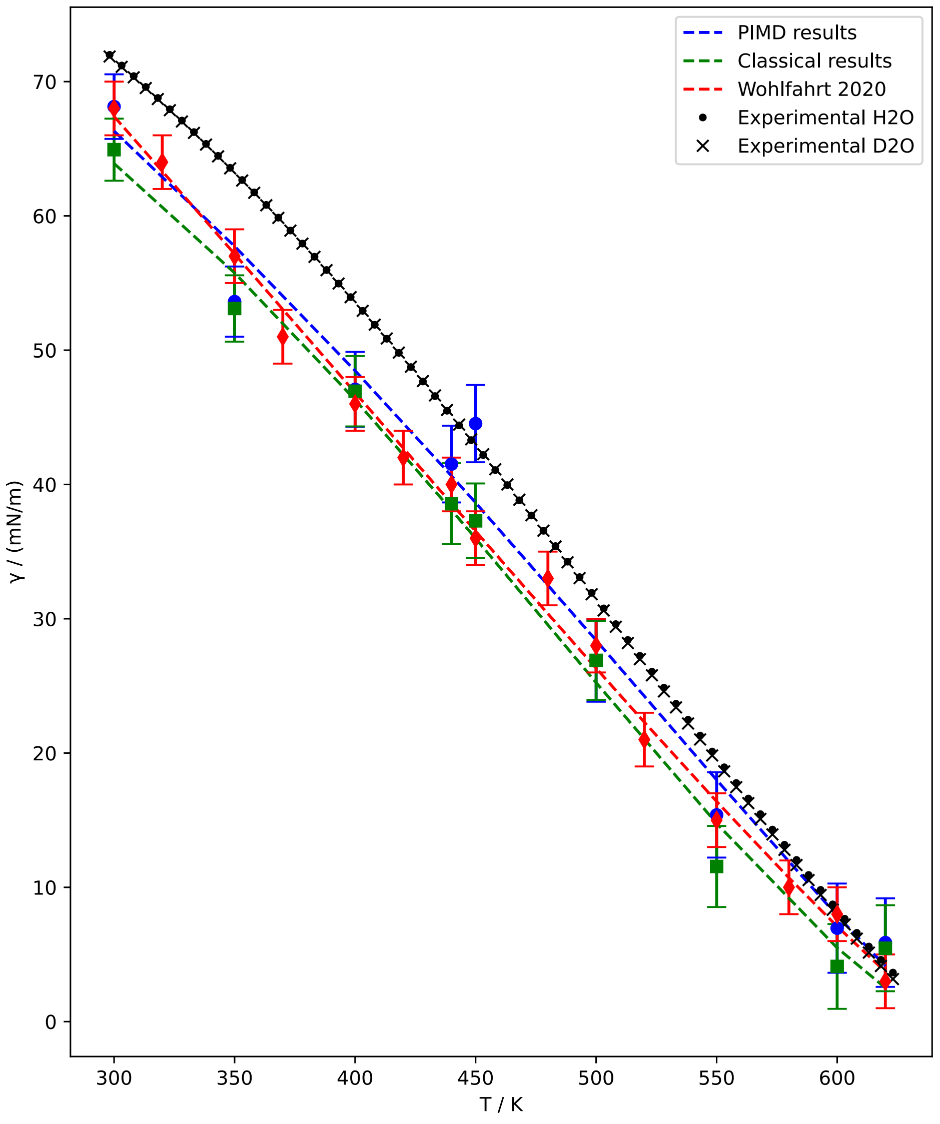


Figure 8.10.: Surface tensions. The coloured symbols represent the measured values including error bars and the dashed lines are the respective fits. Also included is the data from Wohlfahrt [17]. The black symbols are the experimental values for H_2O [116] and D_2O [68].

9. Discussion

As mentioned in the introduction, the work of Wohlfahrt [17] is supposed to serve as a basis of comparison for this work. The results showed a few deviations, especially at higher temperatures and in the intrinsic mass density profiles of the vapor phase. The question arises if these can be traced back solely to finite size effects, as Wohlfahrt used a larger system of 1024 water molecules opposed to the system of 512 water molecules in this work. To investigate this further, a system of 1024 molecules and identical box size ($l_x \times l_y \times l_z = 30 \times 30 \times 100$ Å^3) was created, equilibrated and run at a few select temperatures using the NNP classically, i.e. without the use of PIMD. Figure 9.1 shows a comparison of the intrinsic mass density profiles of this new system and the 512 molecule system used everywhere else in this work. It can be seen that while the profiles in the liquid phase are nearly identical, the peaks in the vapor phase are much more pronounced

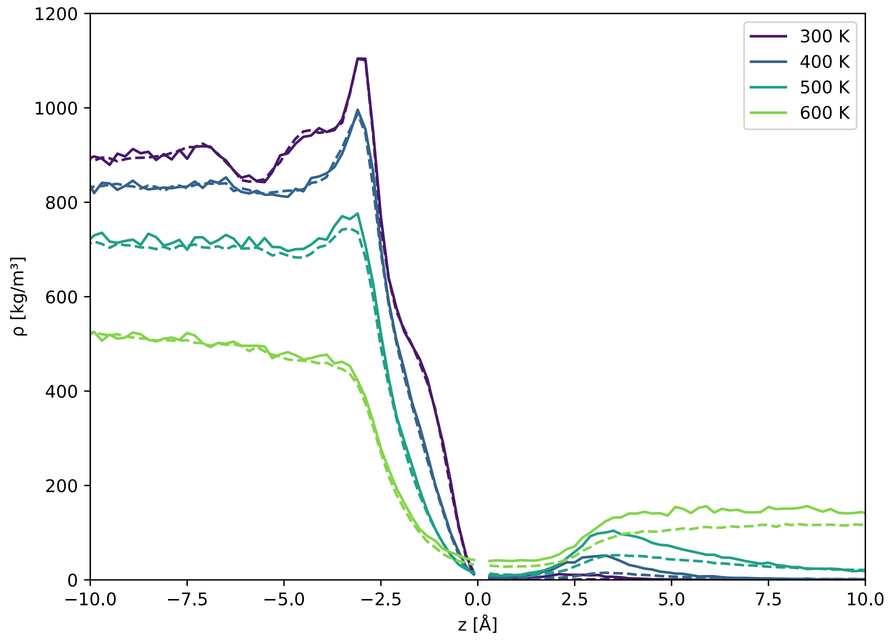


Figure 9.1.: Classically gathered intrinsic mass density profiles for a system of 1024 water molecules (solid lines) and the 512 molecule system used everywhere else in this work (dashed lines).

in the larger system, indicating the impact of finite size effects on the behaviour of the molecules in the vapor phase near the interface. Using the larger system also improved the agreement of the liquid phase density at high temperatures, coming in at 548(2) kg/m^3 at 600 K, as opposed to 535(1) kg/m^3 for the small system, compared to Wohlfahrt's 550(1) kg/m^3. However, when looking more closely at the intrinsic mass density profiles of the vapor phase of the 1024 water molecule system (see Figure 9.2), the data from this work is still missing a distinctive "crossover" between the graphs of different temperatures, a feature which can be observed in Wohlfahrt's work [17]. Thus, the discrepancies seem to stem from both finite size effects and differences in the NNP. Unfortunately, when again migrating to PIMD, the computational effort of using larger systems to alleviate the impact of finite size effects is substantially higher, so great care must be taken to balance the size of the system, the simulation time of trajectories and the number of trajectories desired.

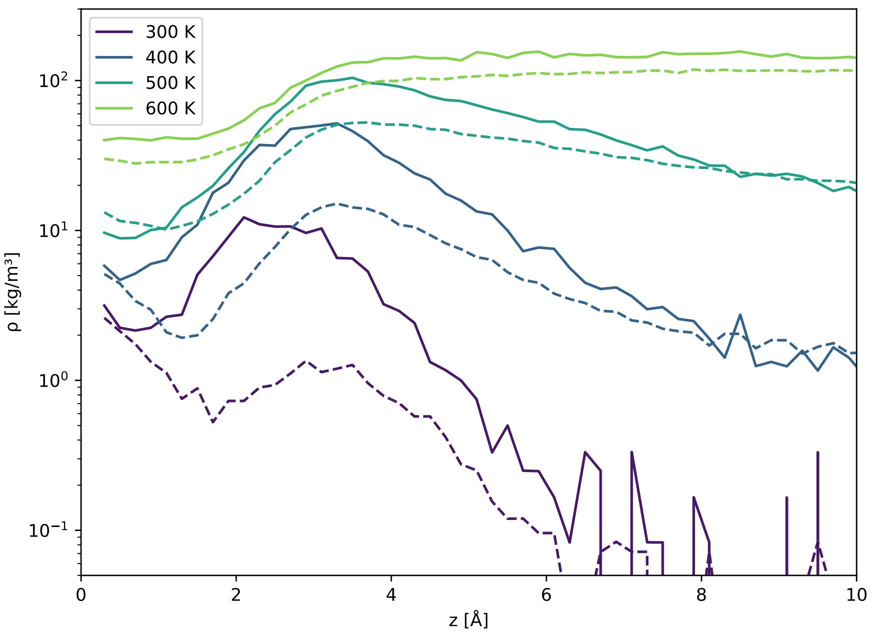


Figure 9.2.: Semi-logarithmic plot of the intrinsic mass density profiles of the vapor phase for the classical 1024 molecule (solid lines) and 512 molecule (dashed lines) systems.

It arises the remaining question how to deal with the extrapolation warnings of NNP based PIMD simulations. Naively one could believe that using $P = 32$ replicas of the system leads to 32 times as many extrapolation warnings. Table 6.1 shows that this is obviously not the case, as the PIMD case has 10^3 to 10^6 as many extrapolation warnings as the classical case (see also Figure 9.3). The increased number of warnings is logical however, when considering that the NMPIMD method treats each atom as a ring of P beads whose centroid coordinates represent the approximate real location of the atom. In other words, the position of all beads is shifted from the atom's real location, generating a differing environment probed by the symmetry functions of the NNP for which the training data set does not account for. From Figure 9.3, it can also be seen that in the classical case the onset of extrapolation warnings starts around 450 K to 500 K, while for PIMD the increase happens immediately. For high temperatures beyond 500 K, the increase in extrapolation warnings with temperature seems to be linear, though at a much higher level for the PIMD simulations. Despite all this, the results of the PIMD

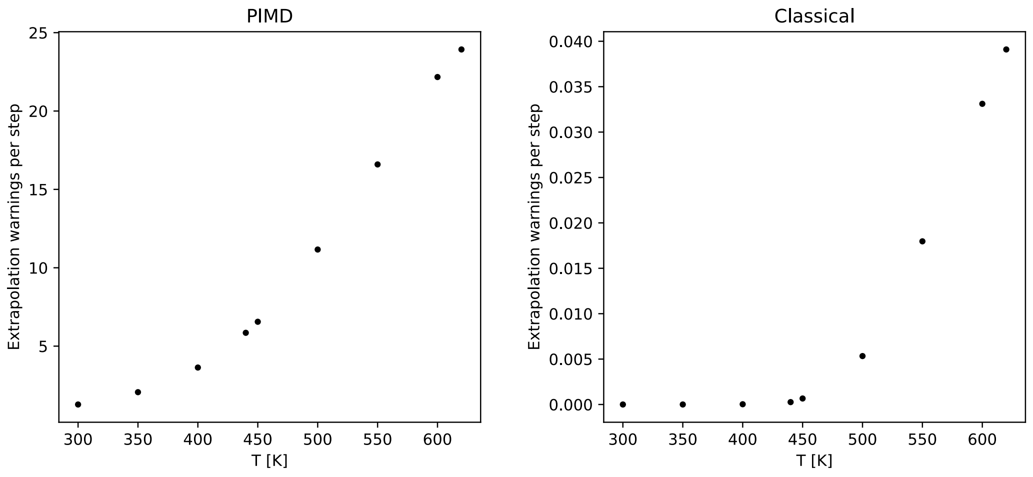


Figure 9.3.: Left: Extrapolation warnings per step for the PIMD simulations. Right: Extrapolation warnings per step for the classical simulations.

simulations are still in good accordance with the expected NQEs.

Nevertheless, care must be taken when choosing this combination of methods, also in regards to how the training datasets are crafted, and the exploration of different approaches may seem wise. For example, instead of simply training an NNP with DFT data and then performing PIMD using this NNP, one could retrain a new NNP with data computed for configurations sampled with PIMD. This would essentially create a purpose-built quantum NNP which can be used with classical MD and still capture NQEs. However, the generation of such training datasets that accurately and adequately sample the PES needed may be especially cumbersome, but may be carried out for small systems aided by the original DFT-based NNP here. Kapil et al. successfully used such an approach to perform first-principles vibrational spectroscopy of aqueous interfaces [73].

Another point worthy addressing is how the second generation of Behler-Parrinello-networks are limited to short-range interactions. As the lack of long-range interactions may lead to inaccurate results, especially at interfaces [118], it could be interesting to study the water/vapor interface with an adapted NNP that includes such contributions.

9.1. Conclusion & Outlook

The comparison of NNP aided classical and PIMD simulations in this work shows trends consistent with the expectations of NQEs. The classical data shows shorter O-H bond lengths, lower liquid densities, a lower critical temperature and a lower surface tension, all in agreement with isotope effects observed when deuterating water. The inclusion of NQEs showed a strong impact on the interface structure, with energetically unfavourable orientations of the water molecules in the first surface layer being much less probable. The coexistence curve shown in Figure 8.8 displays how properties may inch closer to real world data, even if at the moment some are more accurately reproduced using empirical models.

The approximative capabilities of NNPs work well in the context of PIMD, but the implications of the increase in extrapolation warnings should not be ignored. A study of the number of extrapolation warnings and how they may relate to parameters such as the number of beads in the polymer rings could perhaps elucidate this problem.

The effect of the inclusion of long-range interactions on the system setup considered here still remains to be seen. However, as both neural networks and path integral approaches are becoming more and more mainstream, the execution of such an adapted study using the newest generation of NNPs should be quite feasible.

Bibliography

[1] OpenAI, "Gpt-4 technical report," 2023.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[3] D. Cutts, J. S. Hoftun, A. Sornborger, C. R. Johnson, and R. T. Zeller, "Neural networks for event filtering at d0," *Computer Physics Communications*, vol. 57, no. 1, pp. 478–482, 1989.

[4] T. T. Dufera, "Deep neural network for system of ordinary differential equations: Vectorized algorithm and simulation," *Machine Learning with Applications*, vol. 5, p. 100058, 2021.

[5] E. G. Lewars, *The Concept of the Potential Energy Surface*, pp. 9–49. Cham: Springer International Publishing, 2016.

[6] M. P. Allen, *Introduction to Molecular Dynamics Simulation*, vol. 23 of *NIC series*, pp. 1–28. Jülich: John von Neumann Institute for Computing, 2004.

[7] R. Parr and Y. Weitao, *Density-Functional Theory of Atoms and Molecules*. International Series of Monographs on Chemistry, Oxford University Press, 1994.

[8] J. P. Perdew, K. Burke, and M. Ernzerhof, "Generalized gradient approximation made simple," *Phys. Rev. Lett.*, vol. 77, pp. 3865–3868, Oct 1996.

[9] B. Hammer, L. B. Hansen, and J. K. Nørskov, "Improved adsorption energetics within density-functional theory using revised perdew-burke-ernzerhof functionals," *Phys. Rev. B*, vol. 59, pp. 7413–7421, Mar 1999.

[10] S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, "A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu," *The Journal of Chemical Physics*, vol. 132, 04 2010. 154104.

[11] T. Morawietz, A. Singraber, C. Dellago, and J. Behler, "How van der waals interactions determine the unique properties of water," *Proceedings of the National Academy of Sciences*, vol. 113, no. 30, pp. 8368–8373, 2016.

[12] J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," *Phys. Rev. Lett.*, vol. 98, p. 146401, Apr 2007.

[13] J. Behler, "First principles neural network potentials for reactive simulations of large molecular and condensed systems," *Angewandte Chemie International Edition*, vol. 56, no. 42, pp. 12828–12840, 2017.

[14] J. Behler, "Four generations of high-dimensional neural network potentials," *Chemical Reviews*, vol. 121, no. 16, pp. 10037–10072, 2021.

[15] J. Zeng, D. Zhang, D. Lu, P. Mo, Z. Li, Y. Chen, M. Rynik, L. Huang, Z. Li, S. Shi, *et al.*, "Deepmd-kit v2: A software package for deep potential models," *The Journal of Chemical Physics*, vol. 159, no. 5, 2023.

[16] I. Batatia, D. P. Kovacs, G. Simm, C. Ortner, and G. Csányi, "Mace: Higher order equivariant message passing neural networks for fast and accurate force fields," *Advances in Neural Information Processing Systems*, vol. 35, pp. 11423–11436, 2022.

[17] O. Wohlfahrt, C. Dellago, and M. Sega, "Ab initio structure and thermodynamics of the RPBE-D3 water/vapor interface by neural-network molecular dynamics," *The Journal of Chemical Physics*, vol. 153, 10 2020. 144710.

[18] A. Singraber, T. Morawietz, J. Behler, and C. Dellago, "Parallel multistream training of high-dimensional neural network potentials," *Journal of Chemical Theory and Computation*, vol. 15, 04 2019.

[19] M. Ceriotti, W. Fang, P. Kusalik, R. McKenzie, A. Michaelides, M. Morales, and T. Markland, "Nuclear quantum effects in water and aqueous systems: Experiment, theory, and current challenges," *Chemical reviews*, vol. 116, 04 2016.

[20] R. P. Feynman, "Space-time approach to non-relativistic quantum mechanics," *Rev. Mod. Phys.*, vol. 20, pp. 367–387, Apr 1948.

[21] D. Chandler and P. G. Wolynes, "Exploiting the isomorphism between quantum theory and classical statistical mechanics of polyatomic fluids," *The Journal of Chemical Physics*, vol. 74, pp. 4078–4095, 04 1981.

[22] M. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation.* Oxford Graduate Texts, OUP Oxford, 2023.

[23] J. Cao and G. A. Voth, "The formulation of quantum statistical mechanics based on the feynman path centroid density. iv. algorithms for centroid molecular dynamics," *The Journal of chemical physics*, vol. 101, no. 7, pp. 6168–6183, 1994.

[24] T. D. Hone, P. J. Rossky, and G. A. Voth, "A comparative study of imaginary time path integral based methods for quantum dynamics," *The Journal of chemical physics*, vol. 124, no. 15, p. 154103, 2006.

[25] J. Cao and G. A. Voth, "The formulation of quantum statistical mechanics based on the feynman path centroid density. i. equilibrium properties," *The Journal of chemical physics*, vol. 100, no. 7, pp. 5093–5105, 1994.

[26] J. Cao and G. A. Voth, “The formulation of quantum statistical mechanics based on the feynman path centroid density. ii. dynamical properties,” *The Journal of chemical physics*, vol. 100, no. 7, pp. 5106–5117, 1994.

[27] Y. Nagata, R. E. Pool, E. H. Backus, and M. Bonn, “Nuclear quantum effects affect bond orientation of water at the water-vapor interface,” *Physical review letters*, vol. 109, no. 22, p. 226101, 2012.

[28] J. Liu, R. S. Andino, C. M. Miller, X. Chen, D. M. Wilkins, M. Ceriotti, and D. E. Manolopoulos, “A surface-specific isotope effect in mixtures of light and heavy water,” *The Journal of Physical Chemistry C*, vol. 117, no. 6, pp. 2944–2951, 2013.

[29] D. Ojha, A. Henao, F. Zysk, and T. D. Kühne, “Nuclear quantum effects on the vibrational dynamics of the water-air interface,” *arXiv preprint arXiv:2202.09592*, 2022.

[30] B. Cheng, E. A. Engel, J. Behler, C. Dellago, and M. Ceriotti, “Ab initio thermodynamics of liquid and solid water,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 4, pp. 1110–1115, 2019.

[31] A. Reinhardt and B. Cheng, “Quantum-mechanical exploration of the phase diagram of water,” *Nature communications*, vol. 12, no. 1, p. 588, 2021.

[32] V. Kapil, M. Rossi, O. Marsalek, R. Petraglia, Y. Litman, T. Spura, B. Cheng, A. Cuzzocrea, R. H. Meißner, D. M. Wilkins, *et al.*, “i-pi 2.0: A universal force engine for advanced molecular simulations,” *Computer Physics Communications*, vol. 236, pp. 214–223, 2019.

[33] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in ’t Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, “LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales,” *Comp. Phys. Comm.*, vol. 271, p. 108171, 2022.

[34] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[35] D. O. Hebb, *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

[36] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[37] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Cornell Aeronautical Laboratory. Report no. VG-1196-G-8, Spartan Books, 1962.

[38] R. E. Kalman, “A new approach to linear filtering and prediction problems,” 1960.

[39] M. Minsky, *Perceptrons : an introduction to computational geometry.* Cambridge, Mass. [u.a.]: MIT Press, 1969.

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[41] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.

[42] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[43] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[44] C. M. Handley and P. L. Popelier, "Potential energy surfaces fitted by artificial neural networks," *The Journal of Physical Chemistry A*, vol. 114, no. 10, pp. 3371–3383, 2010.

[45] J. Behler, "Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations," *Physical Chemistry Chemical Physics*, vol. 13, no. 40, pp. 17930–17955, 2011.

[46] H. Gassner, M. Probst, A. Lauenstein, and K. Hermansson, "Representation of intermolecular potential functions by neural networks," *The Journal of Physical Chemistry A*, vol. 102, no. 24, pp. 4596–4605, 1998.

[47] S. Hobday, R. Smith, and J. Belbruno, "Applications of neural networks to fitting interatomic potential functions," *Modelling and Simulation in Materials Science and Engineering*, vol. 7, no. 3, p. 397, 1999.

[48] A. Bholoa, S. D. Kenny, and R. Smith, "A new approach to potential fitting using neural networks," *Nuclear instruments and methods in physics research section B: Beam interactions with materials and atoms*, vol. 255, no. 1, pp. 1–7, 2007.

[49] S. Toxvaerd and J. C. Dyre, "Communication: Shifted forces in molecular dynamics," *The Journal of Chemical Physics*, vol. 134, no. 8, p. 081102, 2011.

[50] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," *The Journal of chemical physics*, vol. 134, no. 7, p. 074106, 2011.

[51] P. Werbos and P. John, "Beyond regression : new tools for prediction and analysis in the behavioral sciences /," 01 1974.

[52] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended kalman algorithm," *Advances in neural information processing systems*, vol. 1, 1988.

[53] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on neural networks*, vol. 9, no. 6, pp. 1456–1470, 1998.

[54] S. Haykin, "Kalman filters," *Kalman filtering and neural networks*, pp. 1–21, 2001.

[55] R. Feynman, A. Hibbs, and D. Styer, *Quantum Mechanics and Path Integrals.* Dover Books on Physics, Dover Publications, 2010.

[56] H. F. Trotter, "On the product of semi-groups of operators," *Proceedings of the American Mathematical Society*, vol. 10, no. 4, pp. 545–551, 1959.

[57] M. Herman, E. Bruskin, and B. Berne, "On path integral monte carlo simulations," *The Journal of Chemical Physics*, vol. 76, no. 10, pp. 5150–5155, 1982.

[58] M. Ceriotti, M. Parrinello, T. E. Markland, and D. E. Manolopoulos, "Efficient stochastic thermostatting of path integral molecular dynamics," *The Journal of chemical physics*, vol. 133, no. 12, 2010.

[59] B. Chen, I. Ivanov, M. L. Klein, and M. Parrinello, "Hydrogen bonding in water," *Physical Review Letters*, vol. 91, no. 21, p. 215503, 2003.

[60] J. A. Morrone and R. Car, "Nuclear quantum effects in water," *Physical review letters*, vol. 101, no. 1, p. 017801, 2008.

[61] S. Habershon, T. E. Markland, and D. E. Manolopoulos, "Competing quantum effects in the dynamics of a flexible water model," *The journal of chemical physics*, vol. 131, no. 2, 2009.

[62] D. Marx and J. Hutter, *Ab initio molecular dynamics: basic theory and advanced methods.* Cambridge University Press, 2009.

[63] X.-Z. Li, B. Walker, and A. Michaelides, "Quantum nature of the hydrogen bond," *Proceedings of the National Academy of Sciences*, vol. 108, no. 16, pp. 6369–6373, 2011.

[64] R. H. McKenzie, C. Bekker, B. Athokpam, and S. G. Ramesh, "Effect of quantum nuclear motion on hydrogen bonding," *The Journal of chemical physics*, vol. 140, no. 17, 2014.

[65] L. C. Ch'ng, A. K. Samanta, G. Czakó, J. M. Bowman, and H. Reisler, "Experimental and theoretical investigations of energy transfer and hydrogen-bond breaking in the water dimer," *Journal of the American Chemical Society*, vol. 134, no. 37, pp. 15430–15435, 2012.

[66] B. E. Rocher-Casterline, L. C. Ch'ng, A. K. Mollner, and H. Reisler, "Communication: Determination of the bond dissociation energy (d) of the water dimer,(h2o) 2, by velocity map imaging," *The Journal of chemical physics*, vol. 134, no. 21, 2011.

[67] "Nist chemistry webbook; nist standard reference database, national insitute of standards and technology." `https://webbook.nist.gov/`. Accessed: 2023-06-12.

[68] "International association for the properties of water and steam (iapws) releases, supplementary releases, guidelines, and advisory notes." `http://www.iapws.org/release.html`. Accessed: 2023-06-12.

[69] D. Lide, *CRC Handbook of Chemistry and Physics: A Ready-reference Book of Chemical and Physical Data.* CRC-Press, 1995.

[70] Y. Harada, T. Tokushima, Y. Horikawa, O. Takahashi, H. Niwa, M. Kobayashi, M. Oshima, Y. Senba, H. Ohashi, K. T. Wikfeldt, *et al.*, "Selective probing of the oh or od stretch vibration in liquid water using resonant inelastic soft-x-ray scattering," *Physical Review Letters*, vol. 111, no. 19, p. 193001, 2013.

[71] J. Schmidt, J. VandeVondele, I.-F. W. Kuo, D. Sebastiani, J. I. Siepmann, J. Hutter, and C. J. Mundy, "Isobaric- isothermal molecular dynamics simulations utilizing density functional theory: an assessment of the structure and density of water at near-ambient conditions," *The Journal of Physical Chemistry B*, vol. 113, no. 35, pp. 11959–11964, 2009.

[72] B. Pamuk, J. M. Soler, R. Ramírez, C. Herrero, P. Stephens, P. Allen, and M.-V. Fernández-Serra, "Anomalous nuclear quantum effects in ice," *Physical review letters*, vol. 108, no. 19, p. 193003, 2012.

[73] V. Kapil, D. P. Kovács, G. Csányi, and A. Michaelides, "First-principles spectroscopy of aqueous interfaces using machine-learned electronic and quantum nuclear effects," *Faraday Discussions*, 2023.

[74] L. Wang, M. Ceriotti, and T. E. Markland, "Quantum fluctuations and isotope effects in ab initio descriptions of water," *The Journal of chemical physics*, vol. 141, no. 10, 2014.

[75] B. Santra, A. Michaelides, and M. Scheffler, "Coupled cluster benchmarks of water monomers and dimers extracted from density-functional theory liquid water: The importance of monomer deformations," *The Journal of Chemical Physics*, vol. 131, no. 12, 2009.

[76] M. Gillan, D. Alfè, A. Bartók, and G. Csányi, "First-principles energetics of water clusters and ice: A many-body analysis," *The Journal of chemical physics*, vol. 139, no. 24, 2013.

[77] M. Sega, G. Hantal, B. Fábián, and P. Jedlovszky, "Pytim: A python package for the interfacial analysis of molecular simulations," *Journal of Computational Chemistry*, vol. 39, no. 25, pp. 2118–2125, 2018.

[78] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of computational physics*, vol. 117, no. 1, pp. 1–19, 1995.

[79] W. Shinoda, M. Shiga, and M. Mikami, "Rapid estimation of elastic constants by molecular dynamics simulation under constant stress," *Physical Review B*, vol. 69, no. 13, p. 134103, 2004.

[80] *LAMMPS Documentation (28 Mar 2023 version)*, pp. 1605–1613. Accessed: 2023-06-01.

[81] A. Singraber, mpbircher, S. Reeve, D. W. Swenson, J. Lauret, and philippedavid, "Compphysvienna/n2p2: Version 2.1.4," May 2021. Accessed: 2024-03-05.

[82] A. Singraber, J. Behler, and C. Dellago, "Library-based lammps implementation of high-dimensional neural network potentials," *Journal of Chemical Theory and Computation*, vol. 15, no. 3, pp. 1827–1840, 2019. PMID: 30677296.

[83] F. Heimes, "Extended kalman filter neural network training: experimental results and algorithm improvements," in *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, vol. 2, pp. 1639–1644, IEEE, 1998.

[84] X. Hu, D. V. Prokhorov, and D. C. Wunsch II, "Time series prediction with a weighted bidirectional multi-stream extended kalman filter," *Neurocomputing*, vol. 70, no. 13-15, pp. 2392–2399, 2007.

[85] "n2p2 documentation." `https://compphysvienna.github.io/n2p2/interfaces/pair_nnp.html`. Accessed: 2024-02-22.

[86] M. Ceriotti, J. More, and D. E. Manolopoulos, "i-pi: A python interface for ab initio path integral molecular dynamics simulations," *Computer Physics Communications*, vol. 185, no. 3, pp. 1019–1026, 2014.

[87] T. E. Markland and D. E. Manolopoulos, "An efficient ring polymer contraction scheme for imaginary time path integral simulations," *The Journal of chemical physics*, vol. 129, no. 2, 2008.

[88] T. E. Markland and D. E. Manolopoulos, "A refined ring polymer contraction scheme for systems with electrostatic interactions," *Chemical Physics Letters*, vol. 464, no. 4-6, pp. 256–261, 2008.

[89] G. Bussi, D. Donadio, and M. Parrinello, "Canonical sampling through velocity rescaling," *The Journal of chemical physics*, vol. 126, no. 1, 2007.

[90] G. Bussi and M. Parrinello, "Stochastic thermostats: comparison of local and global schemes," *Computer Physics Communications*, vol. 179, no. 1-3, pp. 26–29, 2008.

[91] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, "Mdanalysis: A toolkit for the analysis of molecular dynamics simulations," *Journal of Computational Chemistry*, vol. 32, no. 10, pp. 2319–2327, 2011.

[92] L. B. Pártay, G. Hantal, P. Jedlovszky, A. Vincze, and G. Horvai, “A new method for determining the interfacial molecules and characterizing the surface roughness in computer simulations. application to the liquid-vapor interface of water.,” *J Comput Chem.*, vol. 29, p. 945, 2008.

[93] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD’96, p. 226–231, AAAI Press, 1996.

[94] M. Sega and G. Hantal, “Phase and interface determination in computer simulations of liquid mixtures with high partial miscibility,” *Phys. Chem. Chem. Phys.*, vol. 19, pp. 18968–18974, 2017.

[95] M. Sega, “Marcello-sega/nn-potential-water-vapor: v1,” July 2020.

[96] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler, “Ab initio molecular simulations with numeric atom-centered orbitals,” *Computer Physics Communications*, vol. 180, no. 11, pp. 2175–2196, 2009.

[97] K. Toukan and A. Rahman, “Molecular-dynamics study of atomic motions in water,” *Physical Review B*, vol. 31, no. 5, p. 2643, 1985.

[98] M. Born and W. Heisenberg, “Zur quantentheorie der molekeln,” *Original Scientific Papers Wissenschaftliche Originalarbeiten*, pp. 216–246, 1985.

[99] R. Car, “Introduction to density-functional theory and ab-initio molecular dynamics,” *Quantitative structure-activity relationships*, vol. 21, no. 2, pp. 97–104, 2002.

[100] P. Hohenberg and W. Kohn, “Inhomogeneous electron gas,” *Physical review*, vol. 136, no. 3B, p. B864, 1964.

[101] W. Kohn and L. J. Sham, “Self-consistent equations including exchange and correlation effects,” *Physical review*, vol. 140, no. 4A, p. A1133, 1965.

[102] Y. Zhang and W. Yang, “Comment on “generalized gradient approximation made simple”,” *Physical Review Letters*, vol. 80, no. 4, p. 890, 1998.

[103] H. J. Berendsen, J. P. Postma, W. F. van Gunsteren, and J. Hermans, “Interaction models for water in relation to protein hydration,” in *Intermolecular forces: proceedings of the fourteenth Jerusalem symposium on quantum chemistry and biochemistry held in jerusalem, israel, april 13–16, 1981*, pp. 331–342, Springer, 1981.

[104] P. K. Yuet and D. Blankschtein, “Molecular dynamics simulation study of water surfaces: comparison of flexible water models,” *The Journal of Physical Chemistry B*, vol. 114, no. 43, pp. 13786–13795, 2010.

[105] *LAMMPS Documentation (28 Mar 2023 version)*, pp. 1234–1238. Accessed: 2023-06-01.

[106] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles.* crc Press, 2021.

[107] *LAMMPS Documentation (28 Mar 2023 version)*, pp. 1030–1037. Accessed: 2023-06-01.

[108] E. Jones, T. Oliphant, and P. Peterson, "Scipy: Open source scientific tools for python," 01 2001.

[109] A. Morita and J. T. Hynes, "A theoretical analysis of the sum frequency generation spectrum of the water surface," *Chemical Physics*, vol. 258, pp. 371–390, Aug. 2000.

[110] M. Sega and C. Dellago, "Long-range dispersion effects on the water/vapor interface simulated using the most common models," *The Journal of Physical Chemistry B*, vol. 121, no. 15, pp. 3798–3803, 2017.

[111] E. Chacón and P. Tarazona, "Intrinsic profiles beyond the capillary wave theory: A monte carlo study," *Phys. Rev. Lett.*, vol. 91, p. 166103, Oct 2003.

[112] P. Wagner, G. Reischl, and G. Steiner, *Einführung in die Physik.* Facultas, 2. ed., 2012.

[113] N. B. Wilding, "Critical-point and coexistence-curve properties of the lennard-jones fluid: A finite-size scaling study," *Phys. Rev. E*, vol. 52, pp. 602–611, Jul 1995.

[114] J. G. Kirkwood and F. P. Buff, "The statistical mechanical theory of surface tension," *J. Chem. Phys.*, vol. 17, no. 3, p. 338, 1949.

[115] J. Alejandre, D. J. Tildesley, and G. A. Chapela, "Molecular dynamics simulation of the orthobaric densities and surface tension of water," *J. Chem. Phys.*, vol. 102, p. 4574, 1995.

[116] N. B. Vargaftik, B. N. Volkov, and L. D. Voljak, "International Tables of the Surface Tension of Water," *Journal of Physical and Chemical Reference Data*, vol. 12, pp. 817–820, 07 1983.

[117] A. Soper and C. Benmore, "Quantum differences between heavy and light water," *Physical review letters*, vol. 101, no. 6, p. 065502, 2008.

[118] H. Hao, L. Ruiz Pestana, J. Qian, M. Liu, Q. Xu, and T. Head-Gordon, "Chemical transformations and transport phenomena at interfaces," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 13, no. 2, p. e1639, 2023.

List of Tables

List of Figures

A. Symmetry Functions

No.	Element j	Element k	η	r_s	λ	ζ	r_c
1	H	-	0.001	0.0	-	-	12
2	H	-	0.010	0.0	-	-	12
3	H	-	0.030	0.0	-	-	12
4	H	-	0.060	0.0	-	-	12
5	H	-	0.150	1.9	-	-	12
6	H	-	0.300	1.9	-	-	12
7	H	-	0.600	1.9	-	-	12
8	H	-	1.500	1.9	-	-	12
9	O	-	0.001	0.0	-	-	12
10	O	-	0.010	0.0	-	-	12
11	O	-	0.030	0.0	-	-	12
12	O	-	0.060	0.0	-	-	12
13	O	-	0.150	0.9	-	-	12
14	O	-	0.300	0.9	-	-	12
15	O	-	0.600	0.9	-	-	12
16	O	-	1.500	0.9	-	-	12
17	O	H	0.010	0.0	-1	4	12
18	O	H	0.010	0.0	1	4	12
19	O	H	0.030	0.0	-1	1	12
20	O	H	0.030	0.0	1	1	12
21	O	H	0.070	0.0	-1	1	12
22	O	H	0.070	0.0	1	1	12
23	O	H	0.200	0.0	1	1	12
24	O	O	0.001	0.0	-1	4	12
25	O	O	0.001	0.0	1	4	12
26	O	O	0.030	0.0	-1	1	12
27	O	O	0.030	0.0	1	1	12

Table A.1.: Parameters for the radial (no. 1-16, using G_i^2) and angular (no. 17-27, using G_i^3) symmetry functions for hydrogen. η is given in Bohr^{-2} and r_s and r_c are given in Bohr.

No.	Element j	Element k	η	r_s	λ	ζ	r_c
1	H	-	0.001	0.0	-	-	12
2	H	-	0.010	0.0	-	-	12
3	H	-	0.030	0.0	-	-	12
4	H	-	0.060	0.0	-	-	12
5	H	-	0.150	0.9	-	-	12
6	H	-	0.300	0.9	-	-	12
7	H	-	0.600	0.9	-	-	12
8	H	-	1.500	0.9	-	-	12
9	O	-	0.001	0.0	-	-	12
10	O	-	0.010	0.0	-	-	12
11	O	-	0.030	0.0	-	-	12
12	O	-	0.060	0.0	-	-	12
13	O	-	0.150	4.0	-	-	12
14	O	-	0.300	4.0	-	-	12
15	O	-	0.600	4.0	-	-	12
16	O	-	1.500	4.0	-	-	12
17	H	H	0.010	0.0	-1	4	12
18	H	H	0.010	0.0	1	4	12
19	H	H	0.030	0.0	-1	1	12
20	H	H	0.030	0.0	1	1	12
21	H	H	0.070	0.0	-1	1	12
22	H	H	0.070	0.0	-1	1	12
23	O	H	0.001	0.0	-1	4	12
24	O	H	0.001	0.0	1	4	12
25	O	H	0.030	0.0	-1	1	12
26	O	H	0.030	0.0	1	1	12
27	O	O	0.001	0.0	-1	4	12
28	O	O	0.001	0.0	1	4	12
29	O	O	0.030	0.0	-1	1	12
30	O	O	0.030	0.0	1	1	12

Table A.2.: Parameters for the radial (no. 1-16, using G_i^2) and angular (no. 17-30, using G_i^3) symmetry functions for oxygen. η is given in Bohr^{-2} and r_s and r_c are given in Bohr.

B. Bivariate distributions

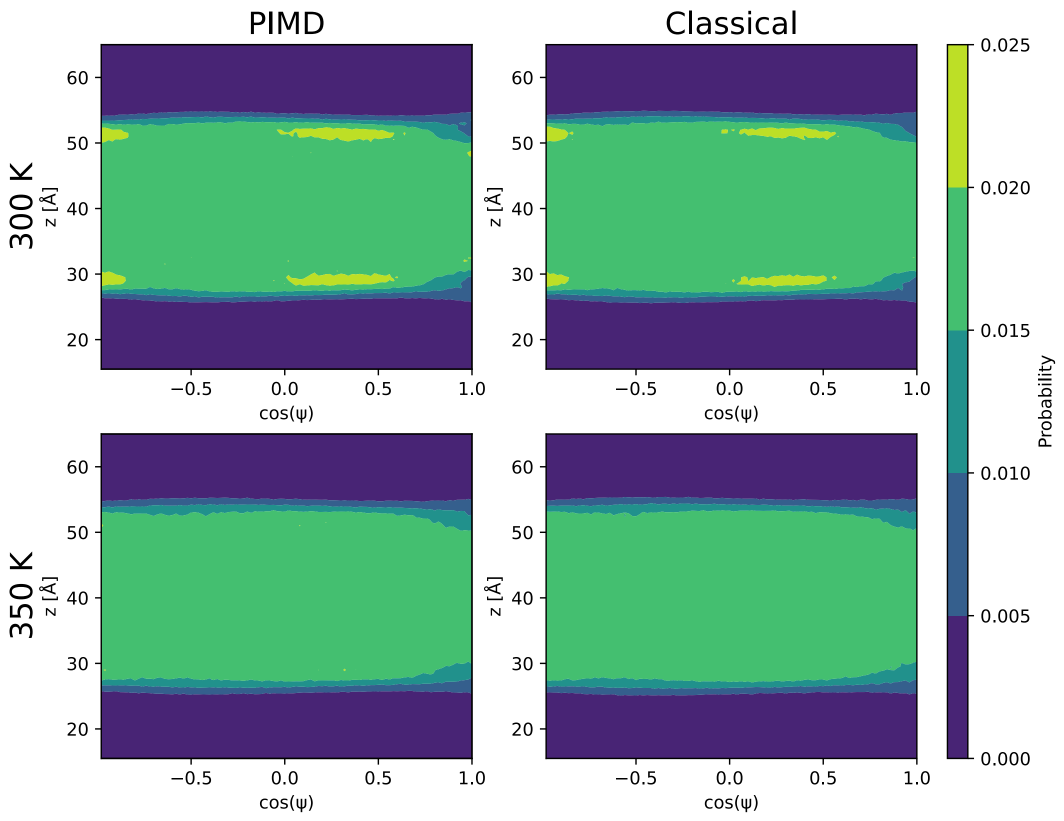


Figure B.1.: Distribution of the angle ψ between O-H bonds and the z-axis (or negative z-axis for molecules located in the lower half of the slab) for 300 K and 350 K.

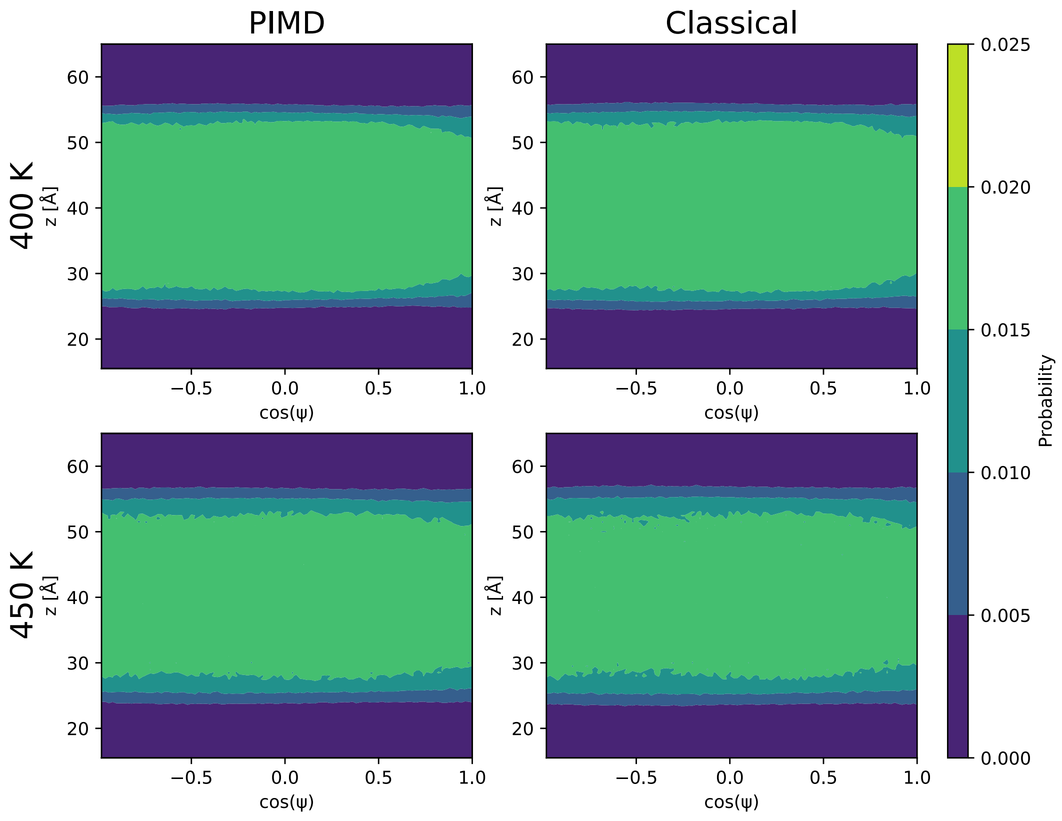


Figure B.2.: Distribution of the angle ψ between O-H bonds and the z-axis (or negative z-axis for molecules located in the lower half of the slab) for 400 K and 450 K.

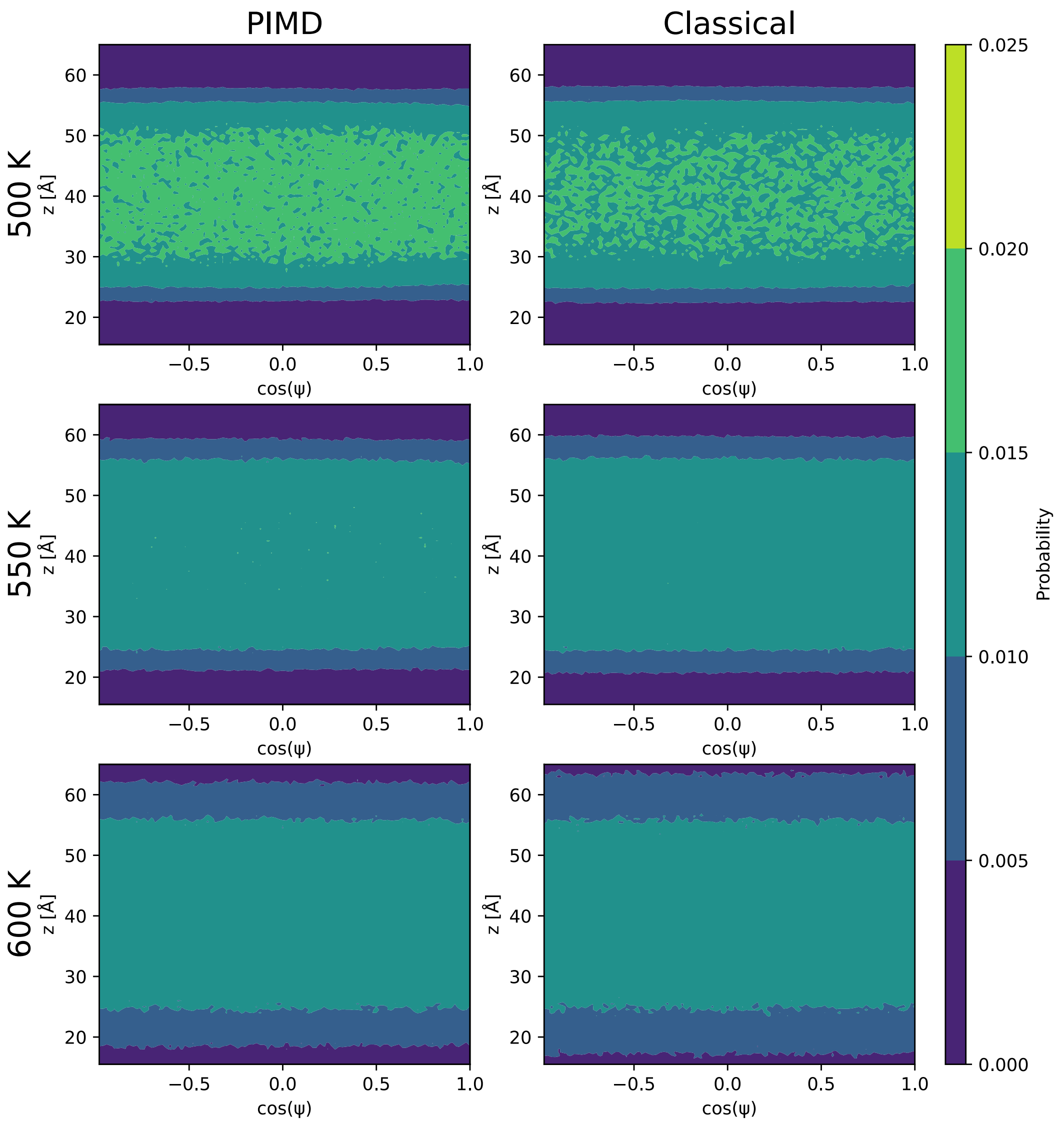


Figure B.3.: Distribution of the angle ψ between O-H bonds and the z-axis (or negative z-axis for molecules located in the lower half of the slab) for 500 K, 550 K and 600 K.

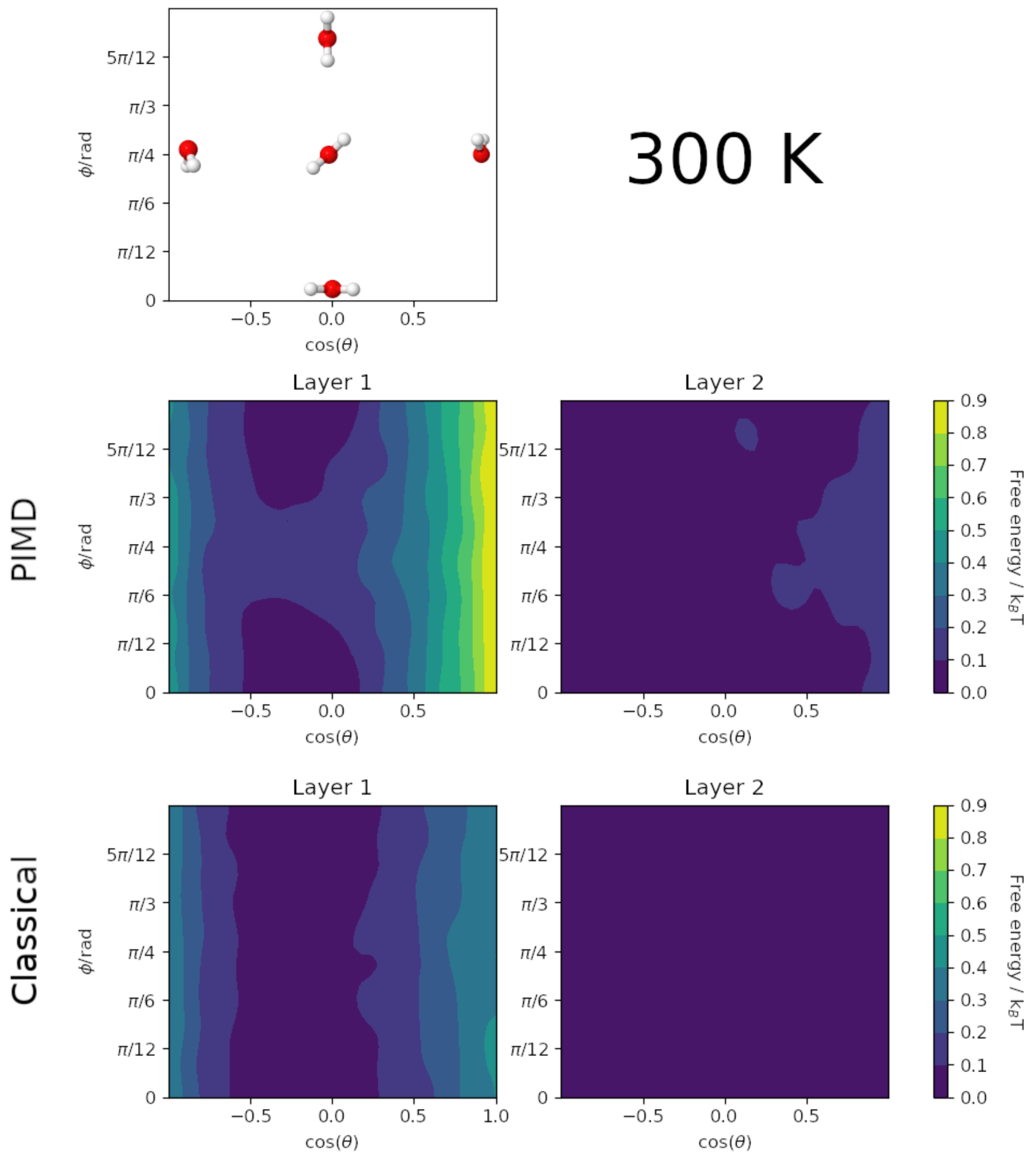


Figure B.4.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 300 K.

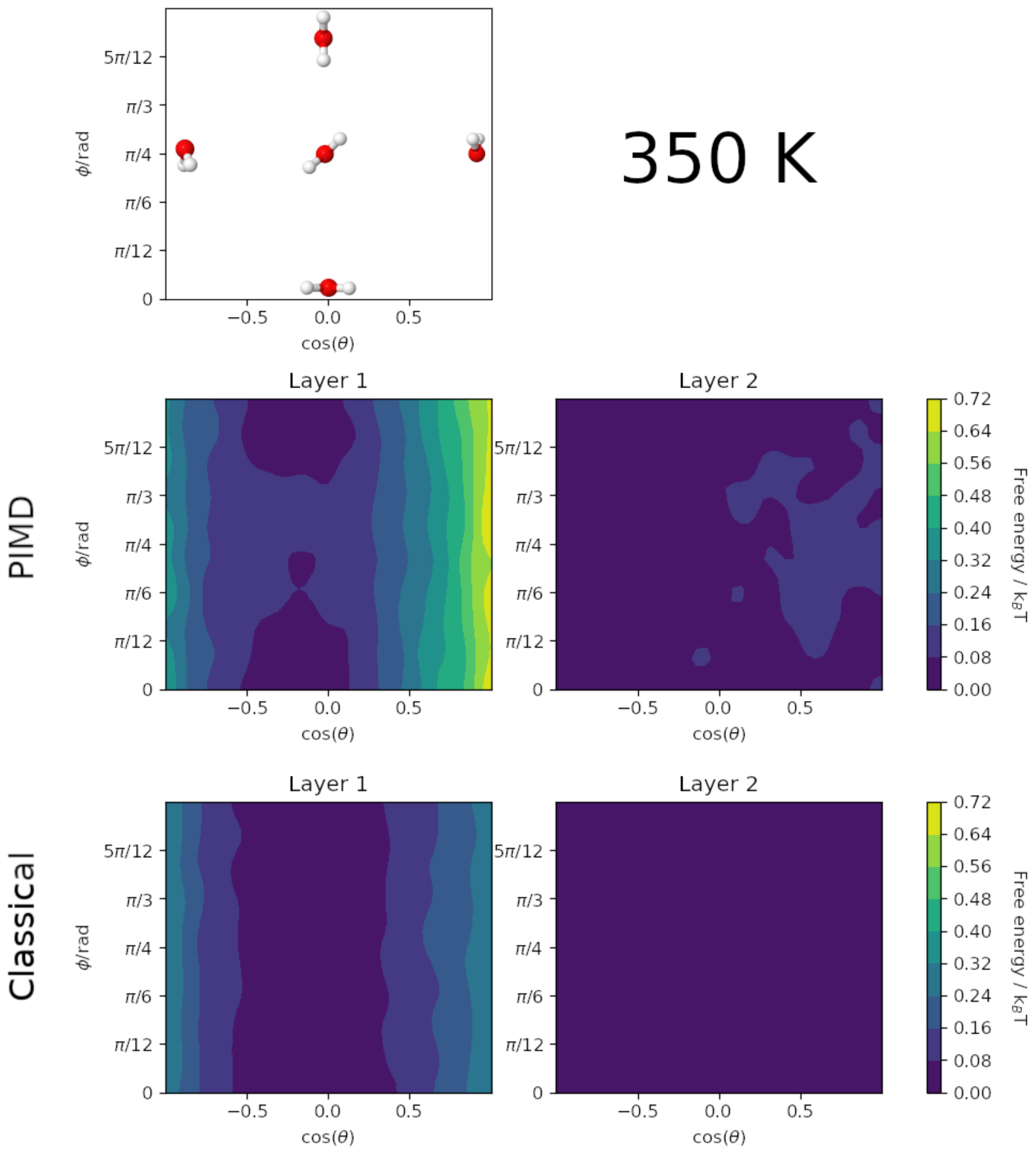


Figure B.5.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 350 K.

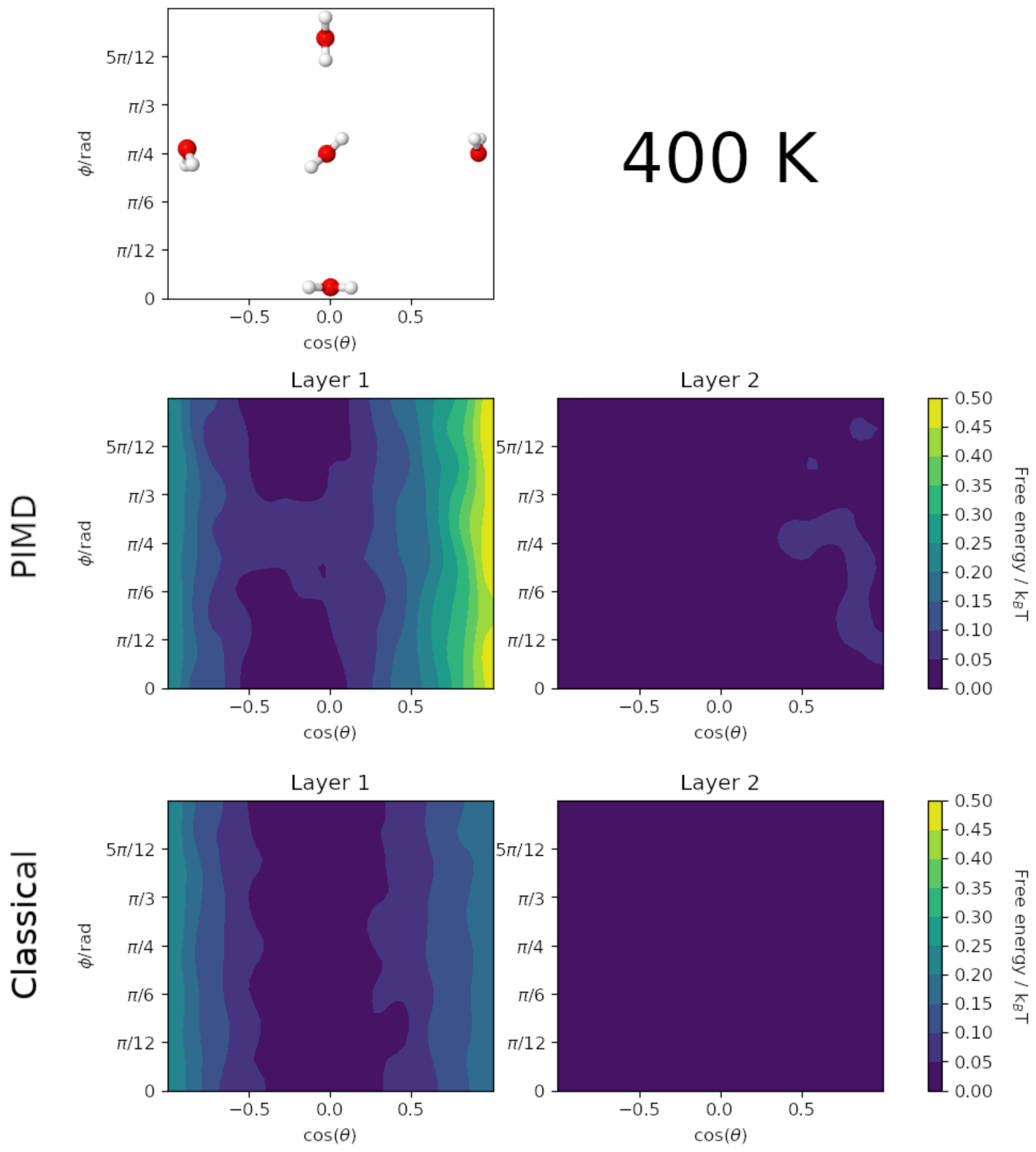


Figure B.6.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 400 K.

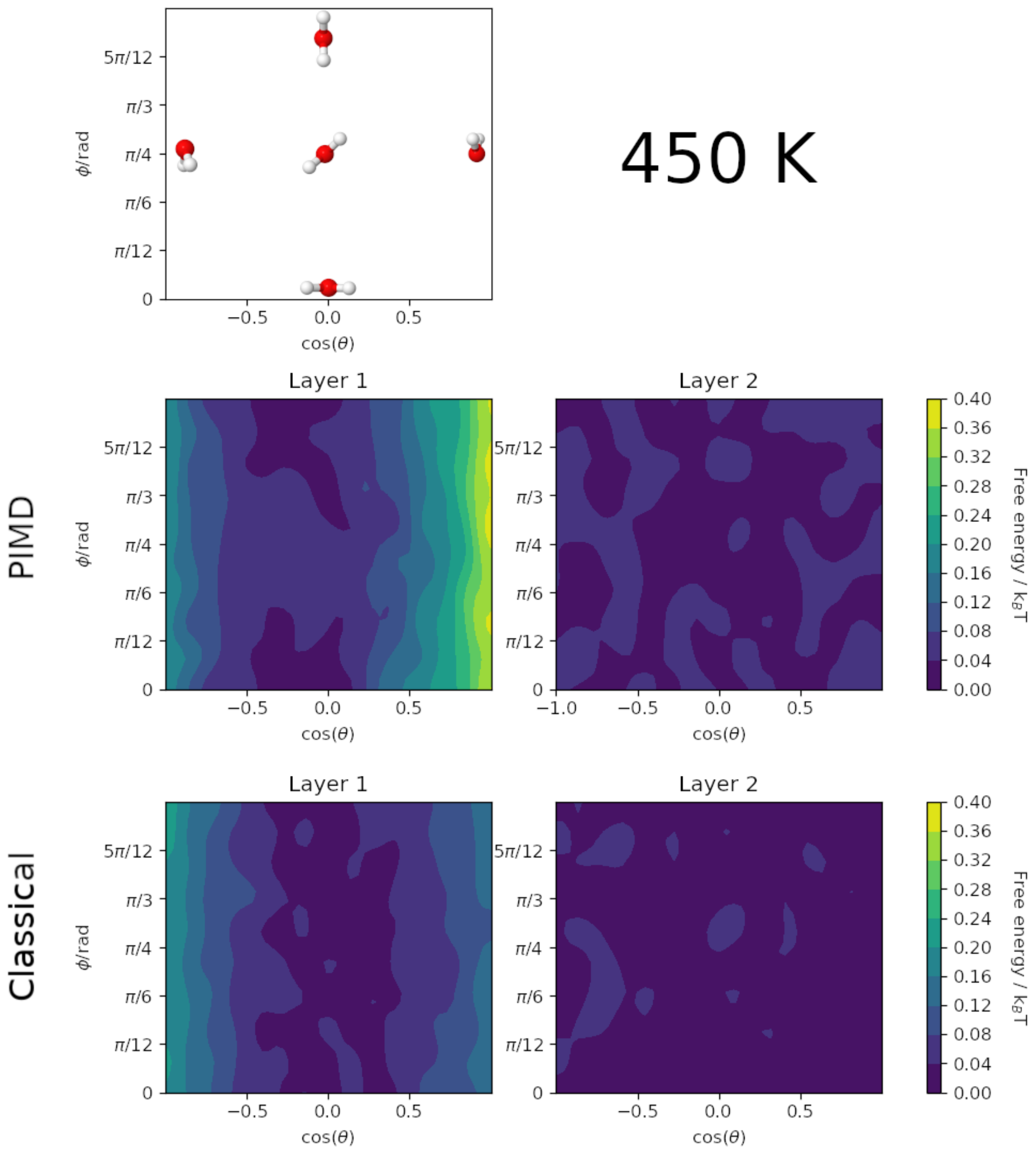


Figure B.7.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 450 K.

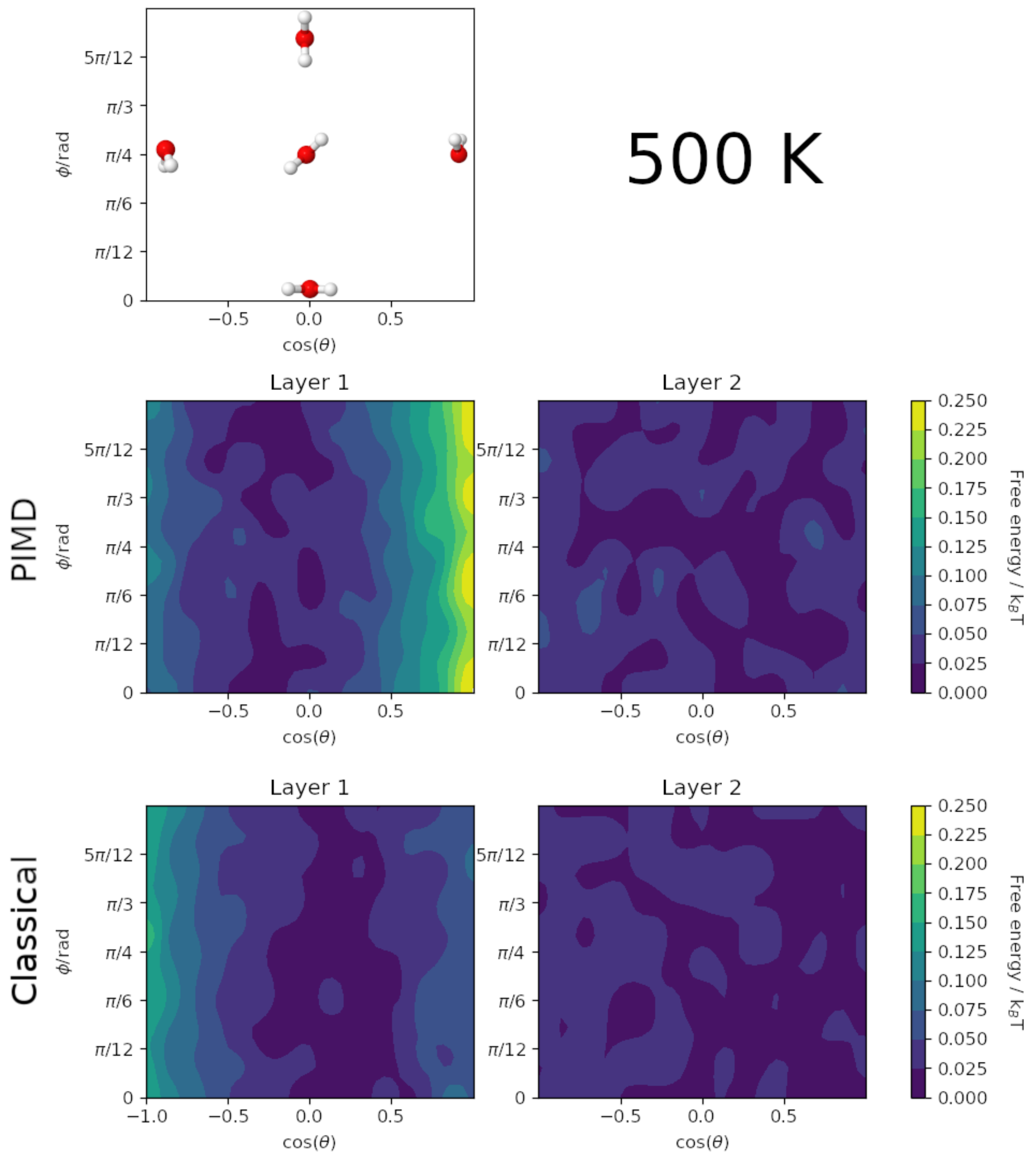


Figure B.8.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 500 K.

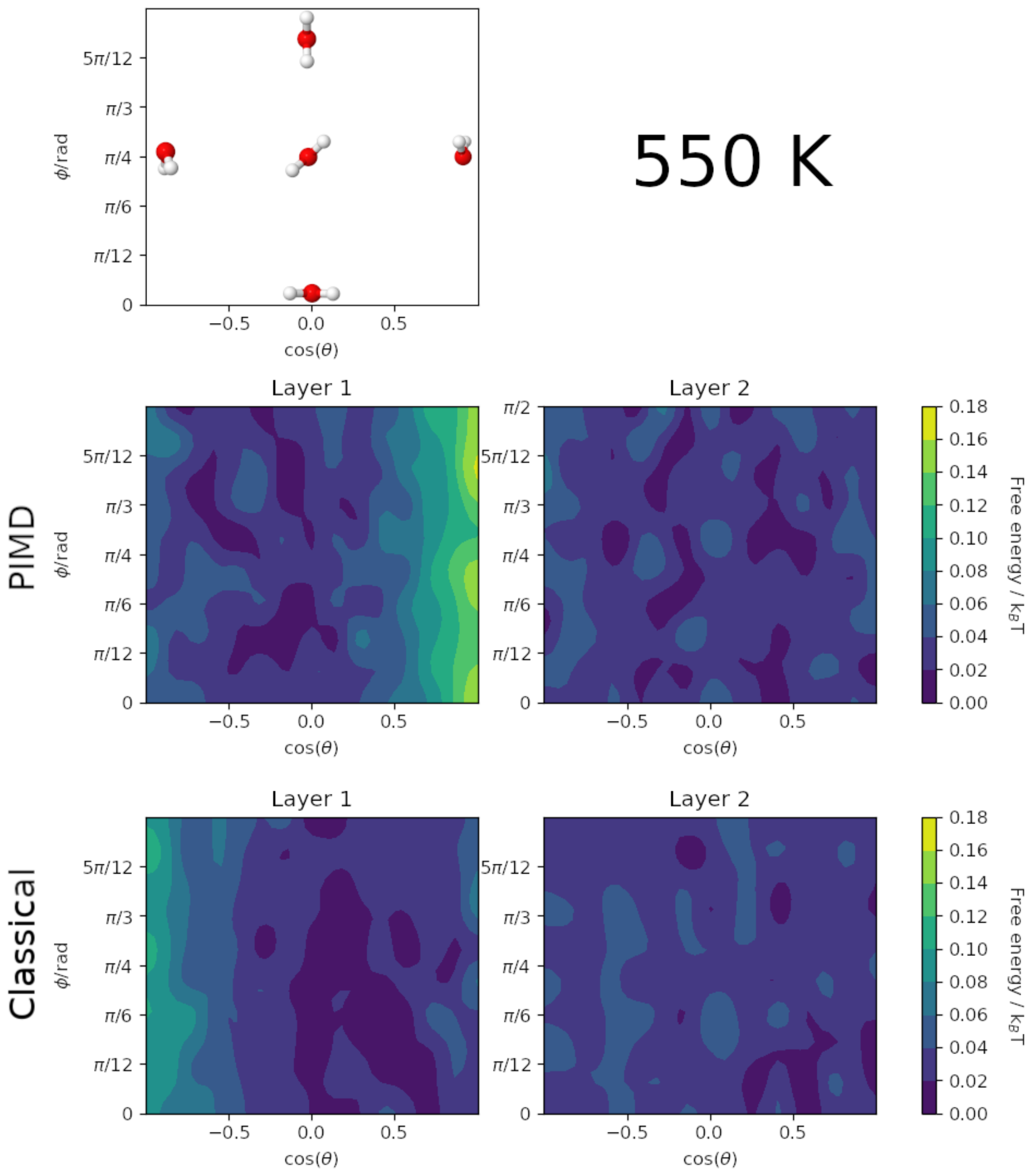


Figure B.9.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 550 K.

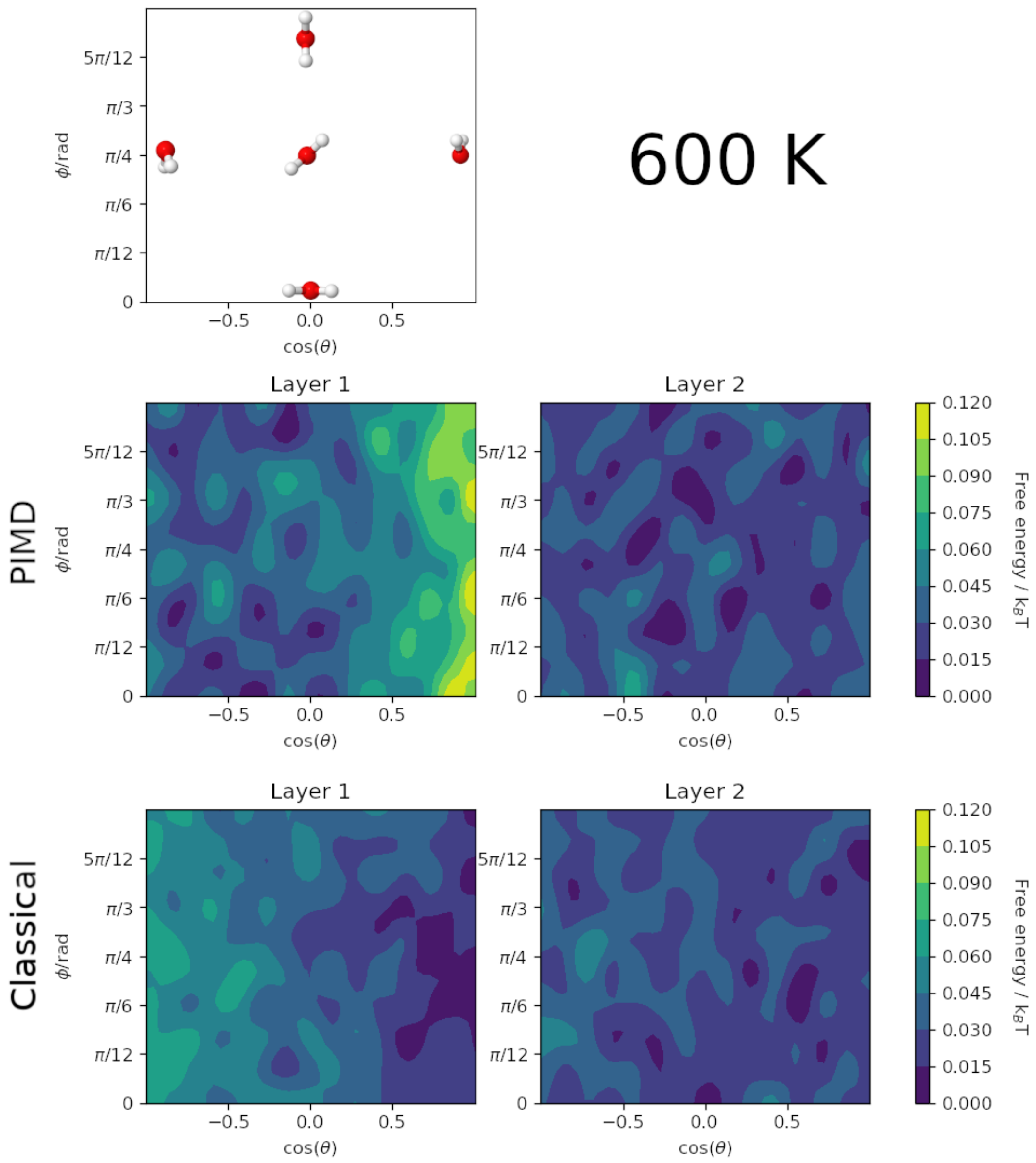


Figure B.10.: Bivariate distributions of the angles θ and ϕ in terms of free energy for 600 K.