MASTER

A Permutation Invariant H2–H2Potential Energy Surface for the Purpose of Studying Vibrational Energy Transfer

Hendriks, F.

Award date:
2021

A Permutation Invariant H_2–H_2 Potential Energy Surface for the Purpose of Studying Vibrational Energy Transfer

Fleur Hendriks (0850287)

January 26, 2021

Abstract

To model the reaction speeds of vibrational energy transfer between molecules, potential energy surfaces are important. Various methods of fitting a full dimensional potential energy surface that is invariant with respect to permutation of identical atoms to ab initio data are discussed. Several methods are implemented for the H_2–H_2 system, using a data set with results of 7291 coupled cluster single-double (CCSD) calculations with the basis set aug-cc-pVTZ. The results are compared to existing methods. Two methods were the main focus of this report; the so-called Proximity Matrix Eigenspectrum (PME) method that uses a neural network, and the Proximity Matrix Invariants (PMI) method that uses a linear regression. These methods achieved a root-mean-square error (RMSE) of (19.58 ± 0.66) meV and (10.83 ± 0.45) meV, respectively. A potential energy surface from the PME method was used for full-dimensional quasiclassical trajectory simulations that were used to approximate cross sections for dissociation as well as vibrational and rotational energy transfer. A major flaw in the PME method is the occurrence of barely avoided eigenvalue crossings. This flaw is investigated and was solved by switching to the PMI method. An important advantage of this approach is that it scales favourably to larger systems in comparison to the already existing method using permutationally invariant polynomials by Braams and Bowman (International Reviews in Physical Chemistry (28:4), p.577-606).

Acknowledgements

First of all I would like to thank dr. ir. Jesper Janssen of Plasma Matters and the University of Technology Eindhoven for his guidance, explanations and advice, as well as coming up with the idea of applying machine learning to the problem of vibrational energy transfer. He also gets credit for the C++ implementation of the neural network and the bond-bond potential as well his patience and help in the debugging of the Hinde potential.

I am grateful to dr. ir. Jan van Dijk of the University of Technology Eindhoven for his supervision, and also specifically for help with the C++ implementations.

I am also grateful for the advice and explanations of quantum chemistry concepts from dr. Jos Suijker of Signify and the University of Technology Eindhoven. Credit for the Braams-Bowman PIP method implementation goes to him, as well for investigating the mysterious peak in the CO_2 potential and calculating the counterpoise correction for the H_2–H_2 system.

My gratitude also goes to Prof. dr. ir. Vianney Koelman of CCER and the University of Technology Eindhoven , for sharing his knowledge on machine learning applied to physics problems, and specifically for coming up with the PME and PMI methods.

I would also like to thank dr. ir. Wouter Graef of Plasma Matters and the University of Technology Eindhoven for investigating weird software problems and going to the locked-down university in the middle of a pandemic specifically to press a button on the computer that I crashed. And then doing it again a few days later.

I would like to thank all of them for the brainstorming sessions which were often very insightful, and adapting to virtual meetings, making it possible for me to finish my thesis in good order despite the current state of the world.

Lastly I would like to thank my family for their support and encouragement. If I had to be cooped up in a house with four other people for the better part of a year, I am glad it was you.

Contents

Chapter 1

Introduction

1.1 Background

Global warming is widely regarded as a problem [1, 2]. The rising concentration of carbon dioxide (CO_2) is regarded as the most significant cause of global warming [3]. To turn CO_2 from a problem into a solution, CO_2 can be captured and split into carbon monoxide and oxygen, using a source of energy that does not contribute to global warming. The resulting carbon monoxide can then be used for the production of syngas, which creates a cycle of combustion and capture that is carbon neutral in total [4]. This decomposition should be as energy efficient as possible. It starts with CO_2 dissociation and oxygen then reacts to form O_2. This can be summarized by

$$CO_2 \longrightarrow CO + \frac{1}{2} O_2, \tag{1.1}$$

and this has an enthalpy of $\Delta H = 2.9\,\mathrm{eV/mol}$ [5, p. 259]. The efficiency of the process is then

$$\eta = \Delta H / E_{CO}, \tag{1.2}$$

with E_{CO} the actual energy cost per CO molecule. When using only thermal decomposition, the possible efficiency is limited to at most 45% (a cost of $6.4\,\mathrm{eV/mol}$), because all degrees of freedom (translational, vibrational, rotational degrees of freedom) get a similar share of the energy, but the vibrational energy is much more useful for dissociation [5, p. 262]. For this reason, in a non-equilibrium plasma (meaning vibrational temperature T_v is much higher than translational temperature T_0) the efficiency can be as high as 80%. Since the influence of the vibrational excitation is so large, cross sections of state-to-state vibrational energy transfer are of great importance in simulations of plasmas that breakdown CO_2 by vibrational excitation [6].

These cross sections can be approximated using statistics, from collision trajectories simulations. However, trajectories like these need an accurate potential for the interaction between the two reacting molecules. One of the most accurate ways to obtain this potential energy surface (PES) is from ab initio calculations, but this is usually very slow. A better approach is to approximate the potential in some way. For this, usually potential points obtained with ab initio methods are fit. For certain molecules, this can be done with a many-body type expansion [7] or with polynomials [8, 9, 10], but for a more general approach artificial neural networks can be used, since training a neural network is essentially performing high-dimensional curve fitting.

Neural networks are a form of machine learning, which is a collection of techniques that can learn from data by building a mathematical model, without using explicit instructions about the shape of this model. The advantage of a neural network approach is that it provides possibilities to scale up to systems with more atoms, and because it is such a flexible approach, smaller data sets might be sufficient than

for other fitting methods. Machine learning is increasingly being applied to physics problems, for example for controlling dynamic systems [11, 12], simulating galaxy formation [13], learning phase transitions [14], exploring geometries relevant to string theory [15], analysing the atmospheres of exoplanets [16], or improving ab initio calculations directly [17, 18].

Neural networks have been applied to PESs before. For example, they have been used to model surface diffusion of $CO/Ni(_{111})$ [19], hydrogen dissociation on metal surfaces [20], molecular atomization energies [21], the potential of TiO_2 [22], the potential surface of amorphous Li_3PO_3 [23] and atomization energies of organic molecules [24].

1.2 Goals

One obstacle in applying neural networks to PESs, is that one needs a way to convert an atom configuration into an input for the neural network that respects translation, rotation and permutation symmetry of identical atoms, preferably in a way that is easily scalable so it can be used for larger systems as well. In this report, two yet unpublished proximity matrix methods by prof. dr. ir. Vianney Koelman are used. An important advantage of these approaches is that they scale favourably to larger systems in comparison to the already existing method using permutationally invariant polynomials (PIPs) by Braams and Bowman [25]. The goals of this study are

- to give an overview of various methods of achieving permutation invariance,
- to implement and test the two new proximity matrix methods,
- to compare them to the existing PIP method,
- and to test the application of these methods to the study of vibrational energy transfer.

1.3 Outline

As a first test, first the H_2–H_2 system is modelled. Because a H_2-molecule has only two electrons, it is very cheap to use ab initio methods on it, and the PES of H_2–H_2 is well-studied, which makes it a good prototype system. A H_2–H_2 system only has 6 degrees of freedom ($4 \times 3 - 6 = 6$, 1 internal per molecule, 4 between the molecules), which makes its PES simpler than that of CO_2–CO_2 which has 12 degrees of freedom ($6 \times 3 - 6 = 12$, 3 internal per molecule, 6 between the molecules).

Figure 1.1 shows the basic steps needed to obtain the desired cross sections, with on the left the approach used in this report involving neural networks. The first step in Figure 1.1 is to use ab initio methods to create a data set of potential points. In Chapter 2, the methodology and the quantum mechanics theory behind these ab initio calculations, and the accuracy of the used methods is discussed. This sampling is performed by calculating some trajectories and picking configurations from these trajectories to do an ab initio calculation with. A neural network is then used to make a fit of these potential points. In Chapter 3, the relevant machine learning theory is discussed and the results of the neural network are shown. The trained neural network then provides a PES that can be used to calculate more trajectories using molecular dynamics methods, much faster than with an ab initio method. In Chapter 4 this molecular dynamics part of the project is discussed, and the results are processed into state-to-state cross sections and probabilities. Chapter 5 is the conclusion. The coordinate systems are explained in Appendix A.

All code related to this project is stored in a Git repository that is operated by the group EPG. All the generated data is also stored. For more details, contact Jan van Dijk (j.v.dijk@tue.nl).

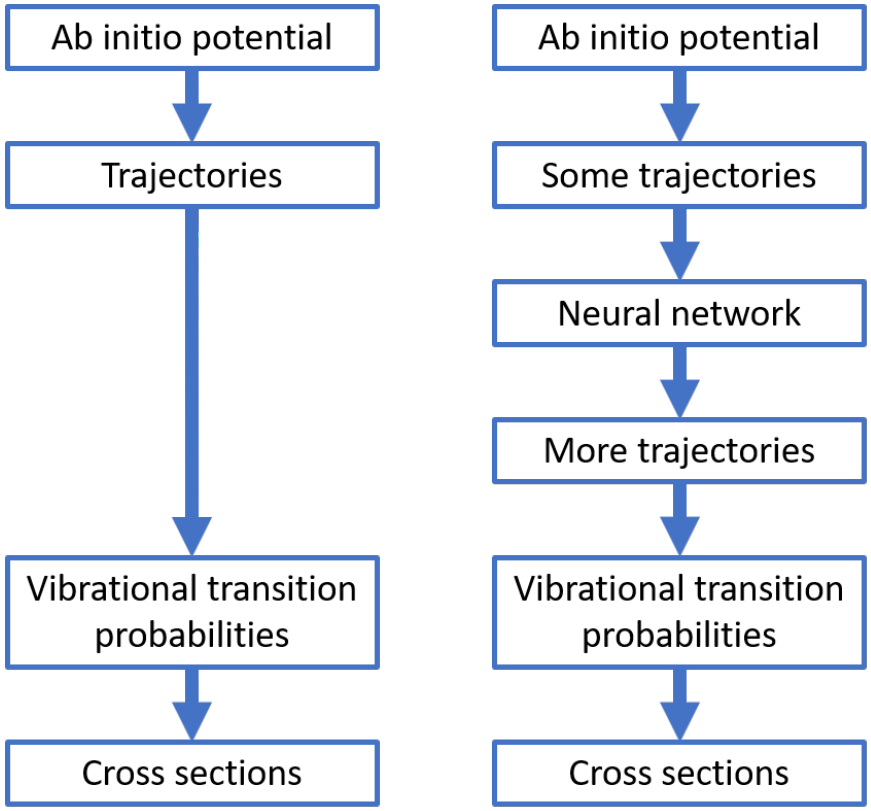


Figure 1.1: Flowchart of the approach without (left) and with a neural network (right).

Chapter 2

Quantum Chemistry

Quantum chemistry is a field where quantum physics and chemistry overlap; it is concerned with the application of quantum mechanics to chemical systems. A big part of this is predicting and analyzing electronic structure, and understanding how quantum effects affect molecular dynamics. This chapter concerns the methods used to compute the data points for the data set. The quantum mechanics theory about the vibrational state of the molecules that is relevant to the calculation of the trajectories is covered in Chapter 4.

2.1 Theory

2.1.1 The electronic problem

Ab initio ('from the beginning' or 'from the ground up' or 'from first principles') methods attempt to solve the electronic Schrödinger equation given the positions of the nuclei and the number of electrons, without using any empirical fits. The goal is usually to construct approximate solutions $|\Phi\rangle$ of the non-relativistic time-independent Schrödinger equation [26]

$$\mathcal{H}\,|\Phi\rangle = \mathcal{E}\,|\Phi\rangle\,, \tag{2.1}$$

where $\mathcal{E}$ is the energy of the solution $|\Phi\rangle$. The Hamiltonian $\mathcal{H}$ for a system of M nuclei and N electrons is [26]

$$\begin{aligned}\mathcal{H} = &-\sum_{i=1}^{N}\frac{1}{2}\boldsymbol{\nabla}_i^2 - \sum_{A=1}^{M}\frac{1}{2M_A}\boldsymbol{\nabla}_A^2 \\ &-\sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{r_{ij}} + \sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{R_{AB}}.\end{aligned} \tag{2.2}$$

Here i and j iterate over the electrons and A and B iterate over the nuclei. $\boldsymbol{\nabla}_i^2$ and $\boldsymbol{\nabla}_A^2$ are the Laplace operator with respect to the coordinates of the electrons and the nuclei, respectively. M_A is the mass of the nucleus A, Z_A is the atomic number of nucleus A, r_{ij} is the distance between the electrons i and j and R_{AB} is the distance between the nuclei A and B. The units in (2.2) are atomic units, so $m_e = \hbar = e = 1$. The first and second term describe the kinetic energy of the electrons and the nuclei, respectively. The third term describes the attraction between electrons and nuclei and the fourth and fifth term describe the repulsion between the electrons and the nuclei, respectively.

Relativistic methods are needed for systems where electrons interact with very massive nuclei (for example the lanthanides and actinides), because then electrons can move at speeds high enough that special relativity needs to be taken into account [27]. Unless very high accuracy is needed, it is a reasonable approximation to neglect relativistic effects for elements up to krypton ($Z = 36$) [28, p. 82].

Most ab initio methods attempt to solve this problem using the Born-Oppenheimer approximation, which uses the fact that the electrons are moving much faster than the heavy nuclei. This means that it is a good approximation to assume the Schrödinger equation can first be solved for the electrons while assuming nuclei are stationary, and then be solved for the nuclei using the average location of the electrons. This is only a good approximation when the nuclei are much heavier than the electrons, are moving much slower than the electrons, the system is adiabatic, quantum corrections such as tunnelling are negligible [28, p. 463] and the electronic ground state and the excited states are far apart [p. 82]jensen2007introduction. 'Adiabatic' in this case means that the configuration of the nuclei changes slowly enough that the electronic states will follow their eigenstate along the trajectory[1].

The magnitude of corrections needed to the Born-Oppenheimer approach depends on the rate of change of the electronic wavefunction when the nuclear configuration is changed. For some configurations of the nuclei, the potential energy surfaces (these energies are eigenvalues of the Hamiltonian) of the electronic states cross [29]. Such a crossing happens in situations where the configuration of the nuclei has certain kinds of symmetry that lead to multiple states with the same energy (the states are degenerate), but can also occur in configurations where there is no symmetry [30]. These crossings become relevant when looking at processes such as dissociation, or when the dynamics start with an excited electronic state [31], or when there is strong mixing between electronic and vibrational modes [32], or when the energy gap between the electronic ground state and the electronic excited state is small or zero (such as in metals).

There are ab initio methods that can correct for deviations resulting from the Born-Oppenheimer approximation and also (usually more computationally expensive) methods that involve no Born-Oppenheimer at all and solve for the nuclear and electronic states at the same time, such as quantum-mechanical wave packet calculations [33].

Because in the Born-Oppenheimer approximation the nuclei are assumed to be stationary when solving for the electronic wave function, the second term in (2.2) (the kinetic energy of the nuclei) can be neglected, and the nuclear repulsion energy (the last term in (2.2)) can be considered a constant [26]. In an eigenvalue problem like this, it means the constant only shifts the eigenvalues (the energy) and can therefore be neglected while solving the problem and be added back in later on. The Hamiltonian of the N electrons in a field of M nuclei that are considered point charges is therefore:

$$\mathcal{H}_{\text{elec}} = -\sum_{i=1}^{N} \frac{1}{2}\boldsymbol{\nabla}_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M} \frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j>i}^{N} \frac{1}{r_{ij}}. \tag{2.3}$$

One can obtain the electronic wave functions Φ_{elec} and the $\mathcal{E}_{\text{elec}}$ by solving the Schrödinger equation with this Hamiltonian,

$$\mathcal{H}_{\text{elec}}\Phi_{\text{elec}} = \mathcal{E}_{\text{elec}}\Phi_{\text{elec}}. \tag{2.4}$$

Adding the constant nuclear repulsion back in, it means the total energy is

$$\mathcal{E}_{tot} = \mathcal{E}_{elec} + \sum_{A=1}^{M}\sum_{B>A}^{M} \frac{Z_A Z_B}{R_{AB}}. \tag{2.5}$$

[1]Note that the meaning of the word 'adiabatic' is different in quantum mechanics compared to macroscopic thermodynamics. In macroscopic thermodynamics it refers to a process in which there is no heat transfer from the system to its surroundings. In quantum mechanics however, it refers to a process that occurs slowly enough that the shapes of the eigenfunctions change slowly and continuously.

After solving for the electronic wave functions and energies, one can solve for the motion of the nuclei, assuming electrons move so fast their average position can be used. This means that essentially, the nuclei move on a potential energy surface that depends on the solution of the electronic Schrödinger equation. Describing the motion of the nuclei means describing the vibration, rotation and translation of a molecule. It is possible to solve the nuclear Schrödinger equation for the motion of the nuclei, but in this report a quasiclassical approach is used and the motion of the nuclei is determined classically.

Solving for the motion of the nuclei classically is a good approximation if quantum effects are not important and the particles are fairly massive, so they behave like classical particles [28, p. 461]. For the CO_2–CO_2 system this is a better approximation than for the H_2–H_2 system, but even the hydrogen nucleus is still more than 1800 times more massive than the electron.

2.1.2 Spin-orbitals, Hartree products, Slater determinants

Ab initio methods attempt to solve (2.3) and (2.4) by constructing an electronic wave function from a set of orthogonal basis functions $\phi_k(\boldsymbol{x})$. These basis functions are one-electron spin-orbitals (wave functions with both a spatial and a spin component). The spin-orbitals consist of a spin part and a spatial orbital $\psi_k(\boldsymbol{r})$. The coordinates $\boldsymbol{x}$ include the Cartesian coordinates $\boldsymbol{r}$ of the electron as well as a spin coordinate ω. Since the spin can only have two values, this means that if the number of spatial orbitals used for the basis set is K, there are $2K$ basis functions in the basis set. These basis set functions are used to construct one-electron spin orbitals $\chi(\boldsymbol{x}_1)$. Generally, more basis functions means a higher accuracy, but also a more computationally expensive calculation.

If the electron-electron interaction (the last term in (2.3)) is neglected or considered a constant, then the electronic Hamiltonian can be factorized into an operator $h(i)$ for each electron i

$$\mathcal{H} = \sum_{i=1}^{N} h(i). \tag{2.6}$$

The wave function for N electrons, with the coordinates for electron i given by $\boldsymbol{x}_i$ is then

$$\Psi^{HP}(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N) = \chi(\boldsymbol{x}_1)\chi(\boldsymbol{x}_2)...\chi(\boldsymbol{x}_N), \tag{2.7}$$

which is called a Hartree product.

However, the Pauli exclusion principle means that the following requirement, called the antisymmetry principle, must be obeyed by the electronic wave function: when any two electrons i and j are exchanged (both their location and their spin), the wave function changes its sign,

$$\Psi(\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_j, ..., \boldsymbol{x}_N) = -\Psi(\boldsymbol{x}_1, ..., \boldsymbol{x}_j, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_N). \tag{2.8}$$

A Hartree product does not obey this requirement, but it can be enforced by using Slater determinants. If there are only two electrons, one can ensure the antisymmetry principle is obeyed by combining two Hartree products as follows:

$$\Psi(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{\sqrt{2}}(\chi_i(\boldsymbol{x}_1)\chi_j(\boldsymbol{x}_2) - \chi_j(\boldsymbol{x}_1)\chi_i(\boldsymbol{x}_2))). \tag{2.9}$$

If $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are exchanged, it results in the same wave function except for a sign change. The generalization of this to N electrons is the normalized Slater determinant

$$\Psi(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_i(\boldsymbol{x}_1) & \chi_j(\boldsymbol{x}_1) & \cdots & \chi_k(\boldsymbol{x}_1) \\ \chi_i(\boldsymbol{x}_2) & \chi_j(\boldsymbol{x}_2) & \cdots & \chi_k(\boldsymbol{x}_2) \\ \vdots & \vdots & & \vdots \\ \chi_i(\boldsymbol{x}_1) & \chi_j(\boldsymbol{x}_2) & \cdots & \chi_k(\boldsymbol{x}_N) \end{vmatrix}, \tag{2.10}$$

which uses the fact that when two rows of a matrix are switched (which corresponds to switching two electrons), the determinant changes sign. The notation for the normalized Slater determinant is

$$\Psi(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N) = |\chi_i \chi_j \cdots \chi_k\rangle \,, \tag{2.11}$$

with the order of the electron labels always in order $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$. The Slater determinant introduces exchange correlation, which means the location of an electron is influenced by electrons with the same spin. However, electrons with opposite spin still do not influence each other at all.

2.1.3 Hartree-Fock method

The Hartree-Fock method (HF), also called the self-consistent field method (SCF) is one of the most basic and cheapest ab initio methods. It attempts to approximate the solution of the electronic Schrödinger equation for the ground state with just one Slater determinant,

$$|\Psi_0\rangle = |\chi_1 \chi_2 \cdots \chi_N\rangle \,. \tag{2.12}$$

The goal is to find the best one-electron spin orbitals χ_i to use for this Slater determinant. The spatial part of these spin orbitals are linear combinations of the basis set functions ϕ_i,

$$\psi(\boldsymbol{r}_i) = \sum_{m=1}^{K} C_{mi} \phi_m(\boldsymbol{r}_i). \tag{2.13}$$

The best one-electron spin orbitals are the ones that lead to the lowest energy

$$E_0 = \langle \Psi_0 | \, \mathcal{H} \, | \Psi_0 \rangle \,, \tag{2.14}$$

and they are called the Hartree-Fock molecular orbitals. These Hartree-Fock molecular orbitals χ_i can be found by solving the Hartree-Fock equation for every electron

$$f(i)\chi(\mathbf{x}_i) = \varepsilon \chi(\mathbf{x}_i), \tag{2.15}$$

with $f(i)$ the Fock operator

$$f(i) = -\frac{1}{2}\boldsymbol{\nabla}_i^2 - \sum_{A=1}^{M} \frac{Z_A}{r_{iA}} + v^{HF}(i), \tag{2.16}$$

with $v^{HF}(i)$ the average potential experienced by electron i from the other electrons. The electron-electron repulsion is approximated by the average potential an electron experiences due to the presence of the other electrons. This means Hartree-Fock does not include all electron correlation, only the exchange correlation that the Slater determinant introduces.

Since $v^{HF}(i)$ depends on the Hartree-Fock molecular orbitals of the other electrons, (2.15) is non-linear and needs to be solved iteratively, by making an initial guess for the spin orbitals χ_i, then using those to calculate the average field $v^{HF}(i)$ and using this to solve for updated χ_i, which can then be used to calculate the updated $v^{HF}(i)$ et cetera. Each Fock operator $f(i)$ has an infinite number of eigenfunctions $\chi(\mathbf{x}_i)$ [26, p.123], but when using a set of basis functions, this is reduced to $2K$ eigenfunctions χ_i.

From spin orbitals χ_i, the N with the lowest energies are used for the Slater determinant that is the Hartree-Fock ground state. These N functions are the occupied spin orbitals χ_a, and the other $2K - N$ spin orbitals are the virtual spin orbitals χ_r. A larger basis set will result in a better approximation of the ground state and in a lower energy that converges to a limit called the Hartree-Fock limit.

The Hartree-Fock method is a variational method, which is any quantum mechanics method that uses the variational principle to find the state with the lowest energy. It involves optimizing the parameters in a

function to obtain the lowest possible expectation value of the energy. The result is then an upper bound to the ground state energy [34, p. 256].

The Hartree-Fock ground state is not the only possible Slater determinant; the total number of Slater determinants that could be formed from the $2K$ spin orbitals is

$$\binom{2K}{N} = \frac{(2K)!}{N!(2K-N)!}, \tag{2.17}$$

which is usually a really big number. These other determinants are classified depending on how they differ from the Hartree-Fock ground state. If they differ in only 1 spin orbital, it is a singly excited determinant, if they differ in 2, it is a doubly excited determinant, et cetera. They are written like this:

$$\begin{aligned} |\Psi_a^r\rangle &= |\chi_1\chi_2\cdots\chi_r\chi_b\cdots\chi_N\rangle \text{ (singly excited)}, \\ |\Psi_{ab}^{rs}\rangle &= |\chi_1\chi_2\cdots\chi_r\chi_s\cdots\chi_N\rangle \text{ (doubly excited)}, \\ &\text{et cetera.} \end{aligned} \tag{2.18}$$

In the first case, spin orbital χ_a was replaced with χ_r and in the second case χ_b was replaced with χ_s as well. These determinants can be considered an approximation to the excited states of the system, but the higher the state, the less accurate this approximation is. The excited determinants can be used as basis functions for more complicated and more accurate ab initio methods.

A big problem with the Hartree-Fock method is that it often is very inaccurate when describing dissociation. For example, even with very large basis sets the Hartree-Fock gives a wrong dissociation of H_2; it dissociates H_2 into H^+ and H^- instead of in two H atoms [35]. This makes the dissociation energy much too high.

2.1.4 Configuration interaction

There are several methods that attempt to improve on the Hartree-Fock method. One of those is CI (configuration interaction), which attempts to improve the results by using a linear combination of Slater determinants

$$|\Phi\rangle = c_0|\Psi_0\rangle + \sum_{ra} c_a^r|\Psi_a^r\rangle + \sum_{\substack{a<b\\r<s}} c_{ab}^{rs}|\Psi_{ab}^{rs}\rangle + \sum_{\substack{a<b<c\\r<s<t}} c_{abc}^{rst}|\Psi_{abc}^{rst}\rangle + ... \tag{2.19}$$

The number of Slater determinants $|\Psi_i\rangle$ is $\binom{2K}{N}$ and this will often be a lot. Full CI uses all these Slater determinants and is the best you can do within the non-relativistic Born-Oppenheimer approximation for a given 1-electron basis set [26]. For H_2–H_2 this is possible, but for CO_2–CO_2 it is prohibitively expensive.

The expansion coefficients for the ground state and excited states can be determined by calculating the eigenvectors of the Hamiltonian matrix, and the energies of the states are the eigenvalues. The Hamiltonian matrix has elements

$$\langle\Psi_i|\mathcal{H}|\Psi_j\rangle. \tag{2.20}$$

The difference between this result and the Hartree-Fock energy is the electron correlation energy. This correlation energy depends only on the coefficients of the double excitations,

$$E_{\text{corr}} = \sum_{\substack{c<d\\t<u}} c_{cd}^{tu}\langle\Psi_0|\mathcal{H}|\Psi_{cd}^{tu}\rangle, \tag{2.21}$$

because the ground state cannot mix with the single excitations or triple or higher excitations. In fact, any state cannot mix with states with which it differs by more than two spin orbitals [26, p.235]. Figure 2.1 also shows this coupling hierarchy.

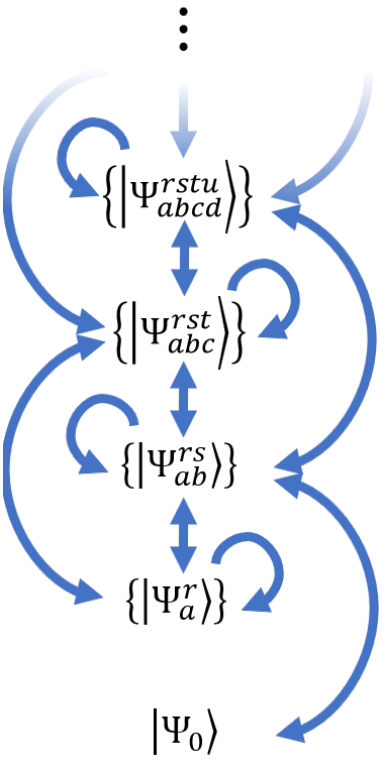


Figure 2.1: Diagram of the states that couple. Only states that differ by at most 2 spin orbitals can couple. Excited states also couple to other states with the same number of excitations. The ground state $|\Psi_0\rangle$ only couples to doubly excited states.

This means the double excitations will often play a predominant role in determining the correlation energy, but this does not mean that other excitations do not play a role at all, since the coefficients $\{c_{cd}^{tu}\}$ are affected by other excitations [26, p.238].

Unfortunately, the full CI method scales factorially with the number of basis functions and for larger systems it becomes intractable, so usually not all determinants are used. Instead the expansion is truncated at some excitation level. The simplest version of CI is therefore CISD (singly and doubly excited CI), which considers singly and doubly excited Slater determinants.

There are other ways of modifying full CI than just truncating it. For example, MCSCF (multi-configurational self-consistent field) uses the variation principle to determine not just the expansion coefficients, but also the orbitals used in the CI expansion [26, p.258].

Unfortunately, all forms of truncated CI are not size consistent; for example, a system of two molecules that are such a large distance apart that their interaction is negligible should have an energy that is twice the energy of only one of the molecules. The energy of the whole system should become proportional to the number of atoms as the number of atoms goes to infinity. The Hartree-Fock method and the full CI method have this property, but truncated CI does not [26, p. 261]. The discrepancy only gets bigger as the number of atoms increases, and as it goes to infinity the correlation energy vanishes completely. This makes CI unsuitable for large systems such as crystals, or situations where molecules are broken up, because in that case it is very important that the energy of the two fragments sums to the energy of the entire molecule. If CI includes quadruple excitations, it is fairly accurate for a system with up to (approximately) 50 electrons [26, p. 265], although in that case calculation times become exceedingly long.

2.1.5 Coupled-Cluster

Some other ab initio methods take a different approach. Figure 2.1 shows the coupling hierarchy. The problem is that to solve for the coefficients c_{ab}^{cd} of the doubly excited states, we need the coefficients of all the other states, which is intractable, so this hierarchy needs to be cut or decoupled somehow. The CISD method essentially just sets all coefficients of triply or higher excited states to zero, but the Coupled-Cluster method uses the fact that

$$c_{abcd}^{rstu} = c_{ab}^{rs} * c_{cd}^{tu} \equiv c_{ab}^{rs} c_{cd}^{tu} - \left\langle c_{ab}^{rs} * c_{cd}^{tu} \right\rangle, \tag{2.22}$$

when the ab and cd electrons are independent, and uses this as an approximation even when those electrons are not independent. This is not a simple product, since there are 18 combinations of coefficients of doubly excited states that can result in c_{abcd}^{rstu}[2], so instead $c_{ab}^{rs} * c_{cd}^{tu}$ is a sum of all these combinations, with the appropriate plus or minus sign, which depends on the antisymmetry property [26, p.286].

The Coupled-Cluster approach starts with the ansatz

$$|\Phi_0\rangle = e^{\mathcal{T}_2} |\Psi_0\rangle, \tag{2.23a}$$

$$\text{with } \mathcal{T}_2 = \frac{1}{4} \sum_{abrs} c_{ab}^{rs} a_r^\dagger a_s^\dagger a_b a_a. \tag{2.23b}$$

This representation uses the second quantization formalism, with $a_i^\dagger$ and a_i creation and annihilation operators, respectively, that excite and de-excite an electron in state i. A Taylor expansion of $e^{\mathcal{T}_2}$ then results in

$$|\Phi_0\rangle = c_0 |\Psi_0\rangle + \sum_{\substack{a<b \\ r<s}} c_{ab}^{rs} |\Psi_{ab}^{rs}\rangle + \sum_{\substack{a<b<c<d \\ r<s<t<u}} c_{ab}^{rs} * c_{cd}^{tu} \left|\Psi_{abcd}^{rstu}\right\rangle + ..., \tag{2.24}$$

[2]There are 3 ways to divide the indices $abcd$ into two pairs ((ab, cd), (ac, bd) and (ad, bc)) and similarly there are 3 ways to divide the indices $rstu$ into two pairs. Then there are two ways to combine the pairs (for example $c_{ab}^{rs} c_{cd}^{tu}$ and $c_{ab}^{tu} c_{cd}^{rs}$), which makes for $3 \cdot 3 \cdot 2 = 18$ combinations in total.

which, when inserted into

$$(\mathcal{H} - E_0)\,|\Phi_0\rangle = E_{\text{corr}}\,|\Phi_0\rangle\,. \tag{2.25}$$

and multiplied with $|\Psi_0\rangle$ and each of the states $|\Psi_{ab}^{rs}\rangle$, gives

$$\sum_{\substack{c<d\\t<u}} \langle\Psi_0|\,\mathcal{H}\,\big|\Psi_{cd}^{tu}\big\rangle\, c_{cd}^{tu} = E_{\text{corr}}. \tag{2.26}$$

and

$$\begin{aligned}\langle\Psi_{ab}^{rs}|\,\mathcal{H}\,|\Psi_0\rangle + \sum_{\substack{c<d\\t<u}} \langle\Psi_{ab}^{rs}|\,\mathcal{H} - E_0\,\big|\Psi_{cd}^{tu}\big\rangle\, c_{cd}^{tu}\\ + \sum_{\substack{c<d\\t<u}} \langle\Psi_0|\,\mathcal{H}\,|\Psi_{ab}^{rs}\rangle\,\big\langle c_{ab}^{rs} * c_{cd}^{tu}\big\rangle = 0.\end{aligned} \tag{2.27}$$

Here the fact that $\langle\Psi|\Psi'\rangle = 1$ if and only if $\Psi = \Psi'$ and else zero, the fact that $\langle\Psi|\,\mathcal{H}\,|\Psi'\rangle = 0$ if Ψ and Ψ' differ by more than two in their number of excitations and the fact that $\langle\Psi|\,\mathcal{H}\,|\Psi\rangle = E_0$ have been used.

Equations (2.26) and (2.27) then form a closed set of equations that only involve the coefficients of the double excitations. These resulting equations combined with (2.22) are the equations of the Coupled-Cluster Approximation (CCA) that involve only double excitations (CCD).

This approach can be extended to include single excitations or triple or higher excitations as well. Adding the single excitations (resulting in the CCSD method) is achieved by replacing $\mathcal{T}_2$ in (2.23a) with $\mathcal{T}_1 + \mathcal{T}_2$, with

$$\mathcal{T}_1 = \sum_{ar} c_a^r a_r^\dagger a_a. \tag{2.28}$$

CCSD is not a variational method, so the resulting energy can be an underestimation or an overestimation, and therefore gives no strict upper limit on the energy. It is a really good method, but results are very basis set dependent and the computational cost scales with K^6, with K the number of spatial orbitals in the basis set. Often, a large basis set or basis set extrapolation is needed. The coupled-cluster equations contain products of coefficients, which means they are non-linear, which means a simple matrix diagonalization is not sufficient to solve them [26, p. 289]. See Figure 2.2 for an overview of the ab initio methods discussed in this and the previous section.

2.1.6 Basis sets

The basis set used for an ab initio method has a large influence on the accuracy of the result. Usually, the bigger the basis set the more accurate the results, but the more computationally expensive the calculation is, although the computational cost also depends on how fast it is to calculate integrals of the basis sets. When the basis set gets bigger, the number of unoccupied orbitals increases and therefore the number of possible excited states increases.

There are 2 kinds of basis sets: Slater type orbitals and Gaussian type orbitals. Slater type orbitals are of the form

$$\chi_{\xi,n,l,m}(r,\theta,\phi) = NY_{l,m}(\theta,\phi)r^{n-1}e^{-\zeta r}, \tag{2.29}$$

which means they are similar to the exact orbitals of the hydrogen atom. Here, N is a normalization constant and $Y_{l,m}$ are the spherical harmonics. Gaussian type orbitals are of the form

$$\chi_{\xi,n,l,m}(r,\theta,\phi) = NY_{l,m}(\theta,\phi)r^{2n-2-l}e^{-\zeta r^2}. \tag{2.30}$$

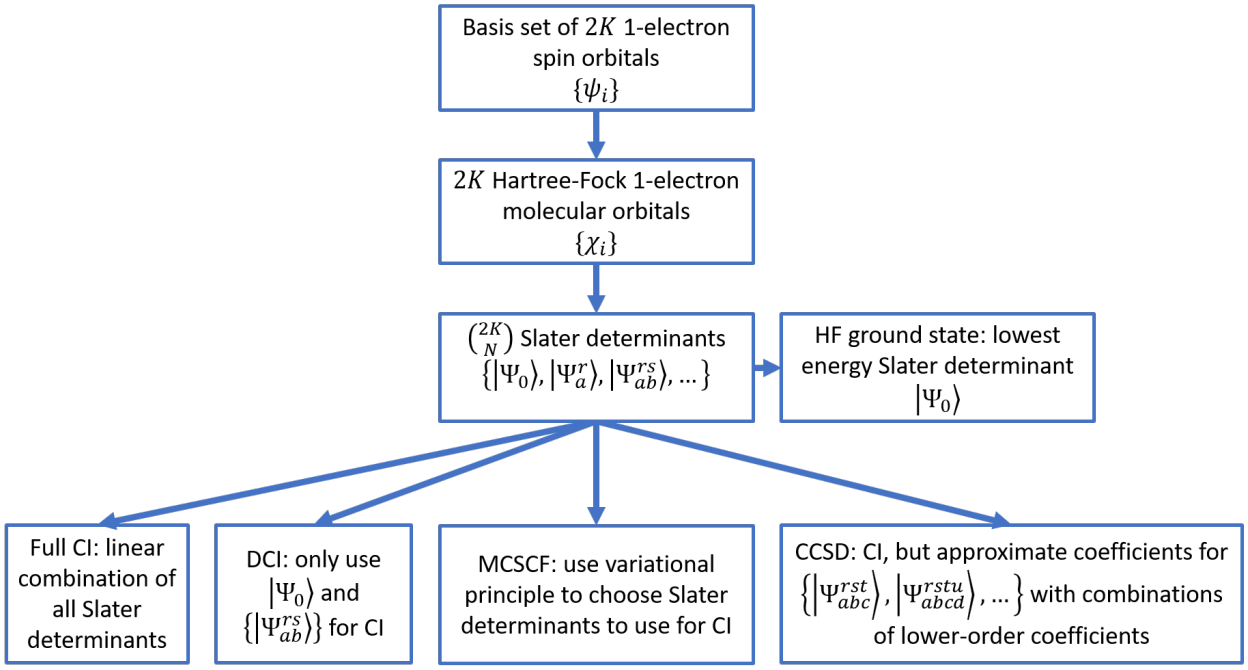


Figure 2.2: Diagram of the steps of the Hartree-Fock method, and how the full CI method, the DCI method and the CCSD method work.

The Gaussian type orbitals have a slope of zero near the nucleus (where r goes to zero), even though there should be a sharp peak there, which the Slater type orbitals do have. This means that Gaussian type orbitals are less accurate near the nucleus. They also have a tail that falls off much faster far away from the nucleus, which means they are also less accurate far away from the nucleus. This means there are usually more Gaussian type orbitals needed to get the same accuracy as Slater type orbitals (rule of thumb is that three times more basis functions are needed), but because integrals with Gaussian type orbitals can usually be calculated much faster [36], Gaussian type orbitals are still faster in most cases and are therefore the most commonly used basis sets.

Specifying a Gaussian type orbital means specifying its center, its exponent ζ and n and its spherical harmonic function (which is specified by l and m). Usually, the centers are chosen such that the basis functions are centered at the nuclei, but in some cases putting them in other places is convenient (for example, at the centre of a bond) [28, p. 192].

In general, it holds that basis functions with large powers ζ are good for the electron density at or near the nucleus [28, p. 199]. Diffuse basis functions (with small powers ζ) are more important in the tail of the wave function, when the atoms are separating [28, p. 200]. Inner electrons contribute more to the energy, but this contribution is very stable, since inner electrons are not strongly influenced by other electrons. Outer electrons are more important for chemical reactions, because they are more strongly affected by the electrons from the other atoms. The energy of a configuration of atoms is therefore not as important as the relative energies between different systems.

This is relevant for the error called the Basis Set Superposition Error (BSSE). This is caused by the fact that the atom-centered basis functions have different accuracies at different geometries, because when two atoms are close, the basis set of one atom can compensate for deficiencies in the basis set of another atom. Technically this makes the energy more accurate in the sense that it will be closer to the energy at the infinite basis set limit, but since the absolute energy is not of interest, it makes for a worse result, because the comparison to other geometries becomes worse. A way to approximate the correction for the BSSE is the counterpoise correction. In the case there are two parts A and B of a system between which there is interaction that suffers significantly from the BSSE, one needs to do 2 extra ab initio calculations for each part of the system (the parts between which there is interaction that suffers significantly from the BSSE; a part can be one atom but more commonly one molecule): one with only that part and only its own basis sets (a or b) resulting in $E(A)_a$ and $E(B)_b$, and one still with only that part but with the basis sets of all parts ab resulting in $E(A)_{ab}$ and $E(B)_{ab}$. The basis sets of atoms that are not present are called 'ghost orbitals'. The counterpoise correction is then

$$\Delta E_{CP} = E(A)_{ab} + E(B)_{ab} - E(A)_a + E(B)_b. \tag{2.31}$$

This is important when the energy differences of interest are very small, for example in the case of Van der Waals interaction and hydrogen bonds [28, p. 227].

Not all coefficients for the Gaussians are always varied; often only one coefficient is used for a fixed linear combination of Gaussians. This grouping Gaussian together is called basis set contraction, and is often used for the inner electrons, so not too much calculation time is wasted on those chemically unimportant electrons.

A lot of basis sets come in hierarchies of increasing size, where each set adds extra spin orbitals to the minimum basis set. A minimum basis set is a basis set with the smallest possible number of basis functions, which is equal to the number of electrons. This means there are exactly N occupied spin orbitals and zero unoccupied ones. The first set to improve on the minimal basis set is usually one that doubles the number of functions, which results in a Double Zeta (DZ) basis, so called because the letter zeta (ζ) is used as the exponent in (2.29) and (2.30). The next step could be to go to Triple Zeta, but this would result in an unbalanced basis set. A better addition is to add higher angular momentum functions, which are called polarization functions [26, 28, p.189], because they can describe the electron density when it falls

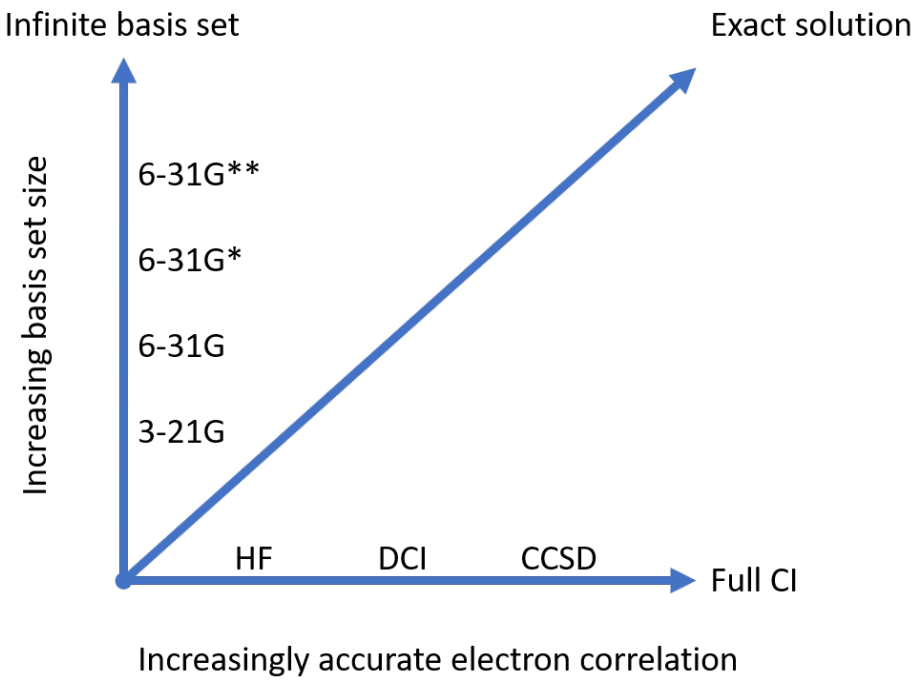


Figure 2.3: Pople diagram that shows the two factors – the basis set size and method used to determine the electron correlation energy – that determine the accuracy of an ab initio calculation (within the nonrelativistic Born-Oppenheimer approximation).

off at different rates in different directions, such as near a bond. These polarization effects are especially important if the ab initio method includes electron correlation.

One such hierarchy of basis sets is the Pople hierarchy of split-valence basis sets, that use more than one basis function for the valence electrons. Each orbital is composed of a number of Gaussians. 3-21G is the smallest basis set of this type [37]. Each of its basis functions for the core orbitals consists of 3 Gaussians, the inner part of the valence orbitals consists of 2 Gaussians and the outer part of 1. In the set 6-31G each core basis function consists of 6 Gaussians. By adding polarization functions to the 6-31G basis, one can get the 6-31G* and 6-31G** basis sets. The 6-31G* basis set contains polarization functions added to the heavy atoms and 6-31G** contains these as well as polarization functions added to hydrogen.

Another hierarchy of widely-used basis sets are the correlation-consistent basis sets by Dunning et al. [38], which were developed to have good convergence properties, and treat electron correlation in a consistent way. They are named cc-pVDZ, cc-pVTZ, cc-pVQZ, cc-pV5Z et cetera, where the D, T and Q stand for 'double', 'triple' and 'quadruple' while for 5 and higher, numbers are used. There also exist augmented versions of these basis sets, aug-cc-pVDZ, aug-cc-pVTZ, aug-cc-pVQZ et cetera, that have extra diffuse functions. This makes them better at describing the wave function far away from the nuclei [28, p. 207]. Often calculations with these correlation-consistent basis sets can be extrapolated fairly accurately to the complete basis set limit [39]. Figure 2.3 shows a Pople diagram, that shows how the basis set size and method used to determine the electron correlation energy determine the accuracy.

2.1.7 Potential H_2–H_2

The hydrogen molecule H_2 has only two electrons, which makes it the simplest possible neutral molecule to study with ab initio methods and it has been extensively studied. For those reasons, it is a good system for prototyping. The hydrogen nucleus (specifically the nucleus of Hydrogen-1, also called protium) consists of only one proton and its mass is therefore only approximately 1836 times that of an electron. This means using classical methods for the motion of the nuclei, or even separating the motion of the nuclei and electrons in the first place (the Born-Oppenheimer approximation) limits the accuracy. However, research by Carmona-Novillo et al. [40] shows that away from dissociation thresholds, the results of quasiclassical trajectories are still comparable to quantum-mechanical wave packet calculations that treat the entire molecule with quantum methods [33].

The bond between two hydrogen atoms is very strong, with a binding energy of 4.476 eV [41, p. 530], but the interaction between two molecules of hydrogen is very weak. The H_2–H_2 system only has a very tiny Van der Waals well (of about 0.3 meV [42]), because hydrogen has a very low polarizability, which means the electrons are very resistant to being redistributed. This means the London dispersion forces are very weak. As a result, two H_2 molecules can form a dimer only at very low temperatures. This dimer supports only one vibrational bound state [43]. Describing this properly would need ab initio methods with very high accuracy.

The H_2–H_2 PES has 5 local minima, which complicates fitting it [43]. The potential has multiple conical intersections between the ground state and the first excited state, where the Born-Oppenheimer approximation breaks down, because the energy of the ground state and the first excited state are too close together [44]. This includes one conical intersection that causes a cusp in the potential and is therefore very difficult to fit [45]. This occurs when the four atoms form a pyramid that has an equilateral triangle as a base (which means the fourth atom is exactly above the center of this base). However, this conical intersection can only be reached when the energy is high enough for H_2 to dissociate. In this report, the focus is on vibrational energy transfer, and dissociation is outside the scope of this report. However, effects of the conical intersection such as a geometric phase effect can affect state-to-state cross sections [46].

Several potential energy surfaces (PES) have been developed for the H_2–H_2 system, usually based on some kind of fit to ab initio data. The goal of these potentials is to reach at least chemical accuracy, which is how accurate a PES needs to be, to make realistic predictions for chemistry. Generally, this is assumed to be about 40 meV [28, p. 136]. This is also of the same order of magnitude as the thermal energy at room temperature (293.15 K), which is $E_{\text{th}} = k_B T = 25\,\text{meV}$, which is the accuracy we are aiming for in this report. This means that the very tiny Van der Waals well does not need to be reproduced exactly. To reach this kind of accuracy, the CCSD method with basis set aug-cc-pVTZ is used in this report. The counterpoise correction pretty small and was therefore not included.

One of the first of such PESs was published by Schwenke in 1988. It was a fit using spherical harmonics based on 85 ab initio energies, mostly in the H_2+H_2 repulsive wall [47]. It is only a 4D PES, because it assumes rigid molecules (the bond length fixed at the equilibrium distance), which is why it also called a 'rigid-rotor' PES.

Over the years, more of such 4D rigid-rotor PESs based on a spherical harmonics expansion fit to more and more accurate ab initio points were published [48, 49, 50, 51], as well as some that attempted to extend it to 6D, where the bond lengths were (slightly) variable [52, 45, 53]. Aguado, Suarez and Paniagua [8] also tried this, but with a polynomial expansion instead of a spherical harmonics expansion. One of the newest of such PESs is by Hinde et al. [53], which uses cubic splines (in the coordinates R, r_1, r_2) for the coefficients of the spherical harmonics expansion. This potential is mostly valid for low temperatures (ca. 40K), because it is meant to describe the bound state of H_2–H_2. It takes into account all 6 degrees of freedom, but only gives the intermolecular potential. For a complete potential, an intramolecular potential needs to be added to it. In this report the Hinde potential is combined with a fit of the H−H interaction of

our own of the form

$$
\begin{aligned}
E = & \frac{p_1}{r} + \frac{p_2}{r^2} + \frac{p_3}{r^3} + \frac{p_4}{r^{1.5}} \\
& + p_6 \exp\left[- \left(\frac{r - p_7}{p_8} \right)^2 \right] \\
& + p_9 \Big\{ \exp\big[-2p_{10}(r - p_{11}) \big] - 2 \exp\big[-p_{10}(r - p_{11}) \big] \Big\},
\end{aligned}
\tag{2.32}
$$

with p_i the fitting parameters that are determined by a non-linear least squares fit to ab initio data. The resulting potential is used to validate the results of the neural network potential.

2.1.8 Potential CO_2

Carbon dioxide (CO_2) is a molecule that is fairly well-studied, because it is common in the Earth's atmosphere and an important factor in global warming. It is a linear molecule made up of one central carbon atom with two oxygen atoms attached to it with a covalent double-bond each. Carbon and oxygen are both a lot heavier than hydrogen, so the Born-Oppenheimer approximation is more accurate for carbon dioxide. Energy transfer processes for CO_2 strongly depend on the vibrational and rotational state [6].

To reach accuracies in the order of 1×10^{-6} eV for CO_2, it is necessary to take into account relativistic effects as well as Born-Oppenheimer corrections [54, 55]. However, in this report a CO_2–CO_2 system (that of course contains twice as many electrons) is studied and since most ab initio methods scale badly with the number of electrons, an accuracy of that level was too expensive. For the purposes of this report, an accuracy of millielectronvolts is enough, since the thermal energy at room temperature is about 25 meV. This is also a reasonable accuracy for vibrational energy transfer in CO_2, because the separation between the lowest vibrations of CO_2 is about $700\,\mathrm{cm}^{-1}$, which is about 90 meV [56].

The bond-bond approach is a way of fitting a potential between two molecules that is based on summing the individual contributions of interactions of atoms from molecule 1 with atoms from molecule 2 [57, 58]. This also means the monomers do not need to have a fixed length. A way to get a reasonably accurate full-dimensional PES for CO_2–CO_2 is to add an intramolecular PES for CO_2 to the intermolecular bond-bond potential [6]. The potential developed by Murrell and Guo [59] and refined by Zúñiga et al. [60, 61] is one such CO_2 PES, which uses a many-body expansion. The combination of the bond-bond potential with the Murrell-Guo-Zúñiga potential is used to generate the initial trajectories to create the training data set, as well as to verify the neural network results.

2.2 Methodology

The ab initio calculations were executed using Dalton (versions 2016.2, 2018.0 and 2018.2), a molecular electronic structure program [62, 63]. At first, CCSD with basis set 6-31G** was used for H_2–H_2, later the more expensive basis set aug-cc-pVTZ was used for H_2–H_2, and for CO_2–CO_2 we settled for slightly lower accuracy to keep the calculation time in check and the basis set cc-pVTZ was used. Dalton will need two inputfiles: the .dal file which describes what needs to be calculated and with which method, and the .mol file, which describes the molecular configuration and the basis set. This is an example of a .dal file:

```
**DALTON INPUT
.RUN WAVE FUNCTIONS
**WAVE FUNCTIONS
.HF
.CC
```

```
*CC INPUT
.CCSD
.MAX IT
50
**END OF DALTON INPUT
```

The section **WAVE FUNCTIONS tells Dalton what kind of wave functions need to be calculated. .HF indicates a Hartree-Fock wavefunction and .CC indicates a coupled cluster wave function. The *CC INPUT then indicates the details for the coupled cluster calculation, in this case that it is a CCSD calculation and that at most 50 iterations should be performed. A value of 50 was chosen because this was enough iterations for any realistic geometry to converge. The *DERIVATIVES keyword tells Dalton to also calculate the analytical gradient of the potential.

This is another example of a .dal file, that also calculates the first excited state (with the coupled cluster calculation), which is relevant for CO_2:

```
**DALTON INPUT
.RUN WAVE FUNCTIONS
**WAVE FUNCTIONS
.HF
.CC
*CC INPUT
.CCSD
.MAX IT
200
*CCEXCI
.NCCEXCI
0
1
**END OF DALTON INPUT
```

The line with a zero specifies that we don't need any excited state with singlet excitations and the second line specifies that we do want a triplet excited state. This is an example of a .mol file for H_2–H_2:

```
BASIS
aug-cc-pVTZ
comment
comment
Atomtypes=1 Nosymmetry
Charge=1.0 Atoms=4
H -1.697 0.001689 -27.33
H 1.697 -0.001689 -26.68
H 0.2476 -0.2954 26.50
H -0.2476 0.2954 27.51
```

This is an example of a .mol file for CO_2–CO_2:

```
BASIS
6-31G**
comment
comment
Atomtypes=2 Nosymmetry
Charge=6 Atoms=2
C  0.004052  0.002325  27.513298
C -0.003962 -0.002334 -27.513045
Charge=8 Atoms=4
```

```
O   1.278   1.643 28.20
O  -1.278  -1.643 26.79
O 0.7150 -6.343e-05 -25.42
O -0.7150 6.766e-06 -29.58
```

The basis set that was used is specified at the beginning of the file. Then two lines are reserved for comments. Leaving these two lines out will cause Dalton to interpret the next two lines as comments, which will give an error, which is why there are two lines with a comment. The rest of the files specifies the charges of the nuclei and their configuration, which is given in Cartesian coordinates in bohr radii. The letter before the coordinates of an atom ('H', 'C' or 'O') are only interpreted as labels by Dalton. They could just as well have been 'H1', 'H2', et cetera. The keyword 'Nosymmetry' tells Dalton not to try to simplify the calculation by considering symmetries in the configuration. The simplification is not very useful in our case anyway, since the configurations are really unlikely to end up in such a symmetric configuration, and this symmetry analysis causes Dalton to reorder the atoms. This is a problem if Dalton needs to calculate the gradient of the potential as well, because in that case the gradient on the atoms can be in a different order than the Cartesian coordinates of the atoms. This can be fixed by using a unique label for each atom, but that does not solve another problem: the symmetry analysis can also cause a peak in the potential at configurations with certain symmetry, because it can make a more accurate calculation when it takes symmetry into account.

Dalton also needs a description of the basis set, but fortunately these are included with Dalton. An example of a basis set specification is this one, of 6-31G** for H:

```
$ HYDROGEN        (4s,1p) -> [2s,1p]
$ S-TYPE FUNCTIONS
    4    2    0
    18.73113700  0.03349460  0.00000000
     2.82539370  0.23472695  0.00000000
     0.64012170  0.81375733  0.00000000
     0.16127780  0.00000000  1.00000000
$ P-TYPE FUNCTIONS
    1    1    0
     1.10000000  1.00000000
```

The first line gives a summary of the basis set. In parentheses is the number of Gaussians used per type of function. The type of function (s-type and p-type in this case) refers to the spherical harmonic function $Y_{l,m}$ in (2.30). In square brackets is the number of basis set functions these Gaussians are contracted to. This means there are 2 s-type functions, which in total use 4 different Gaussians. There is only one p-type function, which consists of just one Gaussian.

The first row below each type of function gives the number of Gaussians and the number of basis functions made with these Gaussians again, as well as a zero with a meaning that Dalton does not bother to clarify in their manual, or anywhere else for that matter (other basis set formats include all other information but not this zero). The rest of the rows specify the basis functions as follows: the first column gives the ζ used in (2.30) in Å^{-2}, and each column after that is one basis function, and gives the contraction coefficients for the Gaussians. In this example that means that there is one s-type basis function that is a sum of three Gaussians, and one that is only one Gaussian. The contraction coefficients of the 3 Gaussians are constant, and the 3 Gaussians are optimized as a group with one coefficient.

Dalton can be run with the command
`dalton dalfile.dal molfile.mol`.
The output from Dalton is given in a .out file, that will appear in the working directory dalton was run from. This is an example of a part of such an output file (under 'Final results from SIRIUS') that shows the energy of the computed orbitals for a Hartree-Fock calculation with 4 H-atoms with the basis set 6-31G**:

Sym	Hartree−Fock orbital energies				
1 A	−0.56832269	−0.42225086	0.03566174	0.21504343	0.80776954
	1.01283520	1.05148215	1.23996694	1.96959607	1.96964932
	2.05510860	2.24017572	2.24019773	2.31865881	2.31868106
	2.52208699	2.64176275	2.82487148	2.82492588	4.14867146

The counterpoise correction for H_2 – H_2 was estimated and is really only relevant for the van der Waals well. The counterpoise correction gives a binding energy of $(H_2)_2$ (meaning the energy it takes to separate one $(H_2)_2$ complex into two H_2 molecules) that is 26% smaller than the binding energy without the correction for the aug-cc-pVTZ basis set and 8.3% smaller for the aug-cc-pVQZ basis set. For the 3-21G basis set the counterpoise correction does make a big difference, since without the correction there is no well at all. Since the well is already smaller (<5 meV) than the desired accuracy on par with the thermal energy (25 meV), the counterpoise correction was neglected.

Dalton gives the energy of a certain configuration for the nuclei, relative to the situation where all the nuclei are infinitely far apart, and completely ionized (meaning the electrons are also infinitely far away from all the nuclei). In this case it makes more sense to use the situation where the nuclei are infinitely far apart but not ionized, with their electrons in the ground state as a reference. To obtain this energy, the sum of the ionization energies for the atoms should be added to the dalton energy. Different basis sets will also have a different approximation of this value, so a different offset is used for each basis set. This offset is calculated as the potential energy for the situation where all the atoms are so far apart their interaction is negligible, and defining this to be zero. For the H_2–H_2 system this reference value was obtained by calculating the energy of two H-atoms 5.0 Å apart. The results are in Table 2.1. The refence value of H_2–H_2 system is twice the H–H value, since CCSD energies can be added. For the CO_2 system, the sum of the energy of each atom can be used. The ionization energy of two hydrogen atoms is $0.999\,946\,65\,E_\mathrm{h}$[64], so the result for aug-cc-pVQZ is very accurate.

For CO_2 this is complicated by the fact that as one oxygen atom dissociates from the molecule, the ground state curve crosses three triplet excited state curves in three different points. This means at some point a triplet state overtakes the singlet state as the lowest energy state. (The transition between these states is forbidden in the Born-Oppenheimer approximation, but possible because of spin-orbit coupling terms [65].) This is caused by the fact that the dissociation products (CO and O) have a singlet and a triplet ground state, respectively. For the initial exploration of the potential of CO_2 this triplet state at infinite separation was used as a reference. Depending on which lengths were being varied this can be the situation where one oxygen atom is infinitely far apart from the leftover CO molecule, or the situation where all three atoms are infinitely far apart. For the machine learning part we only train the network on singlet data, and the potential needed to go smoothly to zero, so there the singlet state at infinite separation of all three atoms was used as a reference, and this is what is shown in Table 2.1. It is important that symmetry was also not used when determining this energy offset, since it could be slightly different with symmetry .

Several Python scripts were used to automate the creation of .mol files, to run dalton, to import the results from the .out files and to make a graph from them. `H2.py` and `CO2.py` are used to explore the potential. The scripts to create a data set to use for the machine learning are discussed in Chapter 3.

CCSD is not very accurate for dissociation. Since we were mostly interested in the potential for vibrational energy transfer, this was not considered a problem. This means the ab initio calculations might not be very accurate in situations where the atoms of one molecule are approaching the dissociation limit.

2.3 Results H_2–H_2

The ab initio methods and analytical potentials were used to explore the PES of H_2 as well as H_2–H_2, to estimate which parameter ranges would be relevant. Figure 2.4 shows the potential between two hydrogen

Table 2.1: The energy in hartree defined to be zero for different basis sets.

	offset (E_h)	
	H_2	CO_2
Theoretical	-0.99994665	
3-21G		-186.1773720959
6-31G**	-0.9964669427	-187.2987100500
cc-pVDZ	-0.9985601387	-187.3408592819
cc-pVTZ	-0.9996255424	-187.5162972456
aug-cc-pVDZ	-0.9986803034	-187.3759377617
aug-cc-VTZ	-0.9996637739	-187.5293160648
aug-cc-VQZ	-0.9999098559	-187.6200372457

Table 2.2: Root-mean-square error (RMSE), mean absolute error (MAE) and the standard deviation of the error (σ), of our results compared to the results of Kołos & Wolniewicz.

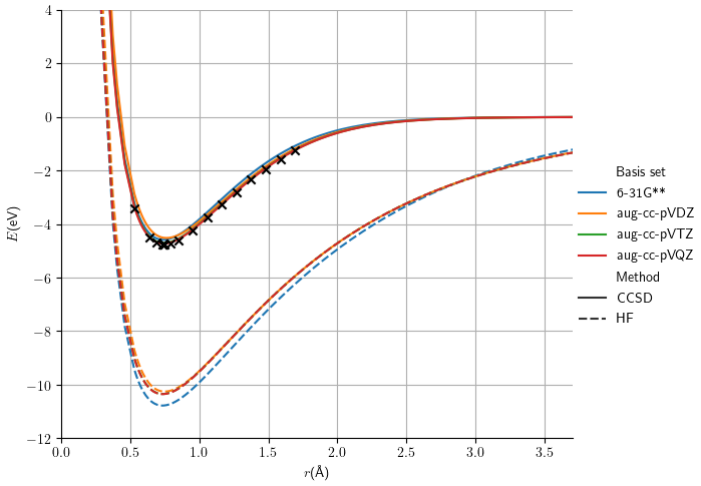
Basis set	RMSE (meV)	MAE (meV)	σ (meV)
6-31G** `(4s,1p) -> [2s,1p]`	275.6	274.7	22.4
aug-cc-pVDZ, `(5s,2p) -> [3s,2p]`	251.2	223.3	115.1
aug-cc-pVTZ, `(6s,3p,2d) -> [4s,3p,2d]`	45.6	43.6	13.2
aug-cc-pVQZ, `(7s,4p,3d,2f) -> [5s,4p,3d,2f]`	14.5	14.0	4.0

atoms. The results are compared to the very accurate ab initio results of Kołos & Wolniewicz [66, 67, 68]. The ab initio method for their results was specifically designed for H_2 and took advantage of the symmetries of the system. Their basis set contained 100 basis functions and the results included diagonal corrections for nuclear motion.

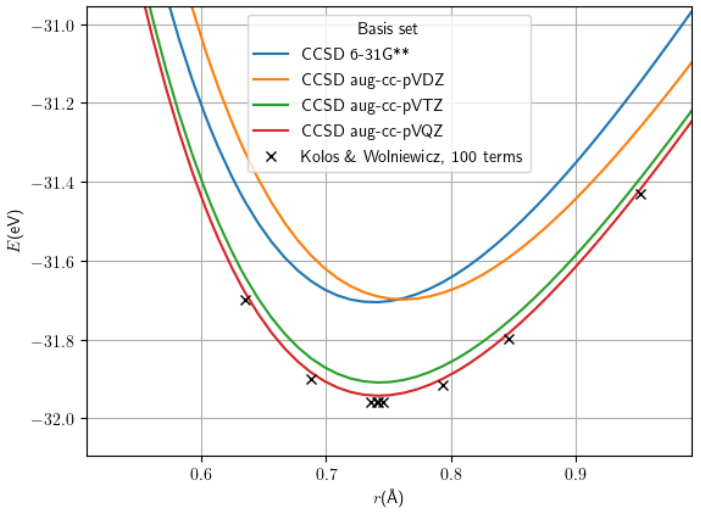
Figure 2.4 shows that the Hartree-Fock calculations are not very accurate, even with a very large basis set such as aug-cc-pVQZ. This is because the Hartree-Fock wave function contains a part that corresponds to a dissociation of H_2 into H^+ and H^- instead of in two H atoms[26, p. 167]. This makes the dissociation energy much too high.

Table 2.2 shows the difference between our results and the very accurate Kołos & Wolniewicz results. It shows a Mean Absolute Error of 43.6 meV with the basis set aug-cc-pVTZ, but this is in large part due to a systematic error, as shown by the standard deviation of this error, which is only 13.2 meV. This shows that in this region, which is the most important for low vibrational states, the error is already very small with the aug-cc-pVTZ basis set, and going to the larger basis set aug-cc-pVQZ only gives a small improvement in accuracy.

Figure 2.5 shows the potential between two parallel oriented H_2 molecules. It shows that the 6-31G** basis set is not good enough to accurately represent the Van der Waals-interaction between the molecules. The results for aug-cc-pVDZ, aug-cc-VTZ and aug-cc-pVQZ are very close together, suggesting they are close to convergence. Also note that the scale of this Van der Waals-well is a few millielectronvolts, which is smaller than the desired accuracy of the thermal energy at room temperature of 25 meV. Figure 2.7 shows the potential for the different configurations in Figure 2.6. It shows that in the T- and S45-configurations



(a)



(b) Zoomed in on the CCSD results, unnormalized.

Figure 2.4: Potential of one H_2 molecule, varying the bond length, for both CCSD and HF calculations with several different basis sets. The results are compared to the very accurate results of Kołos & Wolniewicz [67]. The results are normalized such that zero is at infinity.

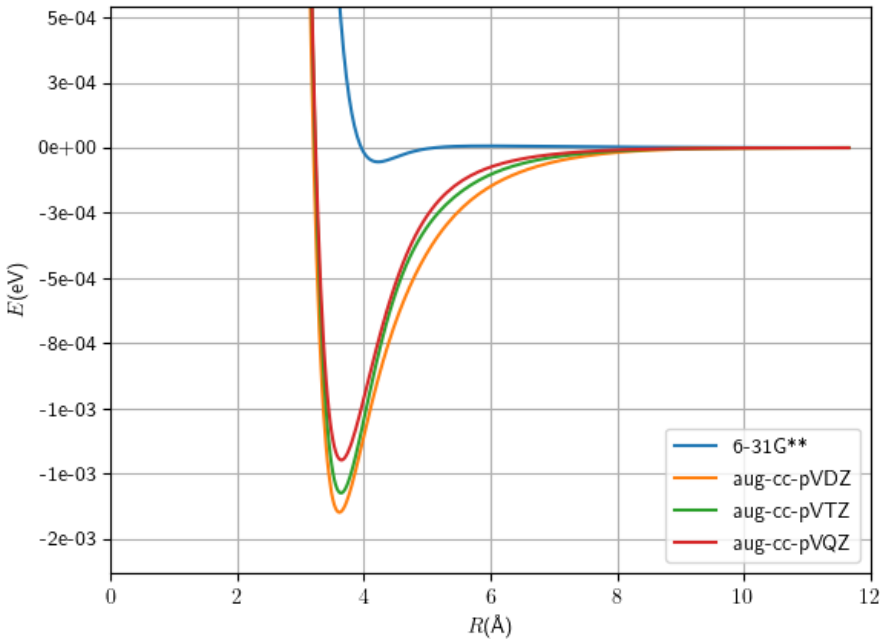


Figure 2.5: Potential of two (parallel oriented) H_2 molecules for various basis sets, varying the distance between them.

the interaction between molecules is the strongest and in the H- and L-configurations the weakest.

Table 2.3 shows approximately how long the calculations took for each basis set.

2.4 Results CO_2–CO_2

The ab initio methods and analytical potentials were used to explore the PES of CO_2 and CO_2–CO_2, to estimate which parameter ranges would be relevant.

First, calculations were done for various configurations of one CO_2 molecule. Figure 2.8 shows the potential as one bond length is kept constant and the other is varied. It shows that the basis set cc-pVTZ is approximately just as accurate as the Murrell-Guo-Zúñiga potential, and going to larger basis sets only gives small changes in the potential. As the molecule starts dissociating, the ground state is overtaken by the $^3B_2$ state. This excited state shows some peaks at r =1.163 Å, which is because Dalton uses this symmetry to make the calculation more efficient here. This results in a different accuracy. These spikes are obviously not a good thing when implementing a potential for the purpose of molecular dynamics, since a smooth potential is very important in that case. This is why the keyword `Nosymmetry` is used for creating a training dataset.

Figure 2.9 shows the potential as the internal angle ϕ is varied. It shows some interesting behavior between $\phi = \pi/4$ and $\phi = \pi/2$, where there is a sharp peak in the ground state and a sudden bend in the excited $^3B_2$ state. This is caused by an abrupt change in the wavefunction from a state where the two oxygen atoms repel each other to a state with a partial bond between them. A method that uses only one Slater determinant (such as Hartree-Fock and CCSD) is unsuitable for these configurations. To properly calculate the potential at these geometries, a multireference method is more suitable. See B for more detail.

Interestingly, the potential minimum of the excited $^3B_2$ state is not near $\phi = \pi$, which means CO_2 does not want to be a linear molecule in its excited state. The peak in the excited state near $\phi = \pi$ is probably again a result of Dalton taking the extra symmetry into account for that configuration, although

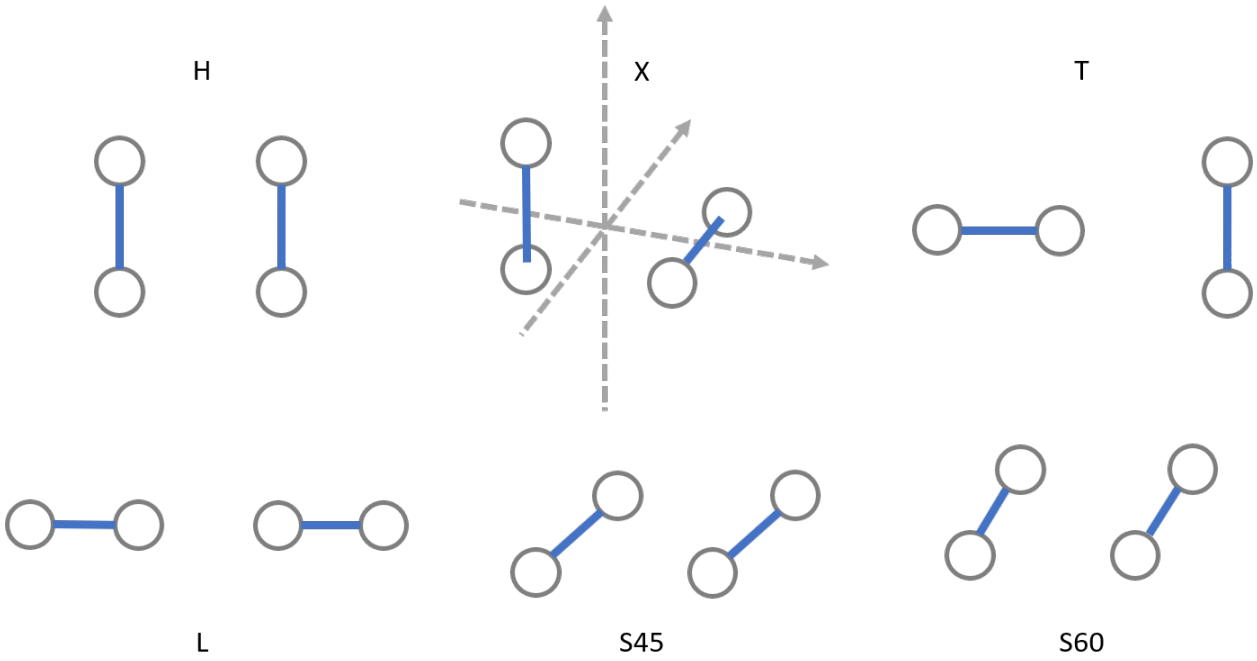


Figure 2.6: Six different configurations of two H_2 molecules.

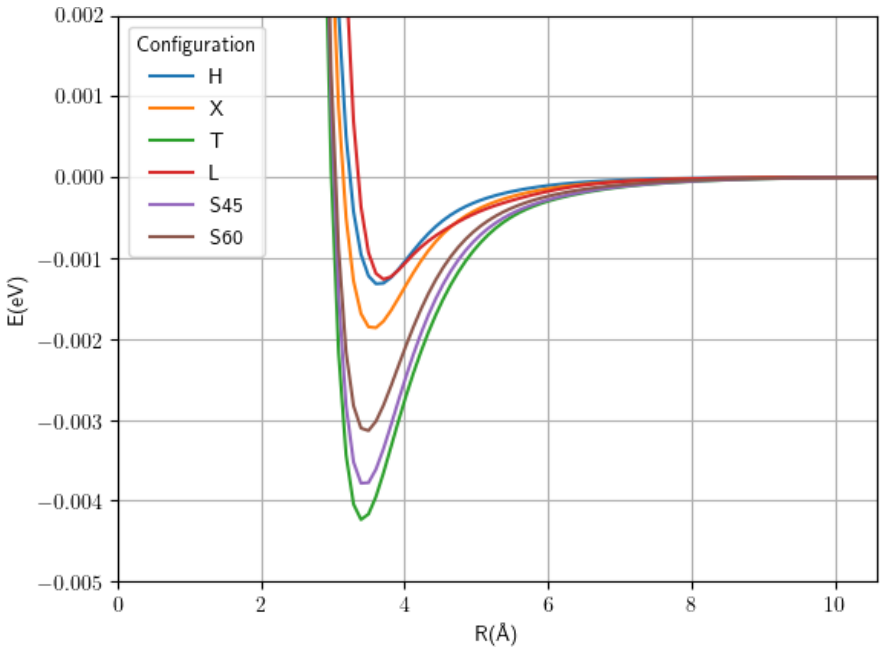


Figure 2.7: Potential of two H_2 molecules in the different configurations of Figure 2.6, for the aug-cc-pVTZ basis set, varying the distance between their centers of mass.

Table 2.3: Time in seconds it takes Dalton to do Hartree-Fock or CCSD calculations for H_2 and H_2–H_2 with various basis sets.

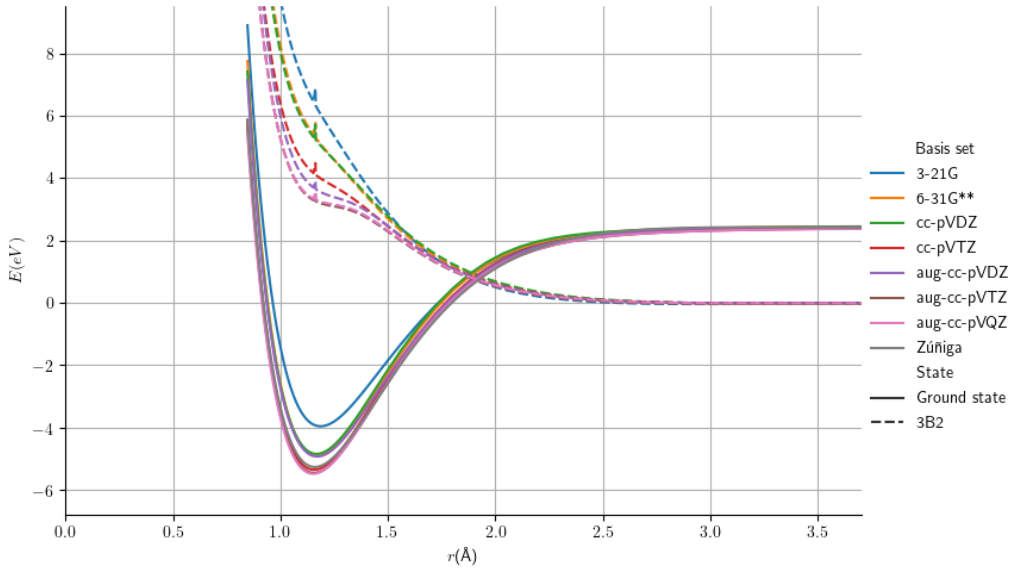
	Basis set	H_2	H_2–H_2
HF	6-31G**	0.002	0.005
	aug-cc-pVDZ	0.004	0.015
	aug-cc-pVTZ	0.023	0.324
	aug-cc-pVQZ	0.277	3.996
CCSD	6-31G**	0.02	0.06
	aug-cc-pVDZ	0.03	0.35
	aug-cc-pVTZ	0.35	11.0
	aug-cc-pVQZ	7.35	176

symmetry was turned off for these calculations, so it is unclear whether this is a physical phenomenon or not.

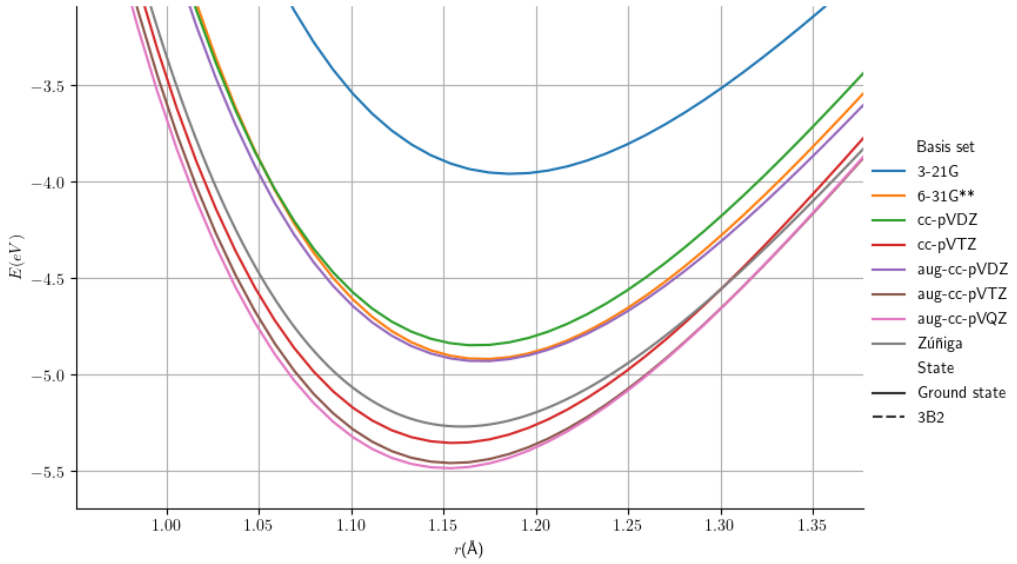
Figure 2.10 shows the potential of the ground state and the first excited state as the two C-O bond lengths are varied. The excited state shows some spikes on the diagonal (where $r_1 = r_2$), which is because of the symmetry.

Figure 2.11 shows the potential of two CO_2-molecules as the distance R between their centers of mass is varied, for the H- and X-configuration, which are similar to those configurations in Figure 2.6 (imagine a C-atom in the middle). It shows that the 3-21G basis set cannot model the van der Waals force at all, since the potential does not have a well in both configurations. For the X-configuration, the 6-31G**, the cc-pVTZ and the aug-cc-pVDZ basis set do show a well. For the H-configuration, aug-cc-pVDZ and aug-cc-pVTZ basis sets show a small well, but the 6-31G** set and, strangely enough, the aug-cc-pVQZ do not show a well here, but ab initio calculations by Bartolomei et al. show that the well for the H-configuration is either very small or not present at all as well [58, Fig. 1], so the aug-cc-pVDZ and aug-cc-pVTZ calculation might just not be good enough to properly describe this interaction. Fig 2.11 also illustrates that just like hydrogen, the interaction between CO_2-molecules can differ a lot depending on the orientation of the molecules with respect to each other.

Table 2.4 shows a very rough estimate of the time it takes Dalton to do Hartree-Fock or CCSD calculations for CO_2 and CO_2–CO_2. Keep in mind the fact that the computation time can vary a lot depending on the configuration; if the configuration of atoms is a realistic configuration with a low energy, the calculation takes fewer iterations to converge. The available hardware is also a big factor. The table shows that the scaling in computational cost is very unforgiving. For the bigger basis sets (aug-)cc-pVTZ and (aug-)cc-pVQZ the amount of memory available and the bandwidth to the disc space were concerns as well.

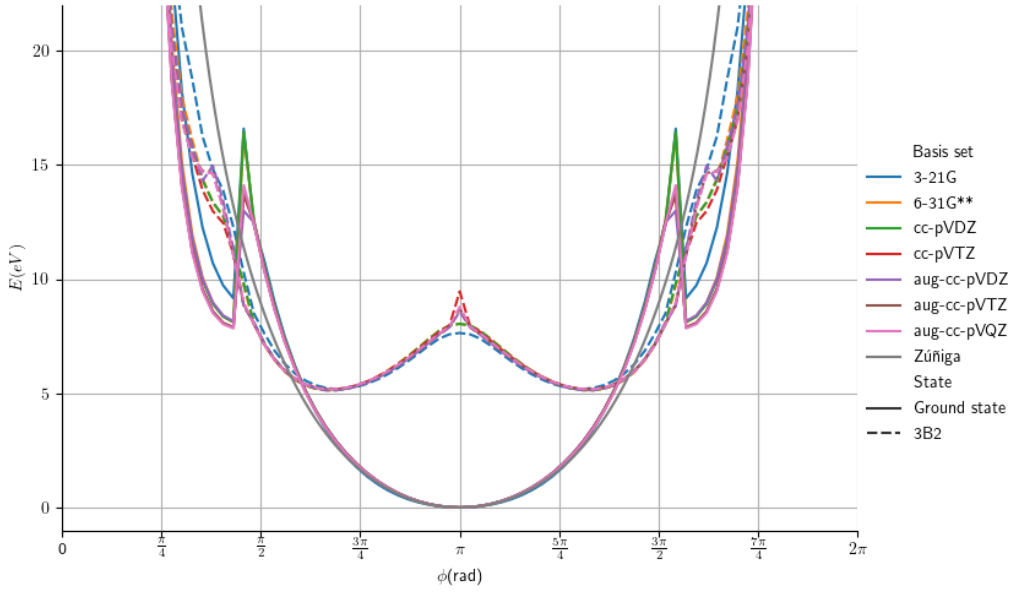


(a)

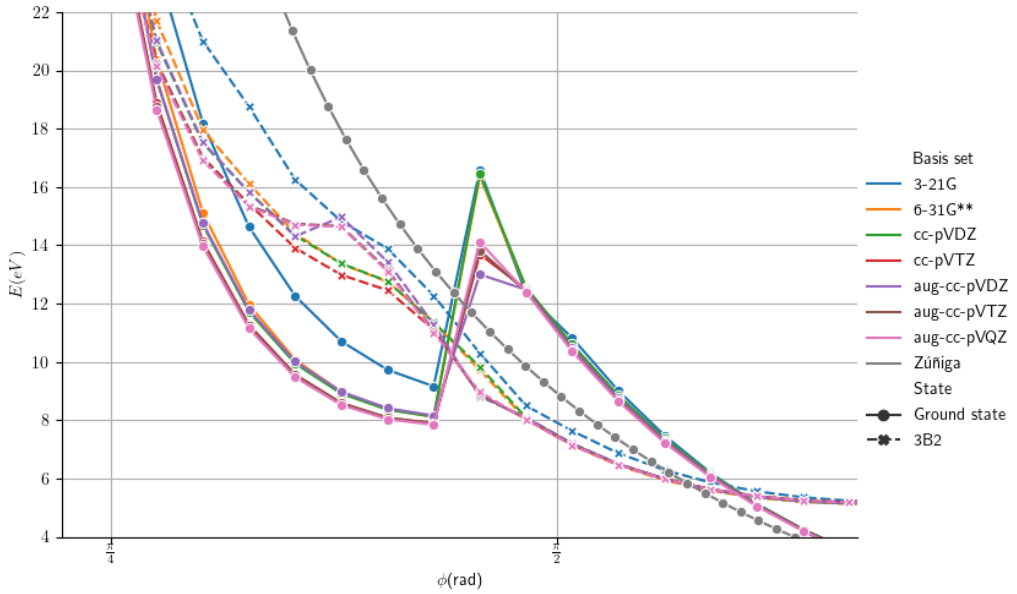


(b) Zoomed in.

Figure 2.8: Potential of one CO_2 molecule, varying one of the C-O bond lengths r. The other $C-O$ bond length is 1.163 Å *(the equilibrium bond length).*

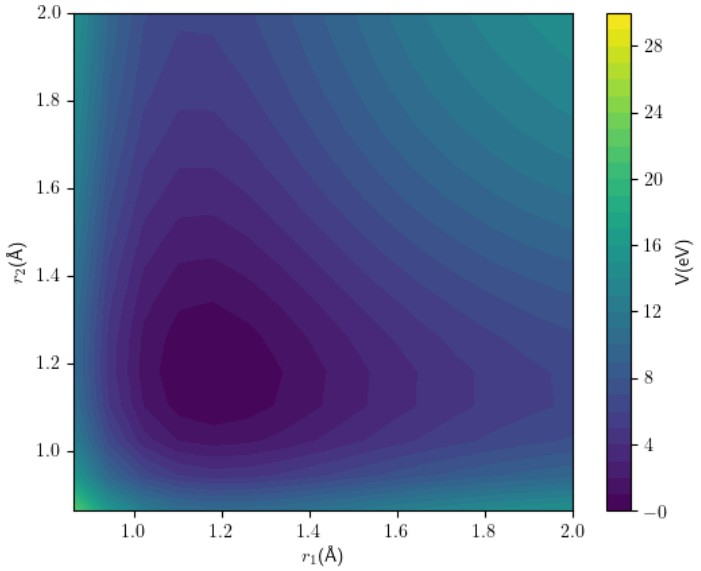


(a)

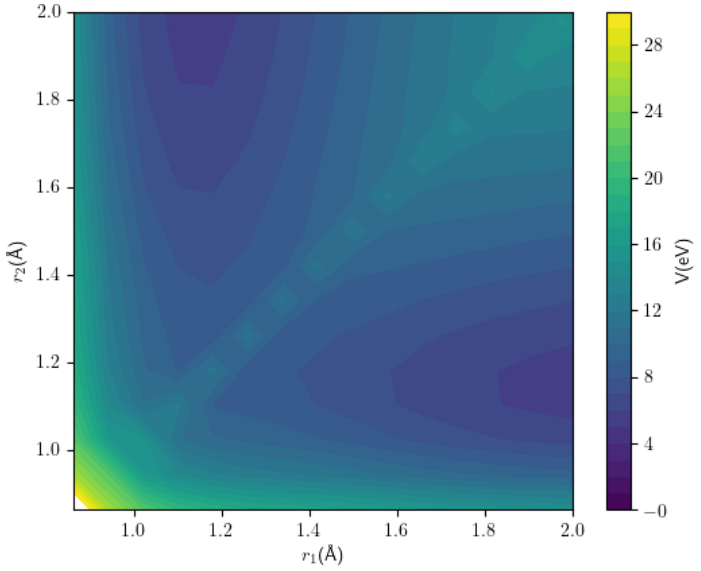


(b) Zoomed in.

Figure 2.9: Potential of one CO_2 molecule, varying the internal angle ϕ. The $C-O$ bond lengths are 1.163 Å*, which is the location of the potential minimum in Figure 2.8.*

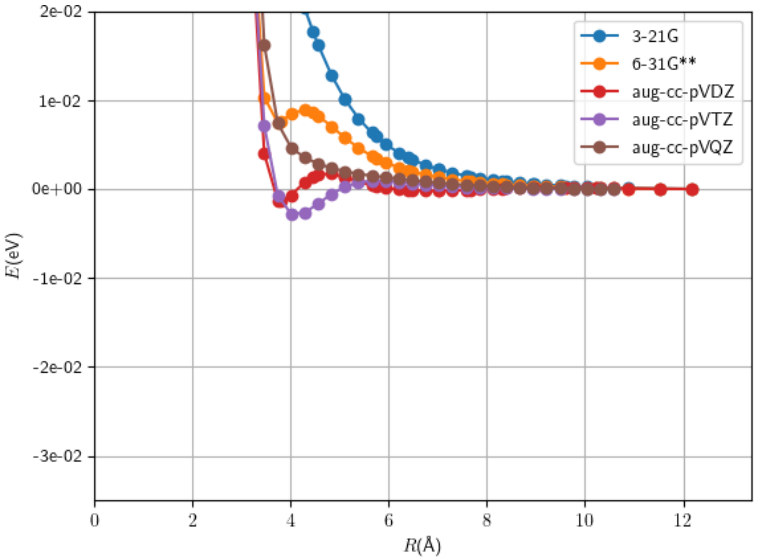


(a) The energy of the ground state (X)

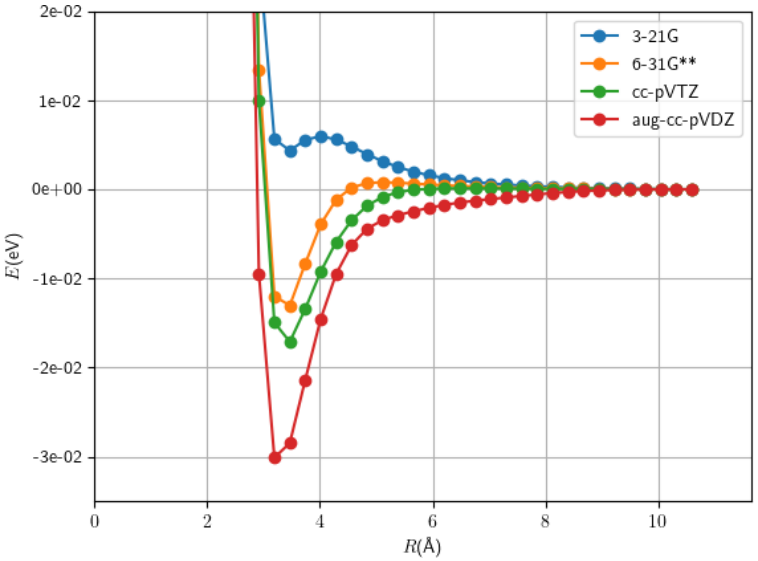


(b) The energy of the first excited state ($^3B_2$)

Figure 2.10: Potential of one CO_2 molecule, varying the lengths of both $C-O$ bond lengths r_1 and r_2.



(a) H-configuration



(b) X-configuration

Figure 2.11: Potential of two CO_2 molecules, varying the distance between their centers of mass R.

Table 2.4: Rough estimate of the time it takes Dalton to do Hartree-Fock or CCSD calculations for CO_2 and CO_2–CO_2 with various basis sets. [a] includes computation of excited state $^3B_2$, [b] Dalton used symmetry to make the calculation more efficient.

	Basis set	CO_2	CO_2–CO_2
HF	3-21G	0.013s[a]	0.12s
	6-31G**	0.040s[a]	0.59s
	cc-pVDZ	0.040s[a]	0.79s
	aug-cc-pVDZ	0.27s[a]	0.59s[b]
	cc-pVTZ	0.72[a]	19s
	aug-cc-pVTZ	3.5s[a]	
	aug-cc-pVQZ	26s[a]	
CCSD	3-21G	1.1s[a]	2m 2s
	6-31G**	5.3s[a]	13m
	cc-pVDZ	5.0s[a]	19m
	aug-cc-pVDZ	1m 30s[a]	8m 50s[b]
	cc-pVTZ	3m 10s[a]	3h 2m
	aug-cc-pVTZ	6m 50s[a]	
	aug-cc-pVQZ	1h 4m[a]	

Chapter 3

Machine Learning

3.1 Theory

Machine learning algorithms are used to build a model to analyze data or make predictions, using only training data, without giving explicit instructions. Modelling a PES is a supervised learning task, which is a task where there are examples of inputs (here: descriptions of atom configurations) and corresponding desired outputs (the energy) available.

One of the simplest and quickest machine learning techniques for supervised learning is a linear regression. One of the most popular machine learning techniques is an artificial neural network (NN), which takes inspiration from the structure of connections in the human brain. The variant relevant to PES fitting is a simple feedforward neural network, which can be used for function approximation (as opposed to recurrent neural networks, which have 'memory'). The advantage of a neural network is its flexibility.

The review article by Manzhos and Carrington gives a good overview of the current state of the research into using machine learning for potential energy surfaces [69].

3.1.1 Linear regression

A linear regression models a function as a linear combination of input variables

$$\text{output} = c_0 + c_1 x_1 + c_2 x_2 + ..., \tag{3.1}$$

with c_i the coefficients that need to be determined and x_i the inputs (x_0 is 1, which means c_0 is the constant term). This can also be written as a matrix equation

$$\mathbf{y}_{pred} = \mathbf{X}\mathbf{c}, \tag{3.2}$$

with $\mathbf{y}_{pred}$ the predicted output for each of the N data points, $\mathbf{X}$ an $N \times M$ matrix with M inputs for each data point, and $\mathbf{c}$ the M parameters that need to be found. Usually $M > N$, which means the system is overdetermined, meaning there are more equations than variables to solve for.

This equation is often solved with the least squares method, which minimizes the total of all squared deviations, $\sum_i (y_{i,pred} - y_{i,real})^2$. The solution is

$$\mathbf{c} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}_{real}. \tag{3.3}$$

The main advantage of this approach is its low computational cost. The main disadvantage is that the model is always a linear function, which means the only way to model a non-linear function is to create inputs that capture the non-linearity adequately. Coming up with such inputs often takes a lot of expertise and guesswork.

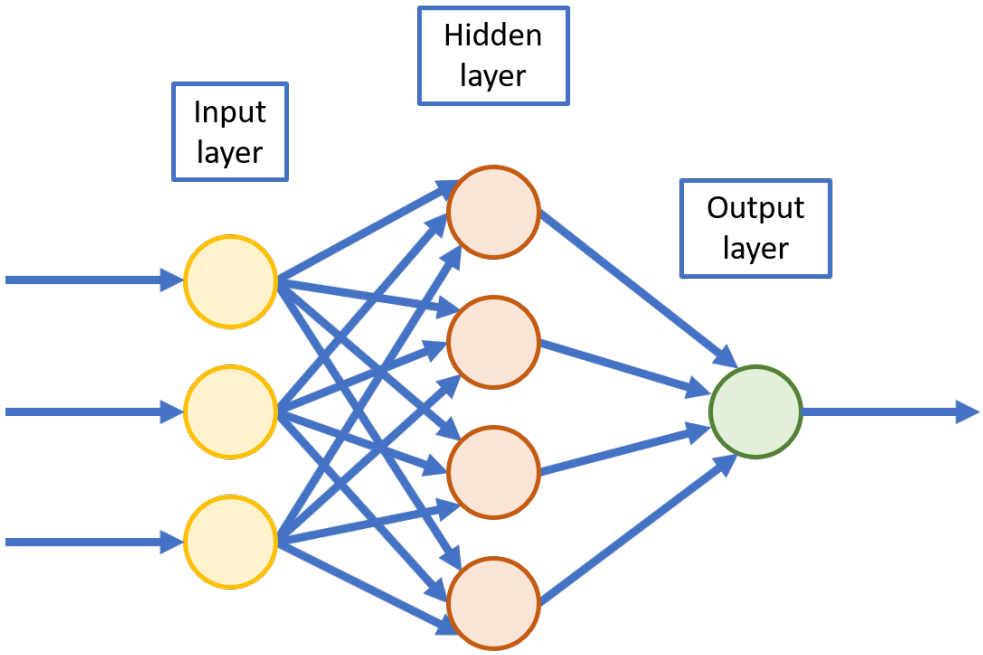


Figure 3.1: The architecture of a neural network with one hidden layer

3.1.2 Feedforward neural network architecture

A neural network consists of an input layer, one or more hidden layers and an output layer. This basic structure is shown in Figure 3.1. Each layer consists of one or more nodes (also called neurons). Each node in the hidden layer(s) does three things: it takes a weighted sum of all its inputs, adds a bias value and applies an activation function on the result. The activation function is important, because it introduces non-linearity. Without it, the network is just a series of matrix multiplications, which means the result is always a linear function of the inputs. The activation function enables the network to approximate any arbitrary function. This assertion is the universal approximation theorem, which states that this holds with just one hidden layer with a finite number of neurons [70], as long as the activation function is not polynomial.

A connection from one node to another has a certain weight. These weights can be represented by a matrix $\mathbf{A}^\ell$ of size $N \times M$, with N the number of nodes in layer ℓ and M the number of nodes in the previous layer $(\ell - 1)$. The output of layer ℓ is then

$$\mathbf{u}^l = \phi^\ell(\mathbf{A}^\ell \mathbf{u}^{\ell-1} + \mathbf{b}^\ell), \tag{3.4}$$

here $\mathbf{u}^{\ell-1}$ is the output of layer $\ell - 1$, $\mathbf{b}^\ell$ is a vector with biases for neurons in layer ℓ and $\phi^\ell(x)$ is the activation function of layer ℓ. ℓ runs from 0 to L, with L the number of layers.

This means the output of the network is

$$\text{output} = \mathbf{b}^L + \mathbf{A}^L(\phi^{L-1}(\mathbf{b}^{L-1}) + \mathbf{A}^{L-1}(...\phi^1(\mathbf{b}^1 + \mathbf{A}^1\phi^0(\mathbf{b}^0 + \mathbf{A}^0\mathbf{x})))), \tag{3.5}$$

with $\mathbf{x}$ a vector of inputs for the network. There is no activation function on the output layer in this case, so the neural network can output any energy. The parameters that will be trained are the weights $\mathbf{A}^\ell$ and the biases $\mathbf{b}^\ell$.

The activation function needs to be continuously differentiable (smooth), so the potential predicted by the neural net is also smooth. It is important that the potential fit is smooth because the gradient of the

potential is needed to calculate the force. ReLU (rectified linear unit, the most common choice of activation function for neural networks), which is given by

$$f(x) = \max(0, x), \tag{3.6}$$

is therefore a bad choice in this case, because it has a kink at $x = 0$. An example of a smooth activation function is softplus, which is given by

$$f(x) = \ln(1 + e^x), \tag{3.7}$$

and can be considered a smooth version of a ReLU, with the kink smoothed out. Several possible activation functions are shown in Figure 3.2. The rounded V-function is

$$f(x) = \begin{cases} -x - \frac{1}{2}, & \text{for } x < -1, \\ \frac{1}{2}x^2, & \text{for } -1 \leq x \leq 1, \\ x - \frac{1}{2}, & \text{for } x > 1, \end{cases} \tag{3.8}$$

and has as an advantage the fact that its gradient is -1 or 1 in most places, and only zero at $x = 0$. This means it is less susceptible to the vanishing gradient problem, which is something that happens when the activation function causes a gradient that is almost always very small or zero, which can happen, for example, if the inputs to the ReLU are always negative, or the inputs to the softsign function have a very large magnitude. If the gradient is very small, the network cannot update its weights and will therefore only learn very slowly.

The rounded V has a continuous but non-smooth derivative, which would lead to a kink in the force, which is not necessarily a problem (a kink in the potential would be) but may also not be desirable.

The logistic or sigmoid function has the form

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{3.9}$$

The softsign function is similar to the logistic function, but is cheaper to calculate. It is given by

$$f(x) = \frac{x}{1 + |x|}, \tag{3.10}$$

which means it also has a continuous but non-smooth derivative. The half square function

$$f(x) = \begin{cases} 0, & \text{for } x \leq 0, \\ x^2, & \text{for } x > 0, \end{cases} \tag{3.11}$$

which means it has a continuous but non-smooth derivative as well. It also does not always deal well with outliers because of the quadratic part.

3.1.3 Training the neural network

A neural network learns from data. The data used for training is called the training data, and consists of the inputs to the network and the desired output. In our case these are a configuration of atoms and the corresponding ab initio energy. The output of the network is then compared to the desired output, and from this a loss is computed, which is a value that reflects how much they differ. A common choice for a loss function is the mean squared error. Then, the gradient of the loss with respect to all parameters in the network is computed using backpropagation, which is an algorithm that uses the chain rule to calculate the gradient backward from the last layer to the first one. This gradient is then used to update the parameters, by changing the parameters in the direction of the negative gradient.

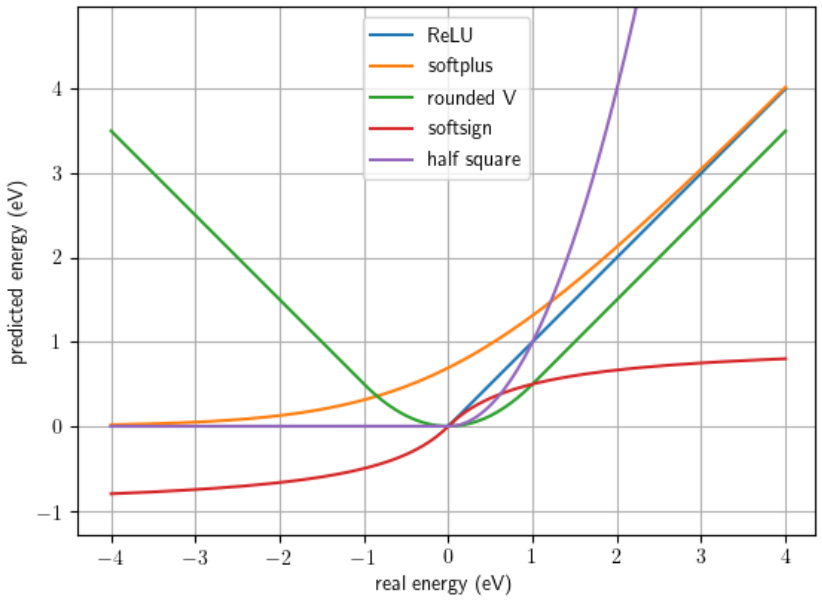


Figure 3.2: Several activation functions

The size of the steps depend on the learning rate. If the learning rate is too high, the model can show oscillations in its predictions and overshoot, or get stuck in a bad state. If the learning rate is too low, training will take very long. It is a common strategy to decrease the learning rate in steps while training.

There are various iterative methods to update the parameters. The simplest is gradient descent, which calculates the gradient for the entire training dataset at once. However, usually it is more efficient and more effective to use only a subset of the training data (a 'batch') at a time to estimate the gradient. This is called stochastic gradient descent and the extra randomness this introduces makes the model converge faster and helps it to escape local minima. This gradient computation and parameter update is repeated a lot of times. When each data point in the training data set has been used one time, one epoch has passed. A variation on stochastic gradient descent is Adam, which is an adaptive method that can be used to train the network instead of stochastic gradient descent [71]. It gives a kind of momentum to the parameter updates, which often makes it converge faster.

Common problems during training a neural network are an exploding gradient or a vanishing gradient. A vanishing gradient is often caused by the activation function, because a lot of activation functions have ranges where their gradient is very small. For example, the logistic function has a very small gradient if the magnitude of its input is large. Often this is exacerbated by having many layers in the network, since the gradient then on average gets exponentially smaller with each layer, causing the first layers to train very slowly. The exploding gradient problem is the opposite problem and happens when a gradient gets exponentially larger.

Another common problem is overfitting. Overfitting means the network 'memorizes' the training data and will then perform very well on this training data, but really bad on unseen data. This is why it is standard practice in machine learning to separate the data set into training, validation and testing data. The training data is the data that is used for computing the parameter updates, and validation data is data that is used to track how the training is progressing. If the loss for the validation increases while the training loss keeps decreasing, it is a sign that the model is starting to overfit and we should stop training. This is called 'early stopping' and is the simplest way to prevent overfitting.

Validation data is also used for hyperparameter optimization, which refers to choosing the right values for the parameters that control how training progresses or describe the structure of the model and cannot be trained by gradient descent. They are, among others, the learning rate, the batch size, the number of neurons, the number of layers in the network, the type of activation functions used, and in our case also how many and which type of proximity matrices to use. It is also possible to combine the training and validation data set and use them for cross-validation, in which case the network is trained a number of times, and each time a different chunk of data is left out and used as validation data. This gives a better idea of how consistent the results are.

In the end, we pick the model that performs best on the validation data. However, this means that in order to know how the model will actually perform on completely unseen data, we need *another* data set with data that has not been used for either training or hyperparameter optimization, and this is why there is a test set in addition to a validation set.

3.1.4 Descriptors

The input of the network is a certain 3D configuration of a system of N atoms. Such a system has $3N - 6$ degrees of freedom (3 per atom, minus 3 for rotational and translational symmetry each) if $N \geq 3$ (for $N = 1$ and $N = 2$ the number of degrees of freedom are 0 and 1 respectively). More generally, in D dimensions the number of degrees of freedom is $DN - D - \frac{1}{2}D(D-1)$ if $N \geq D$. The term DN is the number of coordinates needed to pinpoint all particles, the term $-D$ accounts for translation invariance, and $\frac{1}{2}D(D-1)$ accounts for rotational invariance. This configuration needs to be described somehow, using a descriptor. A descriptor is a numeric representation of the configuration of the system and is used as input for the neural network. One of the simplest descriptors is just the set of Cartesian coordinates of all atoms. Desirable qualities of a descriptor are

- it is smooth (a small change in the configuration always results in a small change in the description),
- it is invariant under rotation, translation and permutation of identical atoms,
- each possible configuration has a unique description,
- it is easily scalable, such that it can easily be adapted for different and/or larger systems,
- the inputs are as orthogonal as possible, such that one input does not depend (strongly) on any of the other inputs, so all inputs give as much information as possible, independent of the other inputs,
- the number of inputs is small, while still not under-specifying the system (so at least a number of inputs equal to the number of degrees of freedom, but not much more than that).

The requirement of invariance under rotation and translation means that when the entire system is rotated or moved, but the distances between the atoms do not change, the potential should be the same. Invariance under permutation means that it should not matter in which order the atoms are described and that if two atoms of the same species are swapped, the potential stays the same. This means the following must hold for the descriptor $f(\mathbf{X})$:

$$f(\mathbf{X}) = f(\mathbf{QX}), \tag{3.12}$$

where $\mathbf{X}$ is a matrix describing the configuration with one row per atom and $\mathbf{Q}$ any permutation matrix.

The set of Cartesian coordinates is not invariant under rotation, translation and permutation. One approach is to use functions that respect the specific symmetry of the system to transform the Cartesian coordinates into a suitable representation. For example, Lorenz, Groß, and Scheffler [72] use inputs that are functions of the coordinates that describe the orientation of a H_2-molecule to a surface to predict the

sticking probability. A disadvantage is that this is not a very scalable approach, because the functions are very specific to the system.

Another approach is to use the neural network not to predict the total energy, but the energy contribution E_i of each atom [73]. The descriptor is then a description of the local environment of this center atom, meaning the atoms that are around it. If the number of atoms is large, this local environment can have some kind of cutoff to make the prediction more efficient. This description of the local environment should respect permutation (of the neighbor atoms), rotation and translation symmetry. The energy contributions of the N atoms are then added up to get the prediction of the total energy

$$E = \sum_{i=1}^{N} E_i. \tag{3.13}$$

This makes it invariant with respect to permutation of identical atoms, since the order in a sum does not matter.

Rotation and translational invariance of the local environment description can be achieved by using atom-centered symmetry functions (ACSF). These depend on the distances of other atoms to the center atom, and the angles pairs of atoms form with this atom. This approach can be extended to take multiple species of atoms into account [74]. Another way to describe the local environment is to use spherical harmonics [75].

An advantage of an approach that attempts to predict the energy contribution of each atom, is that one network can be used to predict the energy systems with varying numbers of atoms and types of atoms (although the training data in that case also needs to include varying numbers of atoms and types of atoms, because neural networks are generally not good at extrapolating).

3.1.5 Permutation invariance in general

Atom configurations are not the only type of data where permutation invariance is important, it is an important consideration for any type of data that is a set instead of a vector. Since neural networks work on vectors, not sets, the input must be transformed in a permutationally invariant representation [76]. The size of this representation is the latent space and its dimensionality must be at least that of the configuration [77].

This can be realized by transforming the inputs into a permutationally invariant representation using symmetric functions, which are functions that have the same value regardless of the order of its arguments. A general way of creating permutationally invariant functions is by applying any function $h(x)$ to each input p_i and then summming them,

$$f(\mathbf{x}) = \sum_i h(p_i). \tag{3.14}$$

The result is always permutationally invariant with respect to p_i. The same goes for any commutative operation, such as the product

$$f(\mathbf{x}) = \prod_i h(p_i). \tag{3.15}$$

Some examples are

$$f(\mathbf{p}) = p_1 + p_2 + p_3 + p_4 + p_5 + p_6 \tag{3.16}$$

$$f(\mathbf{p}) = p_1^2 + p_2^2 + p_3^2 + p_4^2 + p_5^2 + p_6^2, \text{ etc.} \tag{3.17}$$

and

$$f(\mathbf{p}) = p_1 p_2 p_3 p_4 p_5 p_6, \tag{3.18}$$

$$f(\mathbf{p}) = (p_1 + 1)(p_2 + 1)(p_3 + 1)(p_4 + 1)(p_5 + 1)(p_6 + 1), \text{ etc.} \tag{3.19}$$

Taking the maximum

$$f(\mathbf{p}) = \max h(p_i), \tag{3.20}$$

is a type of max pooling and is also permutationally invariant. Max pooling is unsuitable for a potential because it causes kinks, but it is used for other applications, such as classification [78, 79]. Chen, Cheng and Mallat take a [80] slightly different approach; instead of summing over all inputs, they sum over only one pair of inputs at a time, creating only permutation invariance for exchange of one specific pair of atoms, and a cascade of such sums is used to implement global permutation invariance.

Ravanbakhsh, Schneider and Póczos use something similar to (3.14): the outcome of permutation *equivariant* layers is summed or max pooled over a set and the output of this is fed into another neural network [78]. An equivariant function has the following requirement

$$\mathbf{Q}f(\mathbf{p}) = f(\mathbf{Q}\mathbf{p}), \tag{3.21}$$

where $\mathbf{Q}$ is any permutation matrix. This means that when the input is permuted, the output is permuted the same way. A message passing neural network is a generalization of this approach. It involves creating an embedding for each node. During this phase, the nodes can exchange information ('messages'). The embeddings are combined using a commutative operation, usually a sum such as (3.14), and the outcome of this is passed to another neural network [81, 82]. Maron et al. use a similar approach [83].

Another way of looking at a configuration of atoms is to consider it a point cloud [79, 84]. A point cloud is a set of points distributed in a (usually 3D) space. Each point has an x, y and z coordinate and other possible attributes associated with it. Usually the term point cloud is used for such data that is the result from 3D scanners. Neural networks that take a point cloud as an input also need permutation invariance with respect to the order of the points.

3.1.6 Proximities

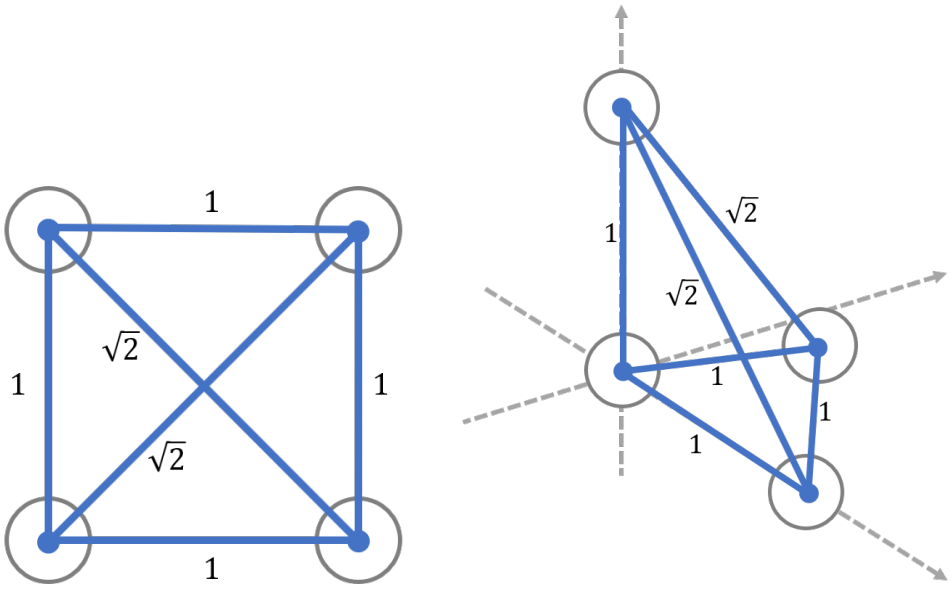
For N atoms, the set of $N(N-1)/2$ interatomic distances $r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ is invariant under rotation and translation, but not under permutation. This can be generalized to interatomic distances that are transformed in some way (for example $p_{ij} = \exp(-r_{ij})$ or $p_{ij} = 1/r_{ij}^n$), such that they go to zero when the atoms are infinitely far apart and therefore non-interacting. The resulting quantities are called proximities[1] and have the same invariance under rotation and translation.

Any representation that is permutationally invariant with respect to the *proximities* is also permutationally invariant with respect to the *atoms*. Achieving permutation invariance by simply sorting the proximities is not a suitable descriptor. If they are ordered from high to low, and the particles then start to move, there can be a point where one distance becomes higher than another, which means they abruptly switch order, which can lead to 'kinks' in the potential. If we use the aforementioned symmetric functions on the proximities, we have a permutationally invariant representation that can be used as a descriptor.

However, a drawback of any method that achieves permutation invariance of the atoms with permutation invariance of the proximities, is that information is lost, because order does matter somewhat; the same set of distances or proximities can describe different configurations. For example, the interatomic distances $\{1, 1, 1, 1, \sqrt{2}, \sqrt{2}\}$ can describe either a square (Figure 3.3a) or a wonky tetrahedron (Figure 3.3b), depending on which distances apply to which atom. However, this might only be a problem in the case of 4 or less atoms, since for any case with more atoms, the number of proximities is more than the number of degrees of freedom, which means there is some redundancy that makes it very difficult to find another possible configuration with the same set of proximities.

Braams, Bowman and Xie [25, 85] give ways of creating permutationally invariant polynomials (PIPs) for different systems with up to 5 atoms, that *do* take into account which proximities belong to the same

[1] Strictly speaking, the resulting quantities are only proximities if the function $p_{ij} = f(r_{ij})$ is monotonic and non-negative, but we are ignoring this.



(a) Square configuration *(b) Wonky tetrahedron*

Figure 3.3: Two different configurations of 4 atoms with the same set of interatomic distances

atom. The PIP method is implemented in a library called ezPES. These PIPs were intended to be (and have been [86, 87, 88, 89]) used for linear regression, but the polynomials can also be used as neural network inputs [90, 91]. However, the number of monomials per polynomial scales badly since the number of possible permutations is

$$\prod_{i=0}^{M} N_i!, \tag{3.22}$$

with M the number of atom species and N_i the number of atoms of type i. For example, for 4 H-atoms there are $4! = 24$ possible permutations, and for 2 CO_2 molecules there are $2!4! = 48$ possible permutations. However, for bigger systems the number of permutations grows faster than exponentially, for example for 15 identical atoms, the number of permutations is about 1.3×10^{12}.

The following equations show the PIPs up to third order that can be used as a neural network input for a system of four atoms of the same type. Each sum sums over all possible permutations of the applicable

indices i, j, k and l, such that none of the indices are equal and each has a value of 1 to 4:

$$f_1(\mathbf{p}) = \sum_{ij} p_{ij} = p_{12} + p_{13} + p_{14} + p_{23} + p_{24} + p_{34} \tag{3.23}$$

$$f_{2a}(\mathbf{p}) = \sum_{ij} p_{ij}^2 \tag{3.24}$$

$$f_{2b}(\mathbf{p}) = \sum_{ijk} p_{ij} p_{ik} \tag{3.25}$$

$$f_{2c}(\mathbf{p}) = \sum_{ijkl} p_{ij} p_{kl} \tag{3.26}$$

$$f_{3a}(\mathbf{p}) = \sum_{ij} p_{ij}^3 \tag{3.27}$$

$$f_{3b}(\mathbf{p}) = \sum_{ijk} p_{ij}^2 p_{ik} \tag{3.28}$$

$$f_{3c}(\mathbf{p}) = \sum_{ijkl} p_{ij}^2 p_{kl} \tag{3.29}$$

$$f_{3d}(\mathbf{p}) = \sum_{ijk} p_{ij} p_{jk} p_{ik} \tag{3.30}$$

$$f_{3e}(\mathbf{p}) = \sum_{ijkl} p_{ij} p_{ik} p_{il} \tag{3.31}$$

$$f_{3f}(\mathbf{p}) = \sum_{ijkl} p_{ij} p_{ik} p_{kl}. \tag{3.32}$$

Because a system with four atoms has six degrees of freedom, not all of these polynomials need to be used.

3.1.7 Proximity matrix

Proximities are a good starting point for a descriptor, because they are already invariant with respect to translation and rotation of the entire system, but which proximities correspond to the same atom is also important information. To take this into account, one can use the proximities arranged in a matrix $\mathbf{P}$, called a proximity matrix. Proximity matrices have been used before to create embeddings of networks to use as input for a neural network [92]. Because $p_{ij} = p_{ji}$, it is a symmetric matrix. The diagonal contains only zeros, making it a hollow matrix. For a system with 4 atoms, the proximity matrix looks as follows:

$$\begin{pmatrix} 0 & p_{12} & p_{13} & p_{14} \\ p_{12} & 0 & p_{23} & p_{24} \\ p_{13} & p_{23} & 0 & p_{34} \\ p_{14} & p_{24} & p_{34} & 0 \end{pmatrix}. \tag{3.33}$$

For example, the following proximity matrix (where the proximities are just the interatomic distances) describes a square (see again Figure 3.3a)

$$\begin{pmatrix} 0 & 1 & 1 & \sqrt{2} \\ 1 & 0 & \sqrt{2} & 1 \\ 1 & \sqrt{2} & 0 & 1 \\ \sqrt{2} & 1 & 1 & 0 \end{pmatrix}, \tag{3.34}$$

and the next proximity matrix describes a wonky tetrahedron (meaning, a pyramid with an equilateral triangle as a base, but the fourth atom placed directly above one of the atoms instead of above the centre of the triangle, see also Figure 3.3b)

$$\begin{pmatrix} 0 & 1 & 1 & \sqrt{2} \\ 1 & 0 & 1 & \sqrt{2} \\ 1 & 1 & 0 & 1 \\ \sqrt{2} & \sqrt{2} & 1 & 0 \end{pmatrix}. \tag{3.35}$$

The goal is then to find a representation $f(\mathbf{P})$ of this matrix that is invariant with respect to simultaneous permutation of the rows and columns. This means the following must hold

$$f(\mathbf{P}) = f(\mathbf{Q}^T\mathbf{P}\mathbf{Q}), \tag{3.36}$$

with $\mathbf{Q}$ any permutation matrix.

A different kind of matrix similar to a proximity matrix is the Coulomb matrix

$$M_{ij} = \begin{cases} 0.5 \cdot Z_i^{2.4}, & \text{for } i = j, \\ \frac{Z_i \cdot Z_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}, & \text{for } i \neq j, \end{cases} \tag{3.37}$$

with Z_i the nuclear charge (atom number) of atom i. The diagonal entries are a polynomial fit of the atomic energies as a function of the nuclear charge, and the off-diagonal elements represent the Coulomb repulsion between nuclei. The Coulomb matrix and any proximity matrix are invariant with respect to rotation and translation, but not to permutation. To achieve permutation invariance as well, one can use the ordered eigenvalues of the Coulomb matrix [21]. Ordering the eigenvalues is less likely to lead to kinks in the potential than ordering the proximities, because eigenvalues almost never cross. Crossings are very unlikely and can usually only cross this way when in very specific configurations with certain kinds of symmetry that the system is unlikely to end up in. In all other situations, avoided crossings happen (see Figure 3.4 for an example), although the separation at these crossings can get very small. Unfortunately, the number of eigenvalues is always the same as the number of atoms N, which means for $N > 3$, the system is underspecified (since the number of dimensions of a system with three or more atoms is $3N - 6$).

3.1.8 Proximity matrix eigenspectrum method

The descriptor used in this report is based on unpublished research by prof. dr. ir. Vianney Koelman. The method is called the proximity matrix eigenspectrum (PME) method and is similar to the Coulomb matrix method in that it uses the eigenvalues, but proximity matrices are used instead of a Coulomb matrix, and the problem of underspecification is solved by using multiple proximity matrices.

The eigenvalues of a matrix $\mathbf{A}$ are the values λ_i that together with the eigenvectors $\mathbf{v}_i$ are the solutions to the eigenvalue equation

$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i. \tag{3.38}$$

If $\mathbf{A}$ is an $N \times N$ matrix, it will have N eigenvectors. If n eigenvalues have the same value λ, the algebraic multiplicity of λ is n. The geometric multiplicity is the number of linearly independent eigenvectors associated with λ. Since a symmetric matrix has N orthogonal eigenvectors, the geometric multiplicity is the same as the algebraic multiplicity.

The inputs used for the neural network are the eigenvalues of a number of proximity matrices. The number of proximity matrices should be at least enough to avoid underdeterminancy. This means the number of independent inputs should be at least as large as the number of degrees of freedom. The number of degrees of freedom/dimension of any system is $N_{\mathrm{DoF}} = 3N - 6$ if $N > 2$, with N the number of particles (for $N = 1$: $N_{\mathrm{DoF}} = 0$, for $N = 2$: $N_{\mathrm{DoF}} = 1$). The size of the proximity matrix is $N \times N$ and therefore it has N

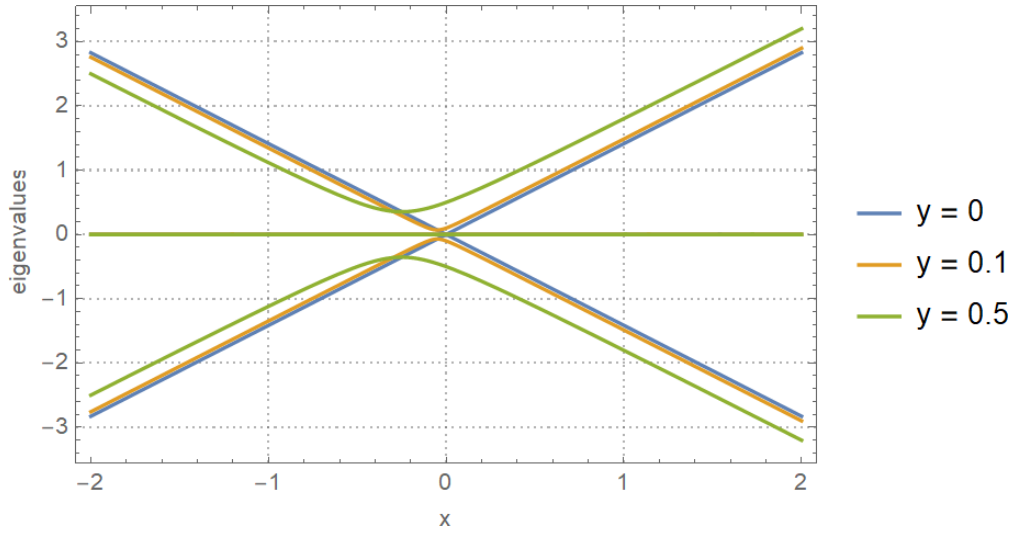


Figure 3.4: The eigenvalues of the symmetric matrix

$$\begin{pmatrix} 0 & x+y & 0 \\ x+y & 0 & x \\ 0 & x & 0 \end{pmatrix},$$

for three values of y. It shows that only when $y = 0$ (which makes the matrix persymmetric: symmetric with respect to the northeast-to-southwest diagonal), the eigenvalues can cross. When $y \neq 0$, an avoided crossing happens.

eigenvalues. The diagonal only has zeros, which means that the trace of the matrix (sum of its diagonal elements) is always zero. The trace is always equal to the sum of the eigenvalues which must therefore also be zero. This means that for each proximity matrix, only $N-1$ of the N eigenvalues give useful information. Therefore, the minimum number of proximity matrices needed to avoid underdeterminancy is always 3 or less.

Eigenvalues are ordered from low to high per matrix. The advantage of this approach is the avoided crossings, which avoids the problem with kinks in the potential with using the interatomic distances themselves, since eigenvalues can only cross this way for very specific configurations with certain kinds of symmetry that the system is unlikely to end up in. The eigenvalues are also invariant under permutation of the atoms (switching two atoms).

We used several different proximities, all of which go to zero as the distance between the atoms goes to infinity. Examples of such proximities are $p_{ij} = 1/r_{ij}^n$ (type A) or $p_{ij} = (r_0/r_{ij} - r_{ij}/r_0)^n$ (type B).

Figure 3.5 shows the eigenvalues for the six different configurations of two H_2 molecules when the distance between them is varied. Figure 2.6 shows the configurations and 3.6 shows special cases of these configurations. It shows that the eigenvalues can only cross for the H and the X configuration when the atoms form a perfect square or a tetrahedron, respectively. It is very unlikely that the system ends up in one of these configurations by coincidence.

When there are atoms of different kinds, the different permutation symmetries need to be taken into account; switching two of the same atoms should lead to the same description, but switching two atoms of a different kind results in distinct configurations and should not have the same description. To take this into account, a weight factor $\alpha_{\ell m}$ is added to the elements of the proximity matrix. It depends on the species

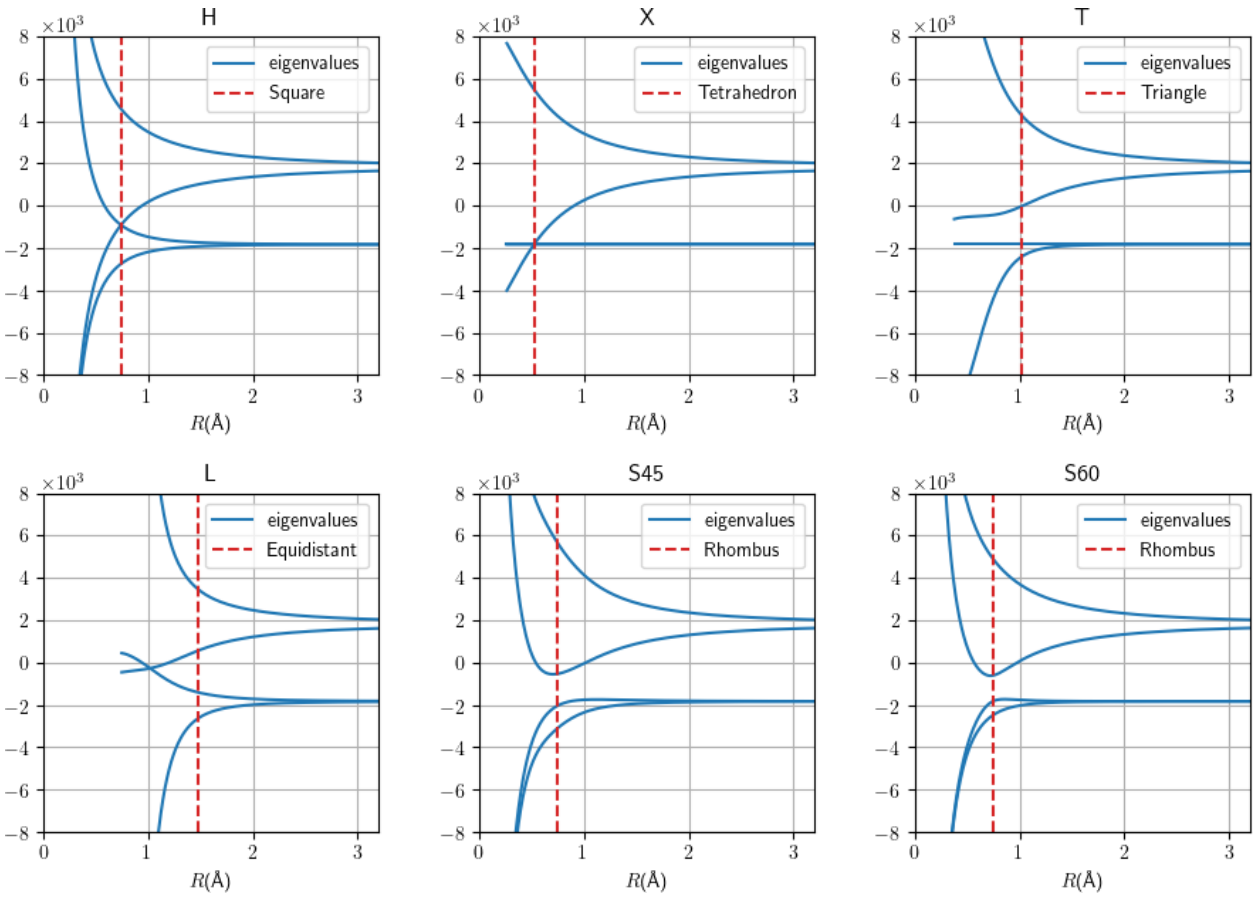


Figure 3.5: The eigenvalues of a proximity matrix (type B) with $n = 2$ and $r_0 = 32\,\text{Å}$ for six different configurations of two H_2 molecules, varying the distance R between the centers of mass of the molecules. The distance between atoms within one molecule is kept constant at the equilibrium distance. The red dashed line marks distances for which the configurations form a special configuration shown in Figure 3.6.

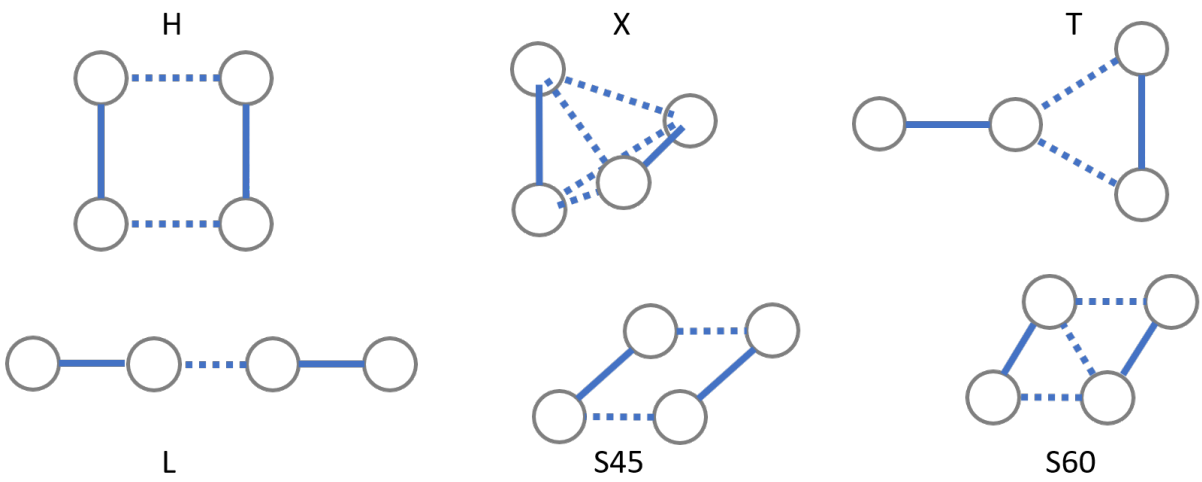


Figure 3.6: Special cases of the six different configurations of two H_2 molecules. They are: H) a square, X) a tetrahedron, T) an equilateral triangle, L) an equidistant spacing between atoms, S45) a rhombus, S60) a rhombus composed of two equilateral triangles.

ℓ, m of the atoms and is different per proximity matrix. For two CO_2 molecules, this looks like

$$\begin{pmatrix} 0 & \alpha_{CC}/r_{12}^n & \alpha_{CO}/r_{13}^n & \alpha_{CO}/r_{14}^n & \alpha_{CO}/r_{15}^n & \alpha_{CO}/r_{16}^n \\ \alpha_{CC}/r_{12}^n & 0 & \alpha_{CO}/r_{23}^n & \alpha_{CO}/r_{24}^n & \alpha_{CO}/r_{25}^n & \alpha_{CO}/r_{26}^n \\ \alpha_{CO}/r_{13}^n & \alpha_{CO}/r_{23}^n & 0 & \alpha_{OO}1/r_{34}^n & \alpha_{OO}/r_{35}^n & \alpha_{OO}/r_{36}^n \\ \alpha_{CO}/r_{14}^n & \alpha_{CO}/r_{24}^n & \alpha_{OO}/r_{34}^n & 0 & \alpha_{OO}/r_{45}^n & \alpha_{OO}/r_{46}^n \\ \alpha_{CO}/r_{15}^n & \alpha_{CO}/r_{25}^n & \alpha_{OO}/r_{35}^n & \alpha_{OO}/r_{45}^n & 0 & \alpha_{OO}/r_{56}^n \\ \alpha_{CO}/r_{16}^n & \alpha_{CO}/r_{26}^n & \alpha_{OO}/r_{36}^n & \alpha_{OO}/r_{46}^n & \alpha_{OO}/r_{56}^n & 0 \end{pmatrix}, \tag{3.39}$$

with the first two rows and columns corresponding to the two carbon atoms and the other rows and columns to the oxygen atoms. The values of the weights are also trained by the neural network and must therefore be included in backpropagation. The disadvantage of this is the fact that taking the backpropagation further back, through the computation of the eigenvalues, means the training becomes slower. This part of the method is still untested, so it is possible there are other, better ways to implement multiple species.

3.1.9 Properties of the eigenspectrum

Since the proximity matrix $\mathbf{P}$ is symmetric and all its values are real, the eigenvalues are always real (this holds for any Hermitian matrix). Any symmetric $N \times N$ matrix has a system of N orthogonal eigenvectors. These eigenvectors can also be used to diagonalize $\mathbf{P}$ [93, §7.2].

In this report, all the proximities that are used go to zero when the distance between two atoms approaches infinity or is larger than a certain cutoff. This means when all atoms are infinitely far away, all $p_{ij} = 0$ and therefore all eigenvalues are zero. This has an extra advantage in the fact that adding [**?**]bias parameters is no longer needed when the potential value in the case of all atoms infinitely far apart is defined to be zero. In that case the potential also goes to zero.

When the system consists of M non-interacting parts, all elements p_{ij} corresponding to two atoms that are in different parts become zero, which means the proximity matrix becomes a diagonal block matrix

with M blocks $A_1, \ldots, A_M$, each corresponding to one part. The eigenvalues of the proximity matrix are just the eigenvalues of each block, which are independent of each other. This means that there can be eigenvalue crossings that are *not* avoided. It also means the sum of the eigenvalues of each block is zero. For a 2-atom part this results in two eigenvalues with the same absolute value but opposite sign. For a 1-atom part (one atom that does not interact with the rest), this results in an eigenvalue of zero.

In the case of proximity matrix type A, the proximity matrix elements do not actually go to zero if the atoms are finite distances apart, so there are no eigenvalue crossings. Instead really small values of p_{ij} outside of the blocks compared to within them lead to crossings that are avoided, but the separation between the eigenvalues at this avoided crossing is very small.

As r_{ij} goes to 0 (meaning two atoms get very close together) p_{ij} goes to infinity (for both proximity matrices type A and B if a positive n is used), which means at least one eigenvalue goes to infinity as well, which can be good because the potential goes to infinity then as well.

3.1.10 Proximity matrix invariants method

Another possible way to use a proximity matrix $\mathbf{P}$ is to use other matrix invariants[2]. The quantities

$$\mathrm{tr}\big(\mathbf{P}^k\big), \tag{3.40}$$

for various $k > 1$ (the invariant with $k = 1$ is $\mathrm{tr}(\mathbf{P}) = 0$, which is why we only use $k > 1$), are invariants of $\mathbf{P}$, meaning they are independent of the basis of the matrix, hence the name proximity matrix invariants (PMI) method. Figure 3.7 shows an example of these invariants. They can be calculated using the numpy functions `numpy.linalg.matrix_power()` or `np.einsum()` and `numpy.trace()`. Just as the Braams-Bowman-Xie method, the invariants can be used as inputs for a linear regression or a neural network.

These PMIs are equivalent to sums of all possible closed walks over the particles. In graph theory, a walk is a sequence of edges that connects a sequence of nodes. 'Closed' means the first and last node in this sequence are the same. To show this, we can write the PMIs as a nested sum. First, the result of squaring $\mathbf{P}$ is

$$(\mathbf{P}^2)_{ij} = \sum_{\ell=1}^{N} p_{i\ell} p_{\ell j}. \tag{3.41}$$

Extending this to higher powers, the result of raising $\mathbf{P}$ to the power k is then the following expression of nested sums:

$$(\mathbf{P}^k)_{ij} = \sum_{\ell_1=1}^{N} p_{i\ell_1} \sum_{\ell_2=1}^{N} p_{\ell_1\ell_2} \cdots \sum_{\ell_{k-1}=1}^{N} p_{\ell_{k-2}\ell_{k-1}} p_{\ell_{k-1}j}. \tag{3.42}$$

This means the trace is

$$\mathrm{tr}\big(\mathbf{P}^k\big) = \sum_{i=1}^{N} (\mathbf{P}^k)_{ii} = \sum_{\ell_1=1}^{N} p_{i\ell_1} \sum_{\ell_2=1}^{N} p_{\ell_1\ell_2} \cdots \sum_{\ell_{k-1}=1}^{N} p_{\ell_{k-2}\ell_{k-1}} p_{\ell_{k-1}i}. \tag{3.43}$$

This expression consists of terms such as, for example, $p_{12}p_{23}p_{31}$ for $k = 3$, where the last index of each p_{ij} is equal to the first index of the next one, which corresponds to a walk over the atoms as nodes. The walk is closed, because the first index of the first p_{ij} is equal to the last index of the last p_{ij}.

This means this method of constructing invariants is similar to the PIP method when using only the polynomials that correspond to a closed walk over the atoms as nodes (for example, of the polynomials in Figure 3.8, only 3.8d qualifies), although multiple PIPs are included per PMI. However, the PMIs are a lot faster to calculate, because not all the monomials need to be calculated separately.

[2]Credit for this method also goes to prof. dr. ir. Vianney Koelman

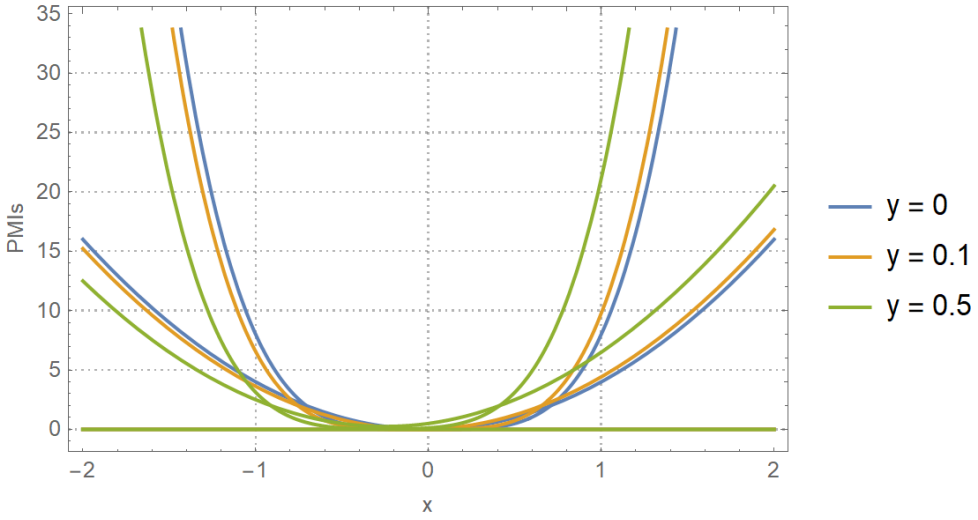


Figure 3.7: The first 3 PMIs $\mathrm{tr}(\mathbf{P}^k)$*, with* $k = 2, 3, 4$ *of the symmetric matrix*

$$\begin{pmatrix} 0 & x+y & 0 \\ x+y & 0 & x \\ 0 & x & 0 \end{pmatrix},$$

for three values of y*.*

It is possible to also include all open walks (walks where the first and last node are not the same), by not only taking the traces, but also using what we call the 'trace complement' of each matrix, which is the sum of all its entries except for the diagonal. The trace complement $\mathrm{tc}(\mathbf{P}^k)$ is an invariant with respect to simultaneous permutation of the rows and columns and is therefore also a PMI. This means in Figure 3.8, graphs 3.8a, 3.8b and 3.8f are also included.

For a linear regression, one can use $\mathrm{tr}(\mathbf{P}^k)$ with any $k > 1$ and $\mathrm{tc}(\mathbf{P}^k)$ with any $k > 0$. However, for a neural network input this is not necessarily very useful, since higher order invariants can be written as linear combinations of products of the lower ones. More specifically, the invariants $\mathrm{tr}(\mathbf{P}^{N+1})$ and $\mathrm{tc}(\mathbf{P}^N)$ and all higher order invariants are linear combinations of products of the lower ones[3]. Feeding these coefficients into the neural network does not give the network any useful extra input, so there are $2(N-1)$ useful PMIs per proximity matrix.

An advantage of this approach over the PME method is that the PMIs do not need to be ordered anymore, which means there will never be any kinks, since sorting is what can cause kinks. In other words, the coefficients are already a vector, not a set. When all proximities are zero, the PMIs are also zero, so this approach also has the advantage that adding bias parameters is unnecessary and the potential will automatically go to zero at infinity. These characteristic polynomials are also promising for characterizing graphs and creating graph representations to use as input of a neural network [95, 96].

[3] A strict proof of this is outside the scope of this report, but the following explanation hopefully shows why this makes sense. Any trace $\mathrm{tr}(\mathbf{P}^k)$ consists of a sum of terms that correspond to a closed walk of length k over the atoms. For example, $\mathrm{tr}(\mathbf{P}^6)$ will contain the term $p_{12}p_{23}p_{32}p_{23}p_{34}p_{41}$. Each of these terms contains k pairs of identical indices, because the last index of each p_{ij} is equal to the first index of the next one and first index of the first p_{ij} is equal to the last index of the last p_{ij}. Since $k > N$, this means there must be at least one pair that occurs twice. If we then split the walk between each of these pairs, we can form two new closed walks. For example, $p_{12}p_{23}p_{32}p_{23}p_{34}p_{41}$ can be split as $p_{12}|p_{23}p_{32}|p_{23}p_{34}p_{41}$, which can be rearranged to form a product of the closed walks $p_{12}p_{23}p_{34}p_{41}$ and $p_{23}p_{32}$. If the resulting shorter walks are still longer than N, they can be split again, until all terms are products of closed walks of a length less than or equal to N. The first N trace PMIs also consist of these terms.

3.1.11 Comparison time complexity PIP vs PMI method

For both the PIP and the PMI method, one can use multiple proximity matrices that are parameterized differently. Here we compare the computation of permutationally invariant quantities for one proximity matrix, for N identical atoms. This gives us $n = N(N-1)/2$ proximities per proximity matrix.

The number of monomials of degree k is

$$\binom{n+k-1}{k}, \tag{3.44}$$

(combinations with replacement), with $n = N(N-1)/2$ the number of proximities. For constant n and $k \to \infty$, this scales as $\Theta(k^{n-1})$[4]. The total number of monomials of degree k **or less** is

$$\binom{n+k}{k}. \tag{3.45}$$

For constant n and $k \to \infty$, this scales as $\Theta(k^n)$. For constant k and $n \to \infty$, this scales as $\Theta(n^k) = \Theta(N^{2k})$. In the PIP method these monomials are then grouped together in permutationally invariant polynomials (PIPs). When all the atoms are identical, the number of PIPs of a certain degree k is the same as the number of non-isomorphic (two graphs that have the same structure but different node labels are called isomorphic) loopless multigraphs possible with N nodes and k edges (loopless: no loops from a node to itself allowed, multigraph: multiple edges between two nodes allowed), because each PIP corresponds to one such graph. Such graphs do not need to be connected. For example, there are 6 possible unique graphs with 4 nodes and 3 edges, and therefore also 6 possible PIPs of order $k = 3$ for a system of 4 atoms. Figure 3.8 shows these 6 graphs and the form of the monomials in the corresponding PIP. A way to calculate the number of possible graphs is described in [97, §8.9.2, p. 664]. Table 3.1 shows the number of monomials and the number of PIPs of a certain order for systems with identical atoms.

Scaling with k

PIP method: The total number of monomials scales as k^n and for each monomial we need to multiply up to k proximities together, which means the number of proximities per monomial scales as $\Theta(k)$, so time complexity is $\Theta(k^{n+1})$.

PMI method: To get the PMIs up to order k, we need to do $k-1$ matrix multiplications of matrices with size $N \times N$. Matrix multiplication using a naive method that implements the matrix multiplication $\mathbf{C} = \mathbf{AB}$

$$c_{ij} = \sum_{k}^{N} a_{ik} b_{kj} \tag{3.46}$$

with 3 nested for loops that loop over i, j and k has time complexity $\Theta(N^3)$[5], which is a constant, because we are keeping N constant. Therefore the time complexity is $\Theta(k)$.

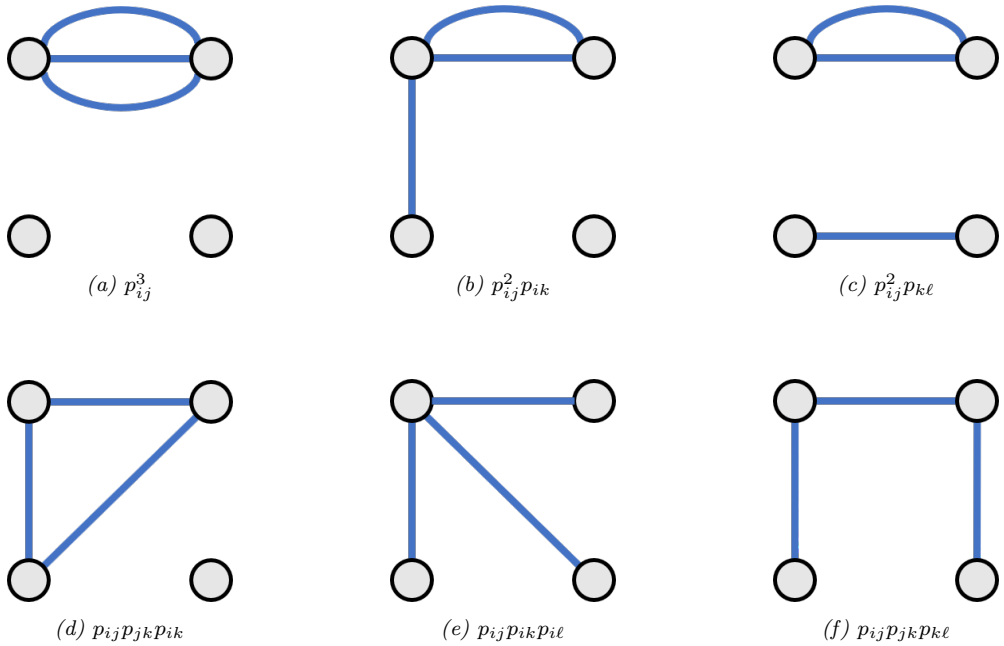
This means the PMI method scales much better with k than the PIP method. However, this is misleading, because the number of PIPs up to order $k_{\max}$ is much larger ($\Theta(k_{\max}^n)$) than the number of PMIs, which is $2k_{\max} - 1$ (one trace and one trace complement per value of k, except no trace for $k = 1$). A better comparison is the time complexity of creating the same number of permutationally invariant quantities.

[4] This is the so-called Big Theta notation, which describes the limiting behavior of a function as its argument goes to infinity; $f(N) = \Theta(g(N))$ means that as N goes to infinity, $f(N)$ grows as fast as $g(N)$. This is similar to the Big O notation, but the Big O notation only provides an upper bound; $f(N) = O(g(N))$ means that as N goes to infinity, $f(N)$ grows at most as fast as $g(N)$.

[5] There are methods that have a better asymptotic scaling, such as Strassen's algorithm, but these only start being more efficient for matrices larger than about 100×100 [99, p. 402]

Table 3.1: Number of PIPs of order k for a system of N identical atoms. The number of monomials per order is indicated in parentheses. The number of PIPs for $N \leq 8$ is taken from A192517 in OEIS [98]. For $N > 8$, parts of the rows were added where $k \leq N/2$, since these the same as for $N = \infty$ (A050535 in OEIS [98]).

N	Order k of the polynomial							
	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
2	1	1	1	1	1	1	1	1
	(3)	(6)	(10)	(15)	(21)	(28)	(36)	(45)
3	1	2	3	4	5	7	8	10
	(4)	(10)	(20)	(35)	(56)	(84)	(120)	(165)
4	1	3	6	11	18	32	48	75
	(5)	(15)	(35)	(70)	(126)	(210)	(330)	(495)
5	1	3	7	17	35	76	149	291
	(6)	(21)	(56)	(126)	(252)	(462)	(792)	(1287)
6	1	3	8	21	52	132	313	741
	(7)	(28)	(84)	(210)	(462)	(924)	(1716)	(3003)
7	1	3	8	22	60	173	471	1303
	(8)	(36)	(120)	(330)	(792)	(1716)	(3432)	(6435)
8	1	3	8	23	64	197	588	1806
	(9)	(45)	(165)	(495)	(1287)	(3003)	(6435)	(12870)
9	1	3	8	23	-	-	-	-
	(10)	(55)	(220)	(715)				
10	1	3	8	23	66	-	-	-
	(11)	(66)	(286)	(1001)	(3003)			
11	1	3	8	23	66	-	-	-
	(12)	(78)	(364)	(1365)	(4368)			
12	1	3	8	23	66	212	-	-
	(13)	(91)	(455)	(1820)	(6188)	(18564)		
13	1	3	8	23	66	212	-	-
	(14)	(105)	(560)	(2380)	(8568)	(27132)		
14	1	3	8	23	66	212	686	-
	(15)	(120)	(680)	(3060)	(11628)	(38760)	(116280)	
∞	1	3	8	23	66	212	686	2389
	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)



(a) p_{ij}^3 (b) $p_{ij}^2 p_{ik}$ (c) $p_{ij}^2 p_{k\ell}$

(d) $p_{ij}p_{jk}p_{ik}$ (e) $p_{ij}p_{ik}p_{i\ell}$ (f) $p_{ij}p_{jk}p_{k\ell}$

Figure 3.8: All possible non-isomorphic loopless multigraphs with 4 nodes and 3 edges, with the form of the corresponding polynomial terms.

Scaling with q

What happens when we scale up the number of permutationally invariant quantities q we want?

PIP method: The total number of monomials of order k scales with k^{n-1}. When N stays constant, but $k \to \infty$, the number of monomials per PIP goes to the number of permutations $N!$, which has $\Theta(1)$ (since we are treating N as a constant). Since the number of monomials per PIP goes to a constant, this means the number of PIPs needs to grow equally fast as the number of monomials, so $q(k) = \Theta(k^{n-1})$ as well.

We only found an explicit expression for the number of PIPs of order k for $N = 3$ and $N = 4$. For $N = 3$, the expression was taken from the page A001399 in OEIS [98]. The expression $N = 4$ is obtained from the generating function given on the page A001399 in OEIS [98] by applying `SeriesCoefficient` in Mathematica to this generating function (see `PIPscaling.nb`).

$$q_3(k) = \text{round}\left(\frac{(k+3)^2}{12}\right) \tag{3.47}$$

$$\begin{aligned} q_4(k) =& \frac{k^5}{2880} + \frac{k^4}{192} + \frac{13k^3}{288} + \frac{7k^2}{32} + \frac{7}{128}(-1)^k k + \frac{9439k}{17280} + \frac{21(-1)^k}{128} \\ &+ \frac{2}{27}(-1)^k k \cos\left(\frac{\pi k}{3}\right) + \frac{2}{9}(-1)^k \cos\left(\frac{\pi k}{3}\right) + \frac{1}{16}\cos\left(\frac{\pi k}{2}\right) + \frac{635}{1152}, \end{aligned} \tag{3.48}$$

which indeed gives $q_3(k) = \Theta(k^2)$ and $q_4(k) = \Theta(k^5)$, which are equal to $\Theta(k^{n-1})$.

The *total* number of PIPs then scales as $\Theta(k^n)$, so the maximum order k we need scales as $\Theta(q^{1/n})$. The number of monomials needed scales with $q \times$ nr of monomials per PIP. As mentioned previously, the number of monomials per PIP goes asymptotically to the constant value $N!$, which is independent of q and

therefore has $\Theta(1)$. For each monomial we need to multiply up to k proximities together, which means the time complexity is $\Theta(q^{1+1/n})$.

PMI method: Just as with k, matrix multiplication also scales as $\Theta(1)$ with q. The number of PMIs is $q = 2k_{max} - 1$, and the number of matrix multiplications needed is $k_{max} - 1$, so the time complexity for finding q PMIs for constant N is $\Theta(q)$.

Scaling with N

However, for the purposes of creating neural network inputs, the scaling with the number of atoms N is more important than that with the number of inputs q, because in that case only 'enough' inputs are needed, and scaling to higher q is not important. Enough inputs for a neural network usually means a number of independent inputs that is equal to the number of degrees of freedom of the system. Adding more inputs can sometimes help the network train faster, but should not be necessary because a sufficiently large neural network is a universal approximator, which means it can model any function, and can model the other possible inputs. This means scaling up to higher accuracies can be done by going to a larger neural network instead of using more PIPs or PMIs.

For this reason, we assume the number of permutationally invariant quantities q that is needed scales with the number of degrees of freedom $3N - 6$ ($N \geq 3$), which gives $q_{\text{needed}} = \Theta(N)$. When using these quantities as neural network inputs, this is probably a reasonable assumption, but it is unclear if this is also true for a linear regression.

PIP method: The number of possible permutations is $N!$ (which has $\Theta(N^N)$), which at first glance makes it seem as if the calculation of the PIPs will scale very badly with N. However, the full number of permutations is almost never necessary, so the method does not scale quite that bad. This is because when $N \geq 2k + 2$, the multigraphs representing the PIPs have at least 2 unused nodes (the k edges can use at most $2k$ nodes). This means permutations that only differ in the unused nodes are equivalent. Even if there are only 2 unused nodes, this still halves the number of relevant permutations.

Similary, when $N > 2k$ the number of PIPs $q_N(k)$ is independent of N, because adding more nodes that the edges cannot use will not change the number of multigraphs. Therefore for $N > 2k$, $q_N(k) = q_\infty(k)$.

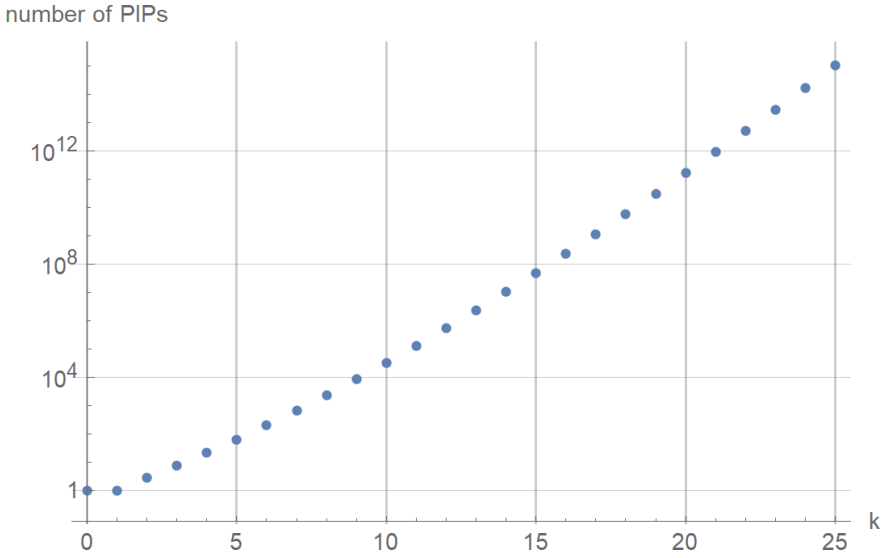


Figure 3.9: A plot of $q_\infty(k)$, which is the number of PIPs of order k, for $N = \infty$. Source: A050535 in OEIS [98].

A precise scaling for how k needs to increase with N is difficult to determine. We know how $q_N(k)$ scales with constant N and $k \to \infty$; namely as $\Theta(k^N)$, but this is obviously not valid for $N = \infty$. However, from Figure 3.9 it seems $q_\infty(k)$ increases faster than exponentially[6] (meaning $q_\infty(k) = \Omega(a^k)$), [7], with a some unknown constant), which means k needs to increase at most logarithmically with N to keep up with the $3N - 6$ degrees of freedom, so $k = O(\log N)$ seems like a reasonable assumption.

This means it will take longer and longer, while increasing N, before a certain k does not give us enough PIPs anymore and k needs to be increased. For example, if we want at least $3N - 6$ PIPs, then for $N = 13$ we need 33 PIPs, which means we need to take k up to 4 (see Table 3.1, $k \leq 4$ gives us $1 + 3 + 8 + 23 = 35$ PIPs). For $N = 14$ we need 36 PIPs, which means we need to take k up to 5, but this gives us $1 + 3 + 8 + 23 + 66 = 101$ PIPs, which means we will not have to go to a higher k until $N = 36$ when we need 102 PIPs. This means we will almost always be in the $N > 2k$ regime (specifically, if q needs to be at least $3N - 6$, we are always in that regime as $N \geq 8$).

As mentioned earlier, the number of monomials of order k scales as $\Theta(n^k) = \Theta(N^{2k})$ when $n \to \infty$ (or $N \to \infty$). For each monomial we need to multiply up to k proximities together. This means the time complexity of the PIP method is $\Theta(kN^{2k}) = \Theta\left(N^{\Theta(\log N)} \log N\right)$. This is a lot better than the complexity of $\Theta(N^N)$ one would expect based on the number of permutations. However, this better time complexity only takes the computation of the monomials into account. Unfortunately, there can also be a significant cost to figuring out what to calculate in the first place. For example, we need an efficient way to generate the PIP-graphs, and then for each PIP we need to know which permutations are relevant, so we know which monomials to calculate. Generating the PIP-graphs might be a bottleneck, because generating non-isomorphic graphs usually means generating some graphs and then throwing away the ones that are isomorphic to a previous graph [100]. The time complexity of determining if graphs are isomorphic is an unsolved problem in computer science. It is not known to be solvable in polynomial time and might be NP-complete [101, 100]. However, this is made easier by the fact that most of the time, the number and form of the PIPs is independent of N (because regime $N > 2k$). Braams-Bowman-Xie use computer algebra system MAGMA [102], another option is the program `nauty` [100].

PMI method: As mentioned earlier, we assume matrix multiplication has time complexity $\Theta(N^3)$. Since creating PMIs up to order k gives us $q = 2k - 1$ PMIs, this means the order k we need scales linearly with the number of PMIs q we need, which we assume scales linearly with the number of atoms N. This gives a total time complexity of $\Theta(N^4)$.

See Table 3.2 for a summary of how various quantities scale with k, q and N. It is worth noting that the scaling for both the PIP as well as the PMI method can be improved once a cutoff is applied, since that results in sparse proximity matrices. To scale the PIP method up to more than 10 particles, a fragmented PIP approach has been developed [103, 104] that uses this sparsity. For the PMI method it means sparse methods for matrix multiplication can be used, which have a better asymptotic scaling.

3.1.12 Relation to graphs

Previously in this chapter, the connection between the PIPs and graphs was already mentioned. However, graph theory is relevant to more than just the PIPs; a proximity matrix can also be considered the adjacency matrix (specifically, an adjacency matrix with the weights of the edges) of an undirected weighted complete graph, with the nodes corresponding to the atoms, the edges corresponding to pairs of atoms and the weight of an edge (i, j) corresponding to the proximity value p_{ij}.

This correspondence means that any method that can transform a configuration of atoms into a neural network input while taking into account permutation symmetry, can also transform a graph into such

[6] Further investigation of this graph leads us to believe the scaling is something like $\Theta(k!) = \Theta(k^k)$, because each time k increases by 1, $q_\infty(k)$ increases by a factor that increases linearly with k.

[7] This is the Big Omega-notation, which is similar to the Big O notation, but the Big Ω notation only provides a lower bound; $f(N) = \Omega(g(N))$ means that as N goes to infinity, $f(N)$ grows at least as fast as $g(N)$.

Table 3.2: Scaling of certain quantities or algorithms with: the maximum order k while keeping N constant, the number of permutationally invariant quantities q while keeping N constant, and the number of atoms N with q proportional to $3N-6$.

	Scaling		
	with k	with q	with N
Total nr. of monomials	$\Theta\left(k^n\right)$	$\Theta\left(q\right)$	$\Theta\left(N^{\Theta(\log N)}\right)$
Total nr. of PIPs q	$\Theta\left(k^n\right)$	-	$\Theta\left(N\right)$
Nr. of monomials per PIP	$\Theta\left(1\right)$	$\Theta\left(1\right)$	-
Max. k needed for PIP method	-	$\Theta\left(q^{1/n}\right)$	$\Theta\left(\log N\right)$
Nr. of proximities per monomial	$\Theta\left(k\right)$	$\Theta\left(q^{1/n}\right)$	$\Theta\left(\log N\right)$
Time complexity matrix multiplication	$\Theta\left(1\right)$	$\Theta\left(1\right)$	$\Theta\left(N^3\right)$
Time complexity PIP method	$\Theta\left(k^{n+1}\right)$	$\Theta\left(q^{1+1/n}\right)$	$O\left(N^{O(\log N)}\log N\right)$
Time complexity PMI method	$\Theta\left(k\right)$	$\Theta\left(q\right)$	$\Theta\left(N^4\right)$

an input, although not necessary the other way around, because of different kinds of atoms. This means inspiration can be taken from graph theory and any descriptor that is based on one or more proximity matrices has a much broader applicability than just physics. Graph-inspired methods have already been applied to the local environment type of descriptors [24] described by (3.13). The PME method and the PMI method can also be used to transform any arbitrary graph into a representation that can serve as input for a neural network.

The eigenspectrum of the adjacency matrix is an example of a graph invariant or graph property. In graph theory, a graph property or graph invariant is a property of graphs that depends only on the abstract structure, not on graph representations such as particular labellings or drawings of the graph [105]. Other examples of graph invariants are the characteristic polynomial and the determinant. Permutation invariance for a graph means two isomorphic graphs will have the same representation.

Using machine learning on graphs is an active area of research [106] and various other ways to achieve permutation invariance for graphs are being researched, such as graph convolution [107, 82, 108].

3.2 Methodology

3.2.1 Sampling

The data sets were created with Dalton, the use of which is explained in section 2.2. Several sampling strategies were considered.

A sampling strategy used to get an initial impression of the potential landscape is quasi-random sampling. For this, first a number of coordinates equal to the number of degrees of freedom to describe the system should be defined. For each of the coordinates describing the system, a realistic range is defined, this range is then divided in N intervals. For each combination of different intervals for the different coordinates (N^6 possibilities for a 6D system such as H_2–H_2), one data point in the volume these interval form together is randomly chosen and the potential at that point calculated. This leads to faster convergence than grid sampling, because usually some coordinates will be more important than others, but with grid sampling only one coordinate is varied, which means the data points that vary that parameter are wasted ([109] discusses this with regards to hyperparameter tuning, but the same principle applies here as well). It also converges faster than random sampling, because the intervals make sure the data points are spread evenly over the sampling space [110]. However, this approach is very coordinate-dependent. There is also the problem that

the sampling should be spherically uniform, which complicates dividing the space in intervals (the problem of spherically uniform sampling is also discussed in A.4).

A good way to sample the potential for coordinates that will actually be useful for calculating trajectories, is to use trajectories to sample. This method we chose. However, to calculate these trajectories, one needs a potential V. Or, more specifically, the gradient of the potential. This gradient is then used to calculate the force $\boldsymbol{F} = \boldsymbol{\nabla} V$ on each nucleus, which is then used to compute the velocities and positions for the next time step using the velocity Verlet integration scheme. One option for the potential is to do an ab initio calculation that also includes a gradient computation for each time step, which Dalton is capable of. The following is an example of a .dal file that specifies that the analytical gradient of the CCSD potential should be calculated as well:

```
**DALTON INPUT
.RUN WAVE FUNCTIONS
**WAVE FUNCTIONS
.CC
*CC INP
.CCSD
.MAX IT
50
*DERIVATIVES
**END OF DALTON INPUT
```

For H_2–H_2 data set 1 (see C.1 for the details) the calculations at all time steps were part of the data set. For H_2–H_2 data set 2, the calculations done at each time step were discarded and every so many time steps a more expensive ab initio calculation was done to create the data set. However, because of the increased computational cost this was not a good option for the CO_2–CO_2 system. Another option is then to use an existing analytical potential, such as the Hinde potential (described in 2.1.7) for H_2–H_2 combined with a H–H potential, or the bond-bond potential combined with the Murrell-Guo-Zúñiga potential for CO_2–CO_2 (described in section 2.1.8), and compute the gradient of this potential numerically[8]. This was the approach used for the H_2–H_2 data set 3 and the CO_2–CO_2 data sets (see C.2).

The trajectories that used only ab initio calculations were calculated with the Python script `trajectory_sampling_2H2.py`. The trajectories that used an analytical potential were calculated using LAMMPS (version 3 Mar 2020), a molecular dynamics program that will be discussed in more detail in Chapter 4. For H_2–H_2, the Python script `lammps_trajectory_sampling_2H2.py` took care of generating the LAMMPS input scripts, importing the LAMMPS results and then creating Dalton input files and calling Dalton. For CO_2–CO_2 this was done with `lammps_trajectory_sampling_2CO2.py`. These two scripts run several LAMMPS trajectories in parallel and then several Dalton calculations in parallel using the Python multiprocessing library.

For the H_2–H_2 data sets 1 and 2 and the CO_2–CO_2 data set 1, the same kind of mistake was made; the sampling of the initial orientation of the each molecule was not uniform. For H_2–H_2 data sets 1 and 2, the initial orientation of the molecules was determined by sampling the angles θ and ϕ, which leads to relatively more configurations that are mostly oriented along or close to the z-axis. This is also discussed in A.4. This did not seem to lead to a less accurate fit for other configurations but is not good practice. For the CO_2–CO_2 data set 1, the initial orientation was determined by applying a random rotation to the molecule three times, once around each axis. This is also not perfectly spherically uniform. In H_2–H_2 data set 3 and CO_2–CO_2 data set 2 this was done correctly, and this is why parts of these data sets were used as the final test set.

When sampling for H_2–H_2 data set 3 and both CO_2–CO_2 data sets, the molecules were also given

[8]Thank you to dr. ir. Jesper Janssen for providing a C++ implementation of the bond-bond potential and for helping with the implementation of the Hinde potential.

an initial rotation. This rotation was around an axis perpendicular to the interatomic axis for H_2–H_2, but around a random axis for CO_2–CO_2. This is not strictly physical, and for trajectories that are used to approximate cross sections, more care was taken (see 4.2.3), but since the purpose of the sampling is just to get a good representation of the possible configurations, we decided this was good enough. A possible consequence of this is that there are more states with high velocity, because rotation around an axis close to the molecular axis has very low moment of inertia.

In all H_2–H_2 data sets, ab initio points with more than 20 iterations were removed, since this indicates that the CCSD calculation was very slow to converge and therefore was most likely not a good result (a multireference calculation might be more suitable for these configurations).

The hardware used for H_2–H_2 sampling was a computer with an Intel Core i7-4930K Processor @ 3.4 GHz with 6 cores and 64GB RAM. For CO_2–CO_2 sampling, two computers were used; one with an AMD Ryzen Threadripper 2990WX 32-Core Processor @ 3.0GHz and 96GB of memory, and one with a AMD Ryzen Threadripper 3960X 24-Core Processor and 64GB RAM.

3.2.2 Proximities

We tried several different kinds of proximities. The first kind of proximity matrices we are using contain proximities that are powers n of $1/r_{ij}$ (proximity matrix type A), with a different value of $n > 0$ for each proximity matrix:

$$\begin{pmatrix} 0 & 1/r_{12}^n & 1/r_{13}^n & 1/r_{14}^n \\ 1/r_{12}^n & 0 & 1/r_{23}^n & 1/r_{24}^n \\ 1/r_{13}^n & 1/r_{23}^n & 0 & 1/r_{34}^n \\ 1/r_{14}^n & 1/r_{24}^n & 1/r_{34}^n & 0 \end{pmatrix}, \tag{3.49}$$

with r_{ij} the distance between atom i and atom j.

To force the potential to zero for distances larger than r_0, the distances r_{ij} can be replaced by an expression involving a cutoff r_0, which results in proximity matrix elements ($i \neq j$)

$$p_{ij}^{(n)} = \begin{cases} (\frac{r_0}{r_{ij}} - \frac{r_{ij}}{r_0})^n, & r_{ij} \leq r_0, \\ 0, & r_{ij} > r_0, \end{cases} \tag{3.50}$$

(proximity matrix type B). This is a continuous function as long as $n > 0$, if $n = 0$ then p_{ij} goes to 1 at $r_{ij} = r_0$ and if $n < 0$ it goes to infinity. The derivative of p_{ij} with respect to r_{ij} is

$$\frac{\mathrm{d}p_{ij}}{\mathrm{d}r_{ij}} = \begin{cases} -n(\frac{r_0}{r_{ij}} - \frac{r_{ij}}{r_0})^{n-1} \left(\frac{r_0}{r_{ij}^2} + \frac{1}{r_0}\right), & r_{ij} \leq r_0, \\ 0, & r_{ij} > r_0, \end{cases} \tag{3.51}$$

which is continuous if $n > 1$. For this reason, all n that are used are more than 1.

These first two types of proximities both go to infinity as r_{ij} goes to zero. If this is not desirable, the following proximities can be used instead (proximity matrix type C)

$$p_{ij}^{(n)} = \begin{cases} \left(\frac{r_0}{r_0 + r_{ij}} - \frac{r_{ij}}{2r_0}\right)^n, & r_{ij} \leq r_0, \\ 0, & r_{ij} > r_0, \end{cases} \tag{3.52}$$

which goes to 1 at $r_{ij} = 0$, and similar to type B proximities is continuous if $n > 0$ and smooth if $n > 1$. These type A, B and C proximities are shown in Figure 3.10.

Another type of proximity we tested are the Morse proximities

$$p_{ij}^{(a)} = \exp(-r/a), \tag{3.53}$$

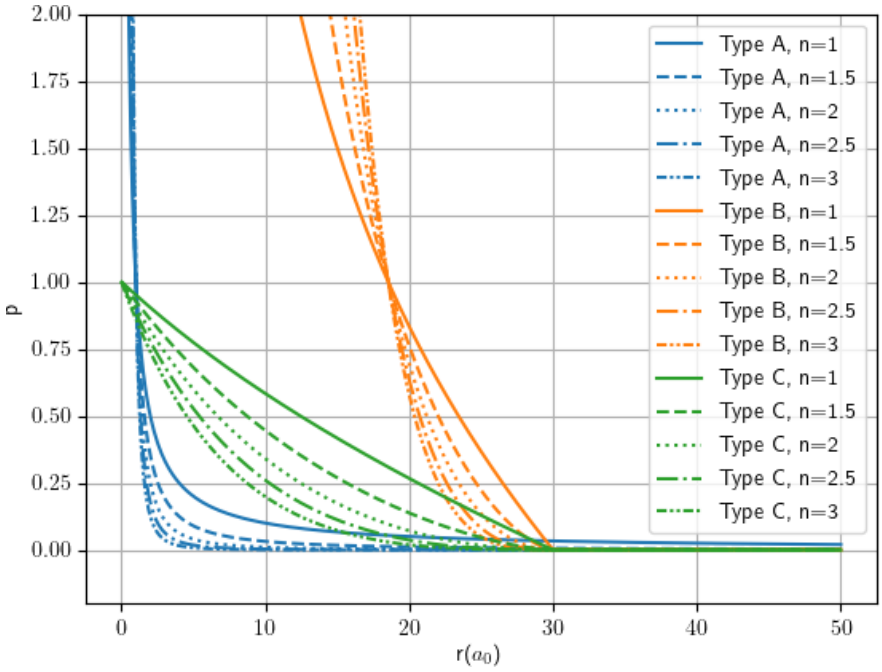


Figure 3.10: A, B and C proximities

which are shown in Figure 3.11. These are the kind of proximities that are used for the PIP method, for which a value of a of around 1 Å is a standard choice [25]. A Morse proximity with $a = 1$ Å is very similar to a C-proximity with $n = 13$ and a cutoff of 10 Å.

Another kind of proximities we tried are Gaussian proximities,

$$p_{ij}^{(a)} = \exp\left(-\zeta r^2\right), \tag{3.54}$$

which are similar to the radial part of the basis functions used for ab initio methods. They are shown in Figure 3.12. Because these functions have been optimized to describe the electron wave functions of a system, we hoped (in vain) they might also be suitable to describe the potential in the same system.

The last type of proximities we tested are the following sine/cosine proximities, which consist of a sine or cosine combined with the proximities B:

$$p_{ij}^{(k_a)} = \begin{cases} \sin((n+1) r_{ij}\pi/r_0) \cdot \left(\frac{r_0}{r_{ij}} - \frac{r_{ij}}{r_0}\right), & r_{ij} \le r_0, \\ 0, & r_{ij} > r_0. \end{cases} \tag{3.55}$$

$$p_{ij}^{(k_b)} = \begin{cases} \cos\left(\left(n+\frac{1}{2}\right) r_{ij}\pi/r_0\right) \cdot \left(\frac{r_0}{r_{ij}} - \frac{r_{ij}}{r_0}\right), & r_{ij} \le r_0, \\ 0, & r_{ij} > r_0. \end{cases} \tag{3.56}$$

$$\tag{3.57}$$

These sine/cosine proximities are smooth as long as n is an integer. See also Figure 3.13. Sine/cosine proximities are not strictly proximities since they are not monotonic and non-negative. However, for the purpose of linear regression or neural network inputs, this is not actually a problem. These proximities can also be combined with the proximities C if it is desired that they do not go to infinity near $r = 0$.

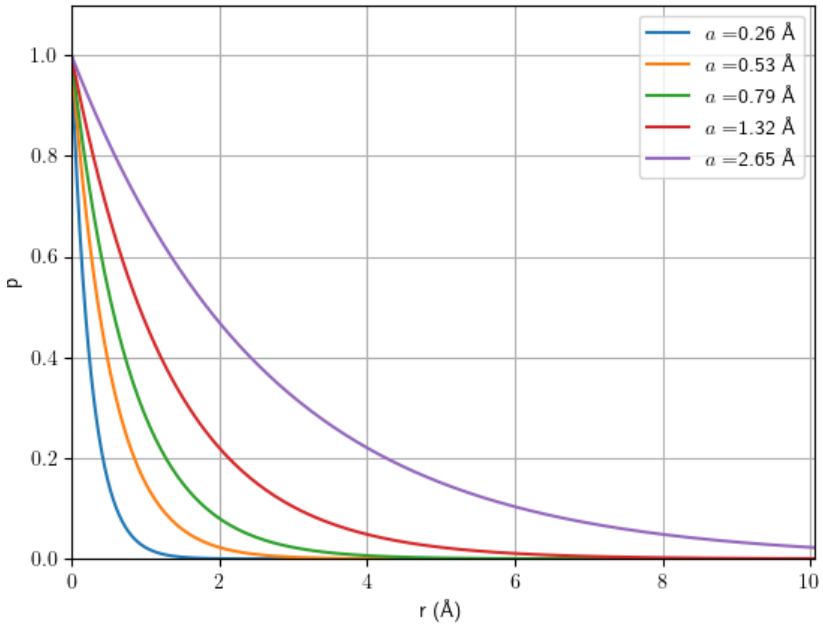


Figure 3.11: Morse proximities for $a = 0.5, 1.0, 1.5, 2.5, 5.0a_0$.

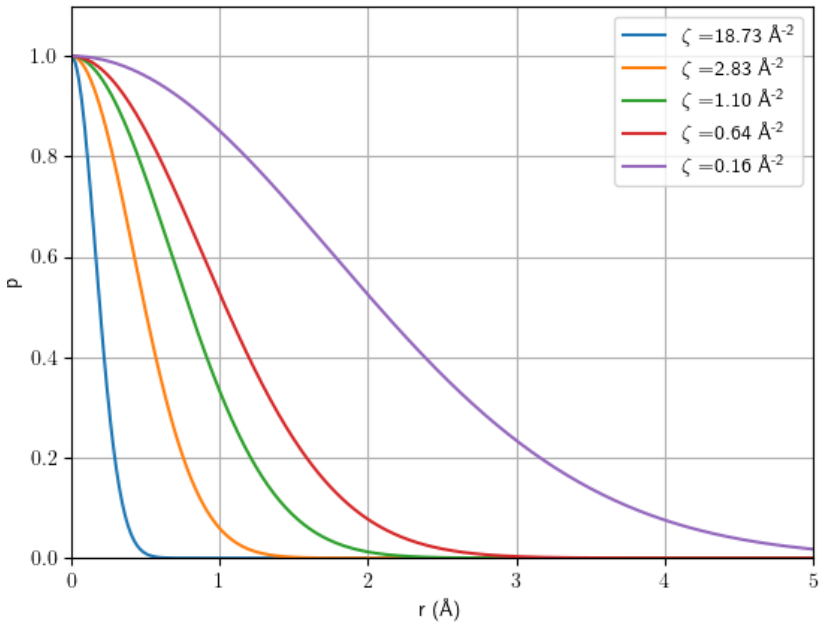


Figure 3.12: Gaussian proximities

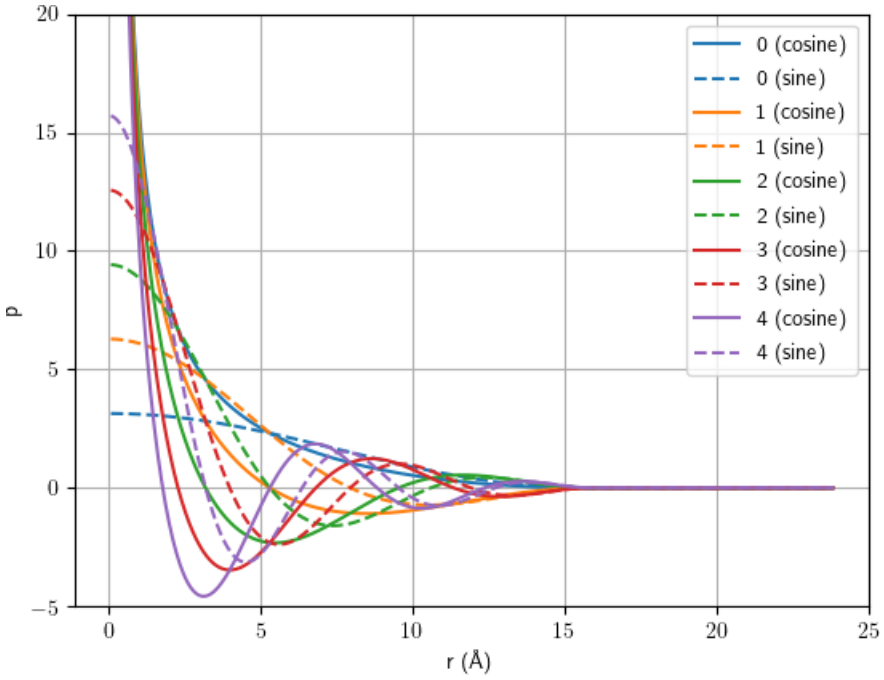


Figure 3.13: Sine-cosine proximities

The Gaussian proximities and sine/cosine proximities are similar to some of the atom-centered symmetry functions used by Behler [111, Fig. 3].

3.2.3 Neural network architecture

The neural network was implemented and trained using PyTorch (version 1.5.0), an open source machine learning Python library [112]. PyTorch can perform automatic differentiation (and therefore automatic back-propagation), through all kinds of operations. This gives a lot of flexibility when designing non-conventional networks, such as the network used for this project, which needed to include not only regular dense layers, but also the step from interatomic distances to eigenvalues. PyTorch also supports GPU computing with CUDA.

For each neural network attempt, 5-fold cross-validation is performed, which means the network is trained 5 times, each time a different 20% of the data is used for testing, and the network is trained on the other 80%. A separate testing data set was saved until the very end.

The data is divided per trajectory; meaning all points of one trajectory end up in the same set. This is done because adjacent points from the same trajectory can be very similar, which would make it easier for the network to estimate the energy of a point if it was trained on the adjacent point, which could lead to a higher testing performance that does not accurately represent how well the method actually works.

For the PME method, the eigenvalues were rescaled such that each feature had a variance of 1. If the variance is not 1, but for example 100, ideally the neural network should be able to compensate for that by simply learning that all the weights connected to this input node should be 100 times smaller, which will give the exact same result. However, this takes up training time, which means rescaling the data can help the training along.

The file `fleur_NN.py` contains some custom activation functions and the neural network classes. This

file is imported in 2 of the scripts that actually train a network, which are `NN_crossvalidation_charpoly.py` (for the PMI method for H_2–H_2) and `NN_crossvalidation_2CO2.py` (for the PME method for CO_2–CO_2). `NN_crossvalidation.py` is for the PME method for H_2–H_2 and includes its own network class definition. The file `fleur_NN.py` contains the following neural network classes, that each subclass the PyTorch Module class:

Net_eigenvals A network class for the eigenspectrum method. It is intended to be a general class (for any system) and is the only class as of yet that can handle different types of atoms. It includes the trainable weights for different pairs of atoms described in 3.1.8.

Net_PIPs A network class that takes PIPs as input, only for 4 identical atoms. It uses the 10 PIPs in (3.23).

Net_charpoly A network class for the PMI method for a system of N identical atoms. Uses proximities type B.

Net_charpoly_altproxes A subclass of **Net_charpoly** that takes a keyword to use different proximities.

3.2.4 Training the neural network

The Adam optimizer was used with a decreasing learning rate. Various settings were tried, but generally a really high number of epochs of around 15,000 epochs was used. Usually, less than a hundred epochs are used to train neural networks, for example, the very successful image recognition NN AlexNet was trained in just 90 epochs[113]. However, it seems it is not unusual to use a much larger number of epochs in PES fitting; the review article by Manzhos and Carrington [69] mentions several NN fits with tens of thousands of epochs[9]. The high number of epochs is probably a result of the fact that we want a high accuracy so we keep training as long as the loss is still decreasing. Since there is not a lot of noise in the data, training can go on for a long time before overfitting becomes a problem.

The learning rate started at 4×10^{-3} and was halved every 2000 or so epochs.

5-fold cross-validation was used to check if the results were consistently good. This means all the data except the final test set was divided into five 'folds', and the network was trained five times, each time leaving out one fold to use as validation data to check overfitting.

The test set for H_2–H_2 consists of 5000 trajectories from H_2–H_2 data set 3, which gives 49925 data points. Removing data points with more than 20 iterations removed 11 data points. Data points where the intermolecular distance was more than 10.6 Å were also removed, as at this distance there was no H_2–H_2 interaction anymore and the energy would be too easy to predict, leading to an inflated test score. Removing these removed 10482 data points (in the future, it would be smarter to implement a check during sampling such that these points are not used for ab initio calculations in the first place).

The resulting 39432 data points were the entire test set. There was also a reduced test set, from which we attempted to remove possible outliers. This meant keeping only configurations with $4 < R < 20$ bohr and all $0.8 < r < 7.56$ bohr, and making sure r_{12} and r_{34} were the smallest distances (meaning, two atoms from different molecules are not closer together than two atoms from the same molecule). This is done because these configuration space around these outliers was sampled very sparsely and its performance there would often make the overall performance look very bad, even though the models were pretty accurate on the configurations that *were* sampled thoroughly.

Cross-validation sets were often also reduced in various ways, but less consistently. Table 3.3 gives an overview.

[9] They suggest using the Levenberg-Marquardt algorithm instead of gradient descent or the Adam optimizer, which for them brought it down to 500-2000 epochs. This was not implemented in this project, but could be worth a try in the future.

Table 3.3: Data sets that the various methods were trained and tested on. (Testing was consistent, to make comparison between methods at least somewhat accurate, the crossvalidation data is not as consistent, because a lot of experimentation was done)

		Data set	Size	Kept configurations where
All methods, test set	entire	3 (part)	39,432	$R < 20$ bohr, it ≤ 20
All methods, test set	reduced	3 (part)	37,809	$R < 20$ bohr, it ≤ 20, 0.8 bohr $<$ r $<$ 7.56 bohr, r12 and r34 are smallest
PME NN method, cross-validation		2	23,908	$R > 4$ bohr, it ≤ 20
PIP method, cross-validation	entire	2	7,291	$R < 20$ bohr, it ≤ 20
PIP method, cross-validation	reduced	2	6,906	$R < 20$ bohr, it ≤ 20, 0.8 bohr $<$ r $<$ 5.67 bohr, r_{12} and r_{34} are smallest
PMI method, cross-validation	entire	2	7,291	$R < 20$ bohr, it ≤ 20
PMI method, cross-validation	reduced	2	6,906	$R < 20$ bohr, it ≤ 20, 0.8 bohr $<$ r $<$ 5.67 bohr, r_{12} and r_{34} are smallest
PMI NN method, cross-validation		2	6,906	$R < 20$ bohr, it ≤ 20, 0.8 bohr $<$ r $<$ 5.67 bohr, r_{12} and r_{34} are smallest

Since the last layer had no activation function (or, equivalently, the linear activation function $f(x) = x$), it can also be trained using a least squares fit. Because a least squares fit gives the result that minimizes the average squared error (and therefore also the root mean square error), this immediately gives the optimal weights for the connections from the last hidden layer to the output layer, given all other weights. This gives a big improvement in prediction accuracy in very little time, if the results were not properly converged yet. If they are, this fit should make only a small difference. This makes the least squares fit a good check of the convergence. Unfortunately, using a least squares fit for the last layer was very prone to overfitting, so unless the last layer contained only a few nodes, this trick is only used as a convergence check.

The hardware that was used for most of the training was a computer with an Intel Core i7-4930K Processor @ 3.4 GHz with 6 cores and 64GB RAM. Later on, training was also done on a GeForce RTX 2080 SUPER GPU. The GPU only sped up training by a factor of about 2, which was less than expected, so perhaps with some careful optimization the training time can be brought further down.

3.2.5 Hyperparameter optimization

For the PME method for H_2–H_2, an expansive search for the hyperparameters was performed, mostly using data set 1. Some tables and figures showing different hyperparameters can be found in D.

Using 3 eigenvalues per matrix instead of 4 performs equally well. Adding more proximity matrices gives no extra advantage, which shows that the eigenvalue representation is a very good way to summarize the configuration.

For the activation function softplus and softsign were about equally good. The half-square activation was a bad choice, because the square part did not handle outliers well.

It was found that a bigger dataset needs fewer epochs to converge, which makes sense because the number of updates to the network is equal to `number of batches` $\times$ `number of epochs`.

Several different combinations of exponents for the proximity matrices were tried for type A proximities. It made sense from a physics point of view to try exponents such as -6 and -2, because the Coulomb force scales with r^{-2}, dipole-dipole forces scale with r^{-3} and London dispersion forces scale with r^{-6}. However, these exponents did not perform any better or worse than similar exponents such as -5 or -1.5. The only noticeable differences were that higher exponents performed worse, presumably because they amplify outliers when the proximities are type A or B. A possible reason for this could be numerical issues such as underflows.

It turned out that when using proximities type A, adding bias parameters improves the result, even though they should not be strictly necessary, presumably because more parameters means better fitting capabilities. However, proximities type B (with no bias) turned out to work better. Proximities C were never used with a PME NN. For the PMI NN method, proximities B, C and Gaussian ones were tried, of which the proximities C performed the best.

3.2.6 PIP method

The PIP method was implemented up to 7^{th} order for the H_2–H_2 system in Python as well, in the script `Braams_auto_7.ipynb`.

Creating all unique PIPs up to a certain order is non-trivial, which is why a brute force method was used. For each order, all possible sets of 6 non-negative integers that sum to that order were generated (for example: $0,0,0,0,1,1$ and $0,0,0,0,0,2$ for order 2), and then for each of those sets, we created a set of all permutations of these integers. For example, for $\{0,0,0,0,0,1\}$ this was

$$\{[0,1,0,0,0,0],[0,0,1,0,0,0],[0,0,0,1,0,0],[0,0,0,0,1,0],[0,0,0,0,0,1]\}\,.$$

In each permutation, the numbers correspond to the exponents for the proximities $p_{12}, p_{13}, p_{14}, p_{23}, p_{24}, p_{34}$. For each of these permutations, a set of permutations that correspond to permutations of the *atoms* was

created. Each of these sets then corresponds to one PIP, but there are a lot of identical PIPs, which are removed. This means that the computational complexity is *far* worse than the $\Theta(q^{1+1/n})$ indicated in Table 3.2. It is possible to do this more efficiently, for example by using graph generating software `nauty`[100].

3.2.7 PMI method

The PMIs are used for both linear regression as well as NN inputs. The former is implemented in the Python script `linear_regression.py` and the latter in `NN_crossvalidation_charpoly.py`. For the linear regression the PMIs $\mathrm{tr}(\mathbf{P}^k)$ and $\mathrm{tc}(\mathbf{P}^k)$ are used, with k from 2 to various orders (the maximum order is experimented with) for the $\mathrm{tr}(\mathbf{P}^k)$ PMIs. For the $\mathrm{tc}(\mathbf{P}^k)$ PMIs, $k = 1$ is also used ($\mathrm{tr}(\mathbf{P}^k)$ is zero when $k = 1$). For the NN method, only the PMIs $\mathrm{tr}(\mathbf{P}^k)$ with $k = 2, 3, 4$ are used, because higher order terms can always be written as a product of these PMIs.

3.3 Results H_2–H_2, PME method

The PME method described in 3.1.8 was the main focus of this project.

In the end, the neural network with the best RMSE in cross-validation was chosen and this network was used on the test set. This best neural network uses proximities type B (see (3.50)) with n =2 and 3 and a cutoff of 30 Å. It has two hidden layers of 200 nodes each that use the softplus activation function. The output layer had no activation function (or, equivalently, the function $f(x) = x$).

This network (NND_48) had an average validation RMSE of (5.9 ± 1.3) meV during cross-validation. On the test set, this network managed an RMSE of (19.58 ± 0.66) meV on the entire test set, and (6.50 ± 0.66) meV on the reduced test set. The performance on the reduced test set is better than on the entire test set, which makes sense, because this network was trained on data more similar to the reduced test set (see Table 3.3 for specifics). The performance on the reduced test set is still slightly worse than the performance on the validation set, which is most likely because the training/validation set (data set 2) contained too many 'easy' configurations (large R). Figure 3.14 shows a crossplot of the neural network results for both the entire as well as the reduced test set. Figure 3.15 shows histograms of the error of the test set of this network. It shows that the goal of getting an accuracy better than the thermal energy at room temperature (approximately 25 meV) is achieved for the vast majority of configurations.

The accuracy is comparable to other neural network approaches to PESs. For example, the PES of amorphous Li_3PO_3 by Li et al. [23] using an approach that sums the contributions of each atom (see (3.13)) inspired by work by Behler [111] reached an RMSE of 5.5 meV/atom. The PESs of H + H_2 and Cl + H_2 systems by Jiang and Guo using PIPs as inputs (see 3.1.6) reached a RMSE of 3.6 and 4.2 meV, respectively. However, Kamath et al. [114] manage to create a NN PES of formaldehyde (H_2CO) with a really low RMSE of 0.14 meV, so it is unlikely our results are state-of-the-art (although it is unclear how meaningful comparisons of RMSE values of different systems really are). Their results used up to 2500 different configurations, which covered a range of energies of approximately 0 – 2 meV, using sampling similar to Boltzmann sampling. This is a smaller range of range that what we are working with, but also a smaller training set.

3.3.1 Comparison to ab initio potential

Figure 3.16 shows the neural network potential for one H_2-molecule, varying the bond length and compares it to the Dalton results [10]. It shows that the neural network does a very good job at recreating the

[10] Because the neural network always needs the same number of inputs, this curve was obtained by using the coordinates of 4 H-atoms, with the two molecules very far apart (beyond the cutoff of 30 Å) and varying the bond length of one of the molecules while keeping the second molecule at the equilibrium bond length. The bond energy of the second molecule is then added to

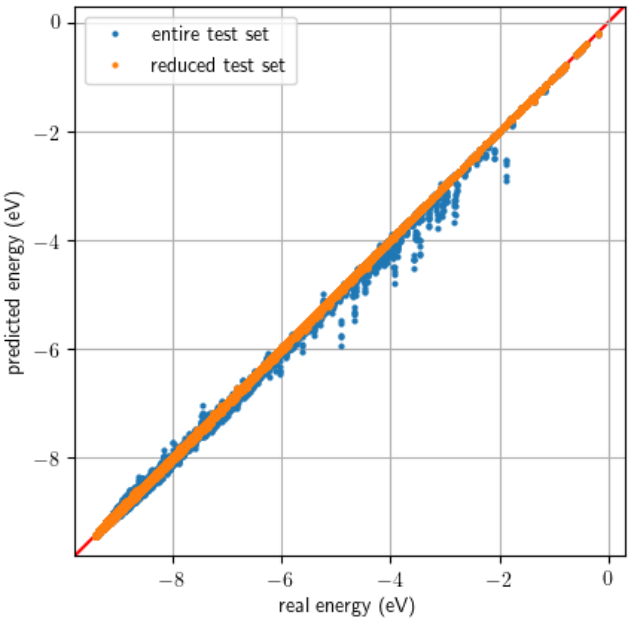


Figure 3.14: Crossplot of the results of the best neural network using the PME method.

intramolecular potential.

Figure 3.17 shows the neural network potential for the 6 configurations of two H_2-molecules in 2.6 (H, X, T, L, S45 and S60 configurations), and compares it to the Dalton results. 2 times the H_2 bond energy was added to normalize the data. It shows that while the potential is qualitatively approximately correct – the steep potential wall and the small Van der Waals well are both present – the location of the potentiall wall is slightly off and the Van der Waals well is often exaggerated. It is unclear why this happens.

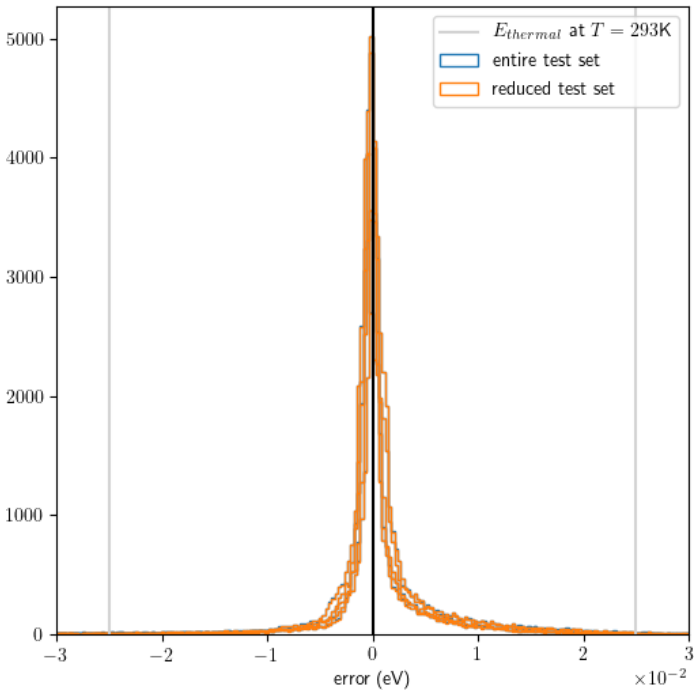
It also shows that the neural network has an error of about 0.5 to 2 meV in the estimated bond energy, since there is a shift of about 1 to 4 meV in the energy at large R. The potential is forced to zero when using proximities type B, but only when all atoms are further apart than the cutoff, not when only two molecules are beyond the cutoff. For this potential that means it goes to a constant value for $R > r_{\text{cutoff}}$.

3.3.2 Gradient

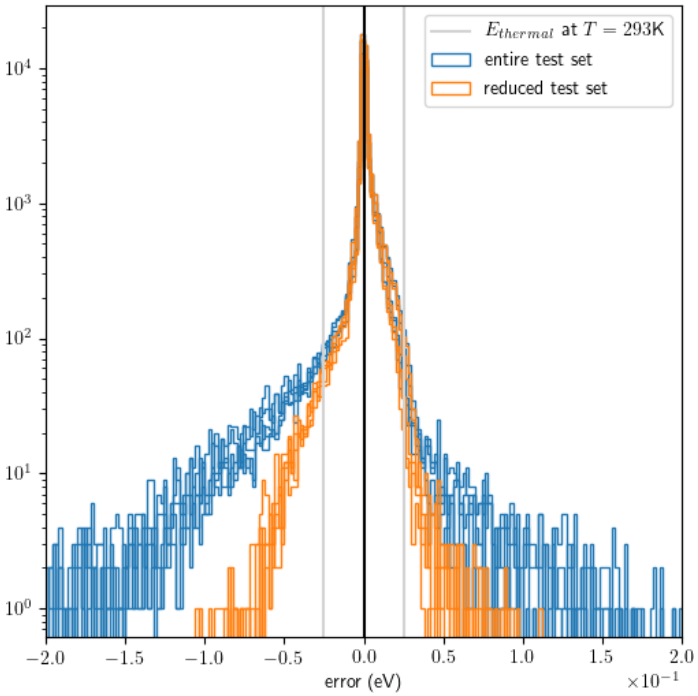
In the end the goal of this potential modelling is to use the potential for trajectories, which means it is not the potential itself, but the gradient which is of interest. The gradient is used to calculate the forces the particles experience. Dalton can also calculate an analytical gradient. One trajectory (the trajectory of H_2–H_2 data set 2 with the smallest R in it, in order to pick the most difficult trajectory) was chosen and was entirely calculated with aug-cc-pVTZ, both the potential as well as the gradient. This trajectory is shown in Figure 3.18. One of the 5 folds was trained on points from this trajectory, which is why results from all 5 folds are shown.

The analytical gradient of the neural network potential is calculated by differentiating the entire chain from Cartesian coordinates to proximity matrix to eigenvalues to the neural network to the predicted energy value F. Figure 3.19 is a crossplot of the magnitude of the gradient on each atom predicted by the neural network against the real (ab initio) gradient. It shows that in general, the magnitude of the gradient is

the predicted energy.



(a)



(b) Zoomed out, with logarithmic y-axis to show outliers.

Figure 3.15: Histograms (one per fold) of the results of the best NN that uses the PME method on the test data set.

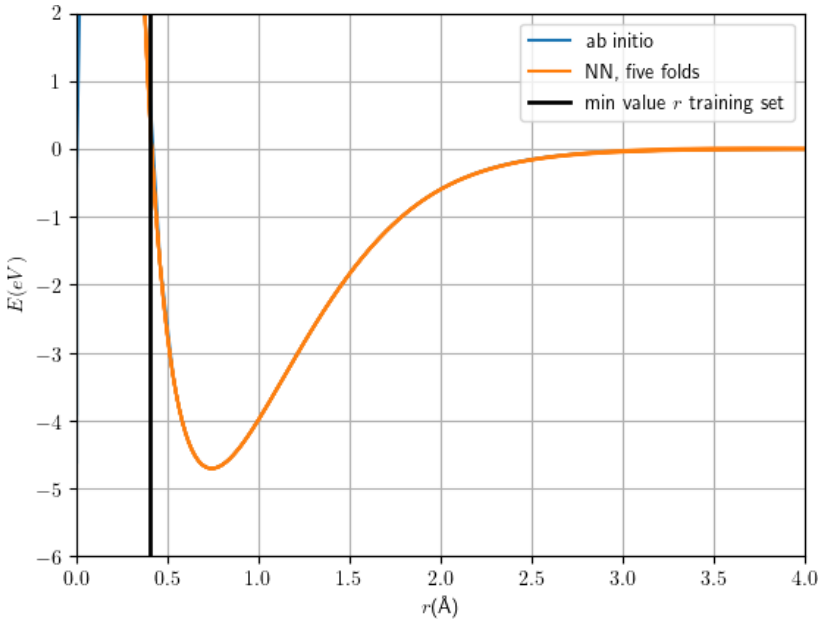


Figure 3.16: The neural network potential (all five folds) for one H_2-molecule when varying the bond length, compared to the Dalton results

very accurate. Figure 3.20 shows a histogram of angle between the gradient vector predicted by the neural network and the gradient vector from the ab initio data on one atom ('gradient vector' refers to the 12D gradient vector). It shows that the vast majority of the time, this angle is very small, which means that the direction of the gradient vector is also very accurate. There are some outliers where the angle is pretty large and therefore the direction is pretty inaccurate. However, Figure 3.21 shows that these large angles only happen when the gradient is really small.

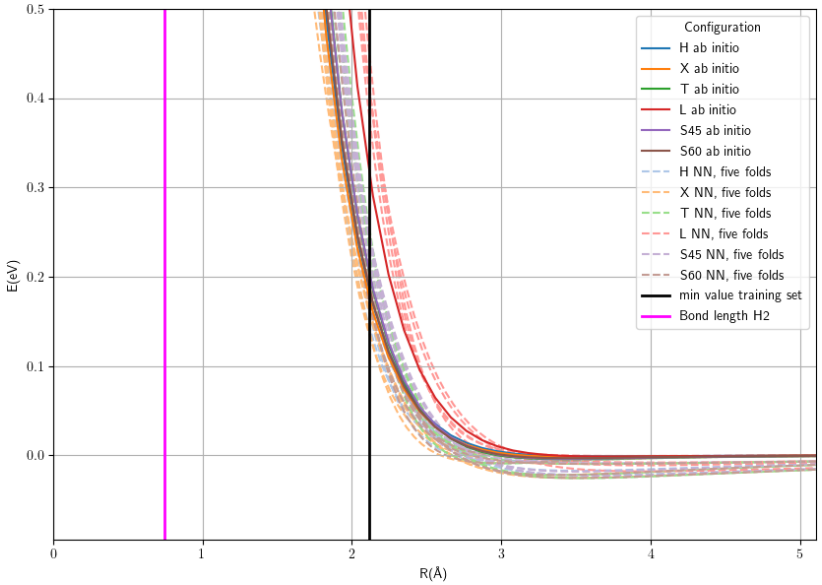
The numerical gradient is also calculated, using with the central difference method, with various dx. Figure 3.22 shows the average relative error in the gradient (the magnitude of the error vector divided by the magnitude of the gradient vector) for the analytical gradient and for the numerical gradient. It shows that strangely enough, for very specific values of dx, the numerical gradient is actually slightly more accurate than the analytical gradient.

3.3.3 Effect of crossing eigenvalues

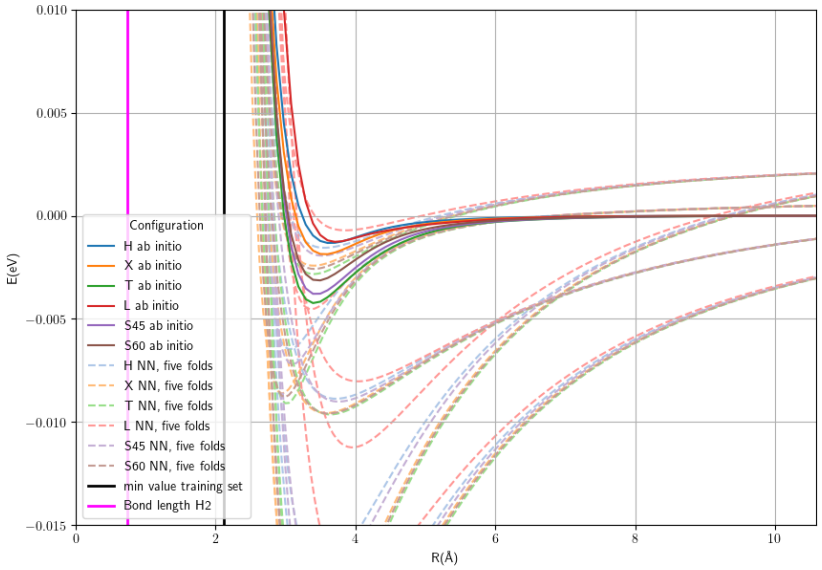
The motivation for using the eigenvalues as a permutationally invariant representation is the avoided crossing phenomenon, which means eigenvalues can only cross when the configuration has a certain kind of symmetry. When eigenvalues cross and are then sorted, there will be a kink in the curve of sorted eigenvalues, which will lead to a kink in the potential.

We expected that the symmetric configurations where this is a problem would be very rare, however this turned out to be wrong for two reasons: 1) when two parts of the system don't interact, the matrix becomes a block matrix and a symmetric state becomes much more likely, 2) even when the crossing *is* technically avoided, the separation between the eigenvalues can be so small that in practice they might as well cross.

With the proximities of type B and C this is the case when the intermolecular distance is larger than the cutoff distance. With proximities A, the Morse proximities and the Gaussian proximities, the molecules



(a)



(b) The Van der Waals well (x-axis is zoomed out, y-axis is zoomed in)

Figure 3.17: Comparison of the intermolecular potential by the neural network (all five folds) and the Dalton results for the 6 configurations shown in 2.6, while varying the distance between the centers of mass of the molecules.

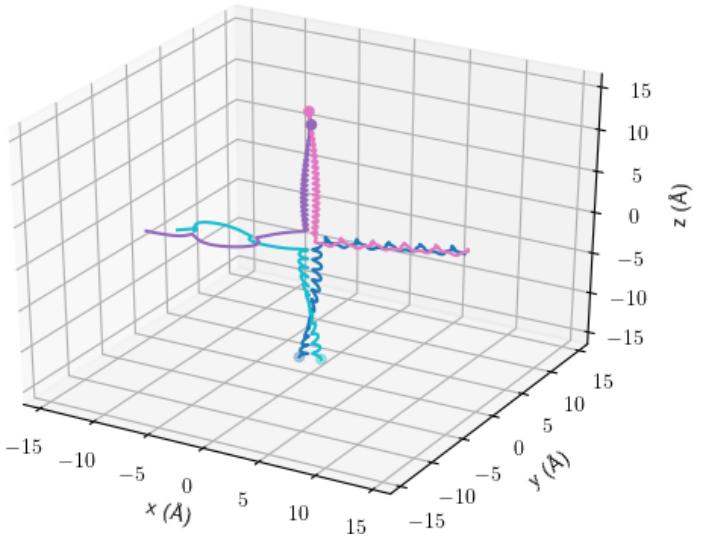


Figure 3.18: Trajectory chosen as test trajectory. Two H_2-molecules exchange an atom. Each colored curve is the path of one atom and the starting point of each atom is marked with a dot. The pink/purple molecule starts at the top, moving down and the cyan/blue molecule starts at the bottom, moving up.

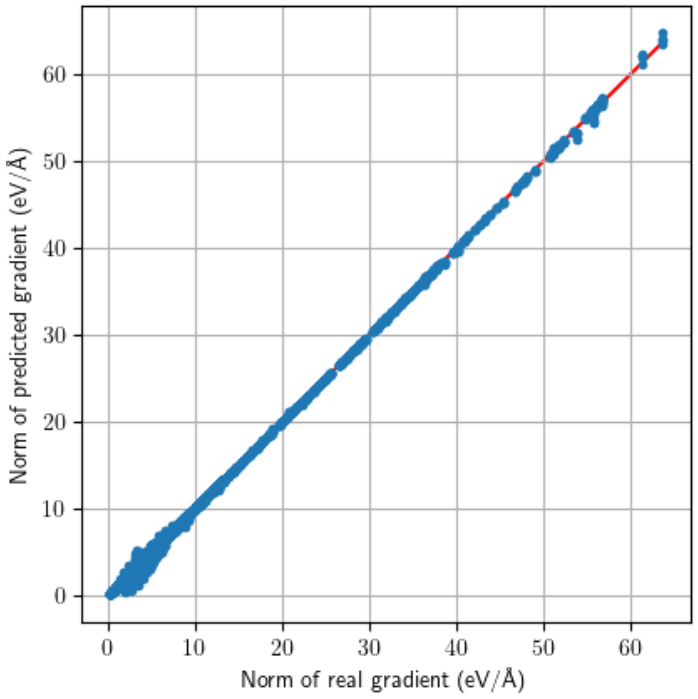


Figure 3.19: The magnitude of the predicted gradient compared to the magnitude of the ab initio gradient

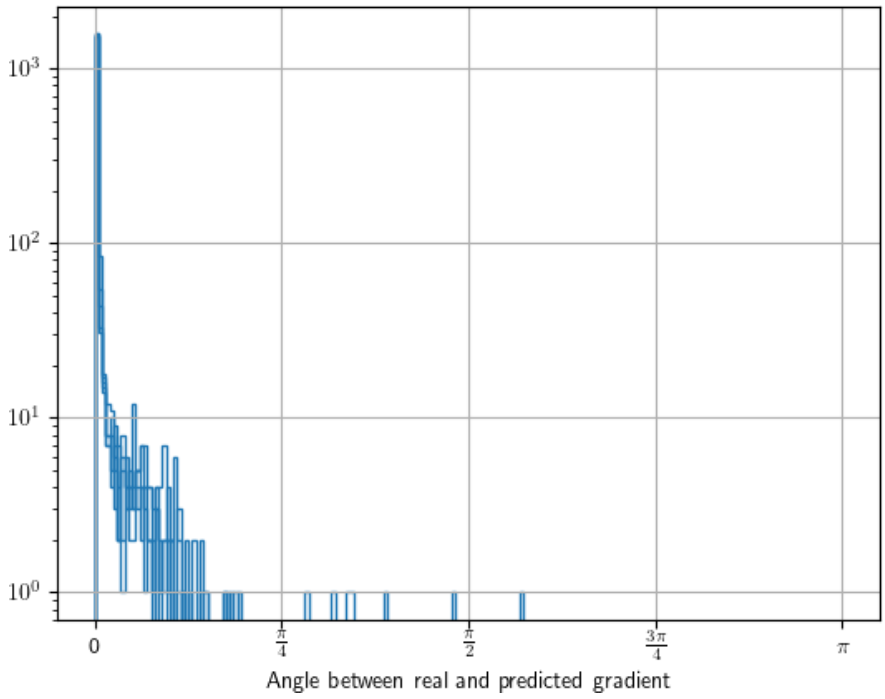


Figure 3.20

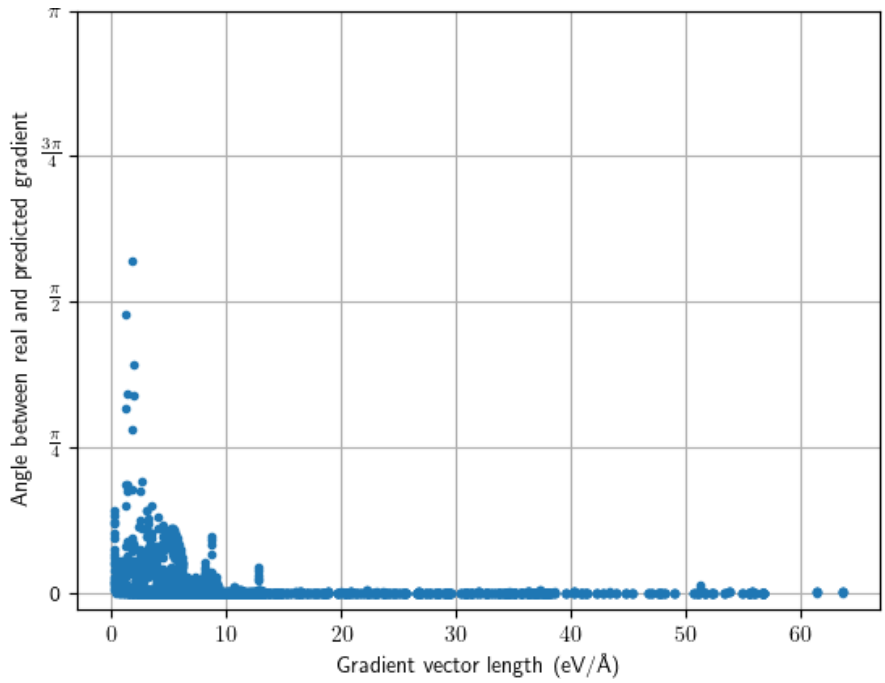


Figure 3.21: A histogram of the angle between the predicted gradient vector and the gradient vector from the ab initio data, plotted against the magnitude of the gradient vector, for all five folds

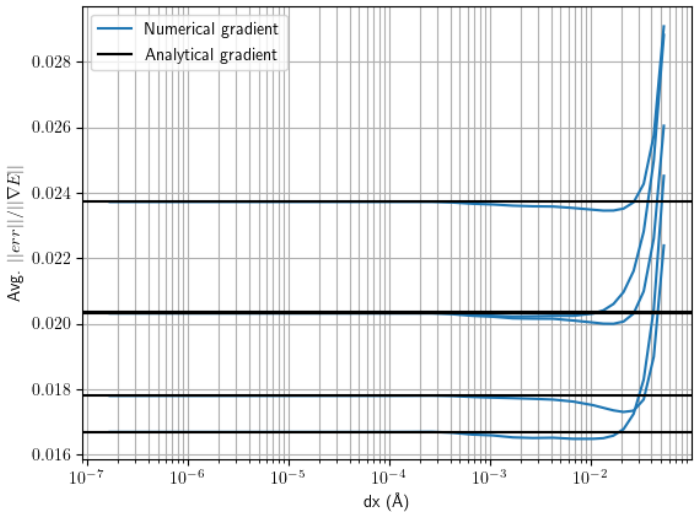


Figure 3.22: Numerical gradient compared to the analytical gradient, for each fold

are never technically completely non-interacting, but the interaction does become negligible, resulting in a very small separation of the eigenvalues. For the B and C proximities, this also happens already before the cutoff.

For H_2–H_2, this means there is an avoided or just-barely-avoided eigenvalue crossing every time both molecules have the same bond length when they are some distance apart. Figure 3.23 shows this occurring during the same trajectory that was discussed in the previous subsection (Figure 3.18). It also shows the absolute value of the dot product between an eigenvector and the same eigenvector at the previous time step. Since each eigenvalue has a unique corresponding eigenvector, and eigenvectors belonging to different eigenvalues are orthogonal, the dot product can be used to check if the eigenvector corresponds to a different eigenvalue than in the previous time step. The network used for this plot and others in this subsection was a different one than the best one discussed in previous parts of this chapter, but with very similar settings. The cutoff was 30 Å, which means R was within the cutoff the entire time.

Figure 3.24 shows that the separation of the eigenvalues in an avoided crossing gets very small very quickly. For example, if the intramolecular proximities are 1 and the intermolecular proximities are approximately 0.1, the minimum eigenvalue separation is about 0.02. If the intermolecular proximities are approximately 0.01, the separation is about 0.002. The ratio between intramolecular proximities and intermolecular proximities gets much smaller than 0.01 during a trajectory. This is shown by Figure 3.25, which shows the value of the proximities (type B, cutoff 30 Å, $n = [2, 3]$) during the trajectory in 3.18.

We caught this problem pretty late in the project, because the results for H_2–H_2 still end up smooth. However, this is purely a result of the neural network simply being trained very well and learning to smooth out the kinks. Figure 3.23 shows this by showing the gradient error along a trajectory for networks that are in various states of convergence near a near-avoided crossing. Figure 3.23 shows the entire trajectory.

At first we suspected that these kinds of crossings are specific to a four-atoms 3D trajectory, and that in 2D or with more atoms the chance of a crossing decreases exponentially. However, this turned out not to be the case. Figure 3.26 shows a 2D H_2–H_2 trajectory, and Figure 3.27 shows that here eigenvalue crossings also occur. This trajectory was created by removing the y component from the trajectory in Figure 3.18. Figure 3.26 shows a 3D 5H trajectory, and Figure 3.29 once again shows that eigenvalue crossings still occur, although not exactly when $r_1 = r_2$. This trajectory was created from the trajectory in Figure 3.18 by adding a fifth 5th atom that moves in a straight line (grey line in figure). For both trajectories no new molecular

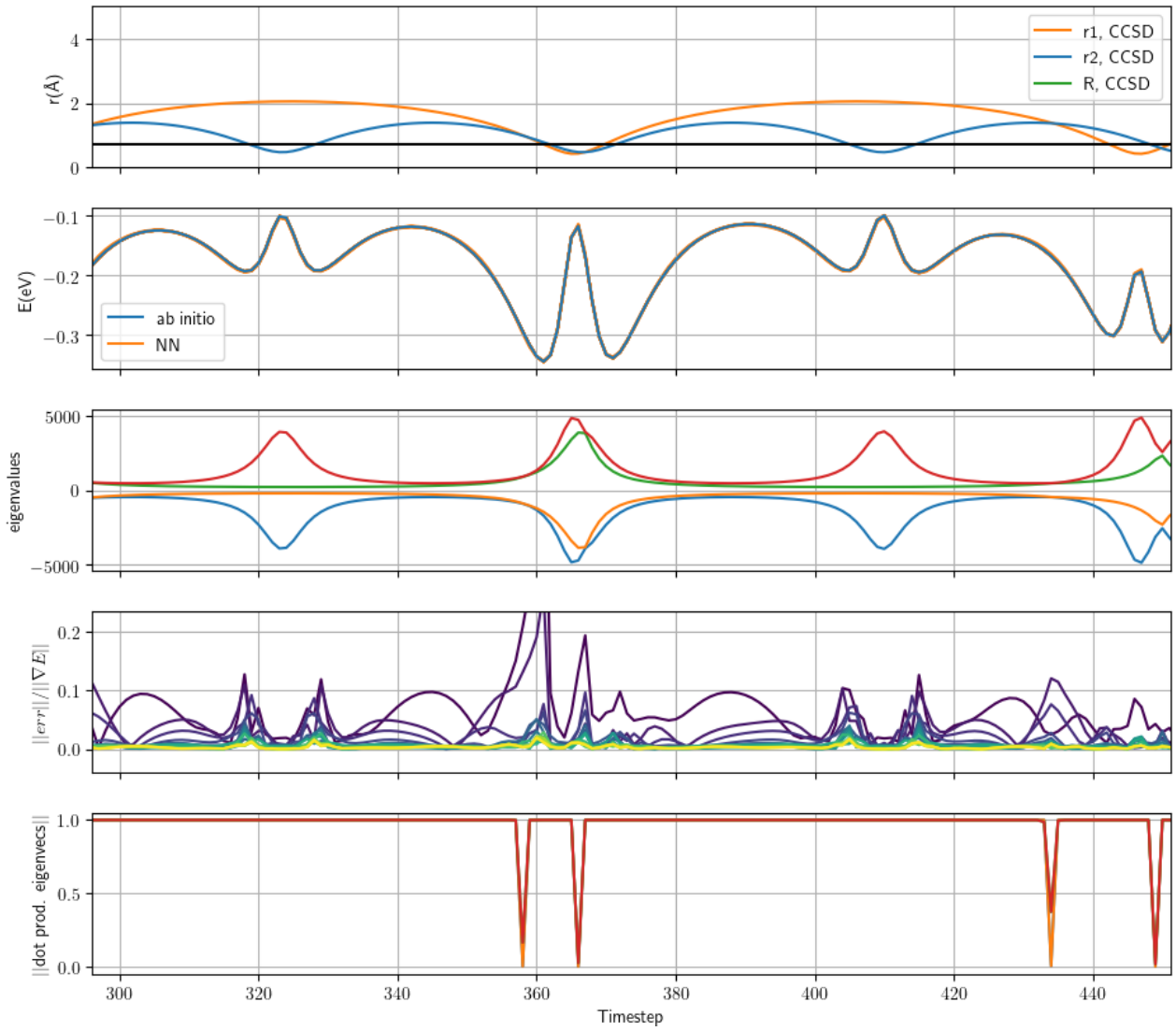


Figure 3.23: The bond lengths r_1 and r_2, the potential, the eigenvalues, the error in the gradient and the absolute value of the dot product between an eigenvector and the same eigenvector in the previous time step. The colors in the gradient error plot gradually go from dark purple to blue to green to yellow and indicate how long the network was trained; the darkest purple line corresponds to a neural network that was trained for only 100 epochs, the yellow line to one that was trained for 7000 epochs and the other colors something inbetween.

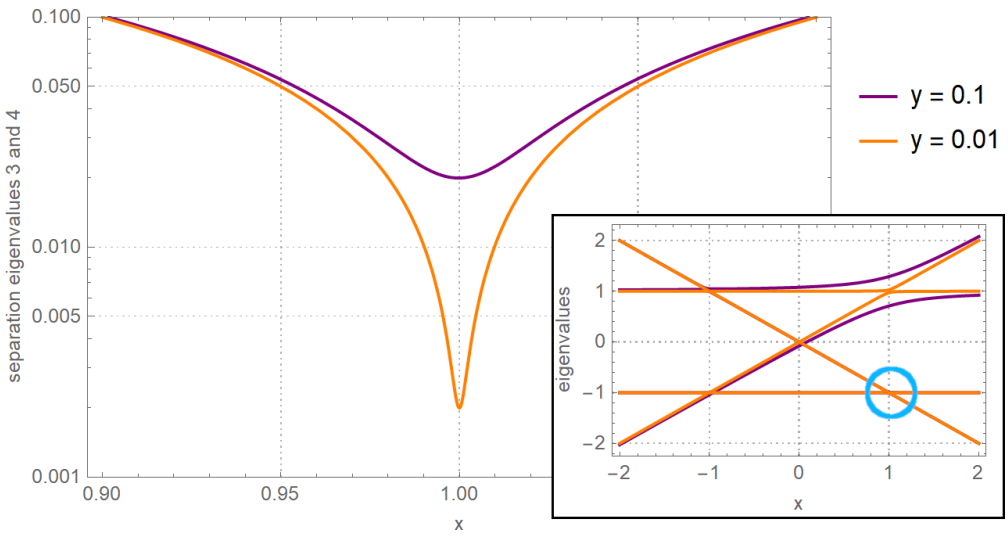


Figure 3.24: The separation between the lowest two eigenvalues of the asymmetric matrix

$$\begin{pmatrix} 0 & 1 & 1.4y & 1.5y \\ 1 & 0 & 1.3y & 1.6y \\ 1.4y & 1.3y & 0 & x \\ 1.5y & 1.6y & x & 0 \end{pmatrix},$$

for two values of y. The inset shows the eigenvalues as x is varied, with the blue circle marking the avoided crossing whose separation is plotted.

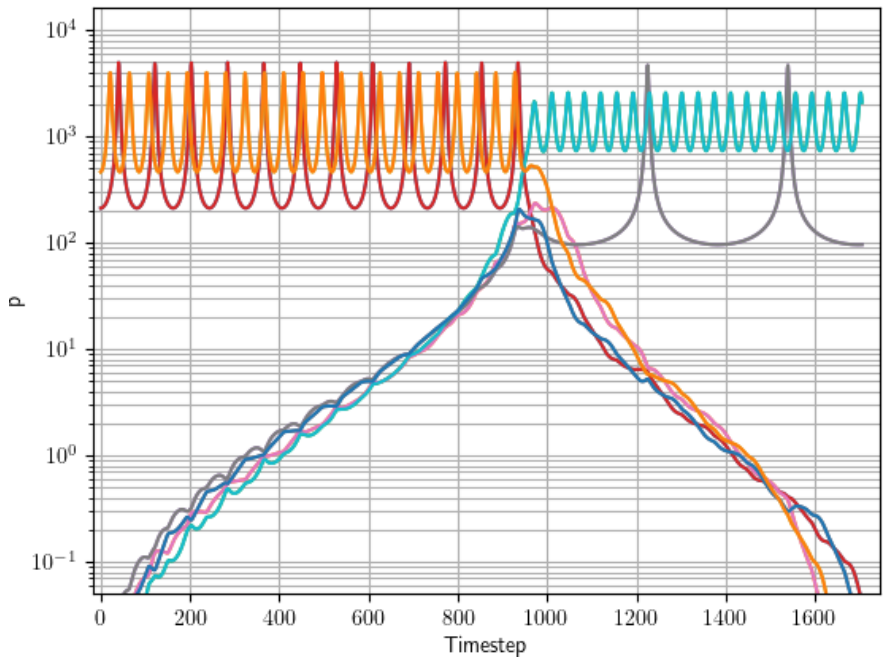


Figure 3.25: The values of the proximities (type B) over the course of a trajectory

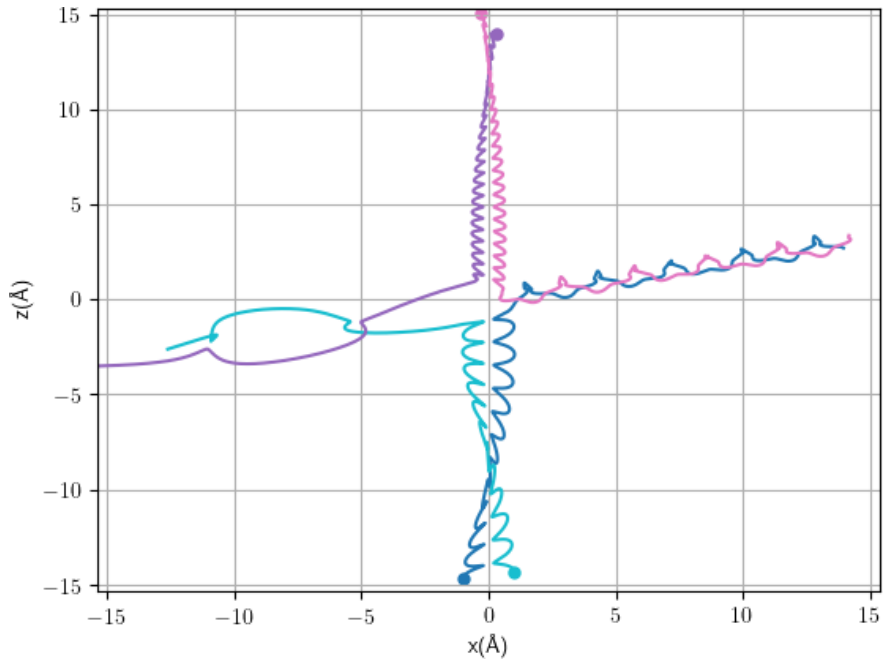


Figure 3.26: A H_2–H_2 trajectory in 2D.

dynamics or ab initio calculations were done. The results were interpolated to make the time step 100 times smaller.

3.4 Results H_2–H_2, PIP method

The PIP method described in 3.1.6 to 7^{th} order, which has 120 coefficients for H_2–H_2, was applied to several different proximities (this is one more than in Table 3.1, because there is also a term of order 0, which is a constant). The results are in Table 3.4. The best result was achieved with Morse proximities with $a = 2\,a_0$. The proximities A and B did not perform very well, most likely because the fact that they go to infinity near $r = 0$. The proximities C however reached an RMSE of 8.5 meV, which is in the same order of magnitude as the Morse result.

Since the best results were obtained with the Morse proximities with $a = 2\,a_0$, these proximities were used for the test set. The results are in Table 3.5. Figures 3.30 and 3.31 show a crossplot and histograms, respectively, of the error of the test set, for the fits that were trained on the entire cross-validation set.

Table 3.6 shows the first 10 coefficients in one of the least squares fits, their corresponding PIPs and the average value of each PIP, as well as the 5 PIPs with the biggest contributions of all 120 PIPs. See also Figure 3.8, which shows the corresponding graphs for the four 3^{rd} order PIPs. It shows that of the first 10 PIPs, almost all the PIPs corresponding to open walks and closed walks seem to be more important than the disconnected graphs and the star graph (a star graph is a graph whose nodes all have only one edge connected to them, except for one internal node that connects to all nodes). For both the 2^{nd} and the 3^{rd} order PIPs, the disonnected graph has the smallest contribution. This makes sense, since then you are trying to sum two different particle interactions. It also shows that the 5 most important PIPs are all open or closed walks. This is a promising sign for the methods that use the PMIs, since the PMIs include only the open and closed walks.

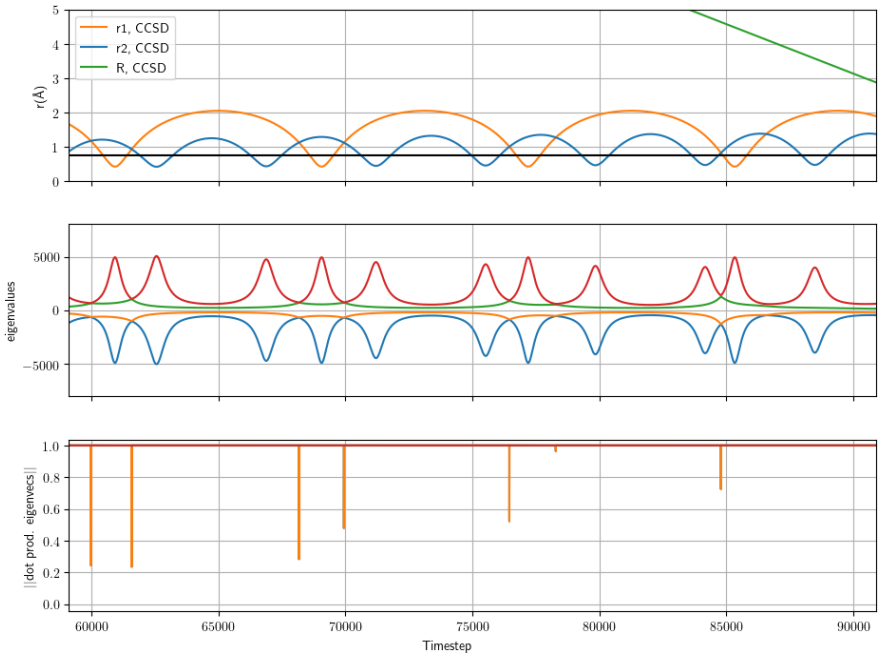


Figure 3.27: The bond lengths r_1 and r_2, the eigenvalues and the absolute value of the dot product between an eigenvector and the same eigenvector in the previous time step, for a 2D H_2–H_2 trajectory

Table 3.4: The results for the PIP method on the validation data for each proximity type. For comparison, the RMSE achieved by simply always predicting the mean is also given.

Type of proximities	Parameter value n or a (a in bohr)	RMSE validation (meV)
Proximities A	2	170.2 ± 161.1
Proximities B (cutoff=15.9Å)	2	1764.7 ± 386.8
Proximities C (cutoff=10.1Å)	2	61.5 ± 2.8
Proximities C (cutoff=10.1Å)	13	24.92 ± 23.6
Proximities C (cutoff=10.1Å)	15	52.5 ± 45.2
Proximities C (cutoff=15.9Å)	15	8.5 ± 3.0
Morse	2	4.7 ± 0.4
Morse	5	19.3 ± 1.5
Always predict mean	-	1815. ± 64.

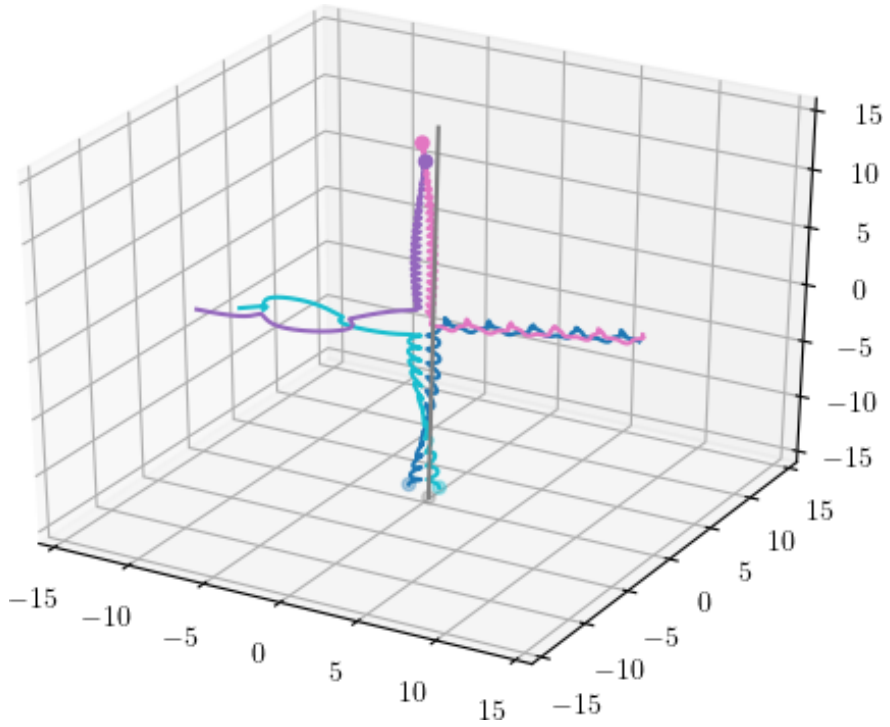


Figure 3.28: Trajectory with 5 H-atoms.

Table 3.5: Results of the PIP method on the validation data and on the test data. Morse proximities met $a = 2\,a_0$.

	RMSE on entire test set (meV)	RMSE on reduced test set (meV)
Trained on entire cross-validation set	10.70 ± 0.33 meV	5.94 ± 0.41 meV
Trained on reduced cross-validation set	26.40 ± 3.67 meV	4.37 ± 0.14 meV
	Validation RMSE	
Trained on entire cross-validation set	11.94 ± 2.93 meV	
Trained on reduced cross-validation set	4.71 ± 0.36 meV	

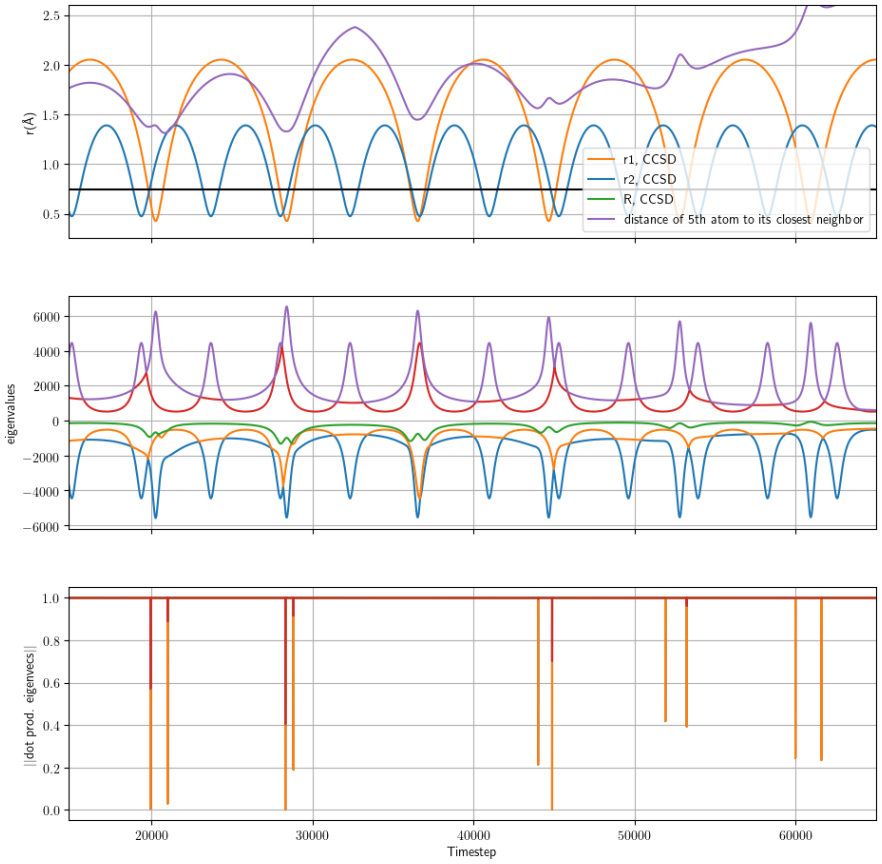


Figure 3.29: The bond lengths r_1 and r_2 and the distance of the 5^{th} atom to its closest neighbor, the eigenvalues and the absolute value of the dot product between an eigenvector and the same eigenvector in the previous time step, for a trajectory with 5 H-atoms

Table 3.6: PIPs up to order 3 with their corresponding coefficient, as well as the average value of the PIPs applied to the training set, and the average contribution to the potential (which is the average value times the coefficient). Morse proximities with $a = 2\,a_0$, trained on one fold of the full crossvalidation set.

Contributions of PIPs up to order 3				
PIP	Graph type	Avg. value	Coefficient value	Avg. contribution (E_h)
p_{ij}	Open walk	0.89	4.08×10^{-2}	3.63×10^{-2}
p_{ij}^2	Closed walk	0.37	0.29	0.11
$p_{ij}p_{ik}$	Open walk	7.86×10^{-2}	-7.37×10^{-1}	-5.80×10^{-2}
$p_{ij}p_{kl}$	Disconnected graph	0.16	-4.21×10^{-2}	-6.88×10^{-3}
p_{ij}^3	Open walk	0.17	-1.76×10^{1}	-3.01
$p_{ij}^2p_{ik}$	Open walk	4.05×10^{-2}	5.75	0.23
$p_{ij}^2p_{kl}$	Disconnected graph	0.14	-1.99×10^{-1}	-2.84×10^{-2}
$p_{ij}p_{jk}p_{ik}$	Closed walk	3.05×10^{-3}	10.52	3.21×10^{-2}
$p_{ij}p_{ik}p_{il}$	Star	3.16×10^{-3}	-1.94×10^{1}	-6.14×10^{-2}
$p_{ij}p_{ik}p_{kl}$	Open walk	1.74×10^{-2}	3.93	6.85×10^{-2}
PIPs with the highest contributions of all 120 PIPs				
p_{ij}^4	Closed walk	8.51×10^{-2}	68.94	5.87
p_{ij}^6	Closed walk	2.34×10^{-2}	94.91	2.22
$p_{ij}^3p_{ik}p_{kl}$	Open walk	5.88×10^{-3}	58.29	0.34
$p_{ij}^4p_{ik}p_{il}$	Open walk	3.65×10^{-4}	780.86	0.28
$p_{ij}^4p_{ik}$	Open walk	7.58×10^{-3}	36.57	0.28

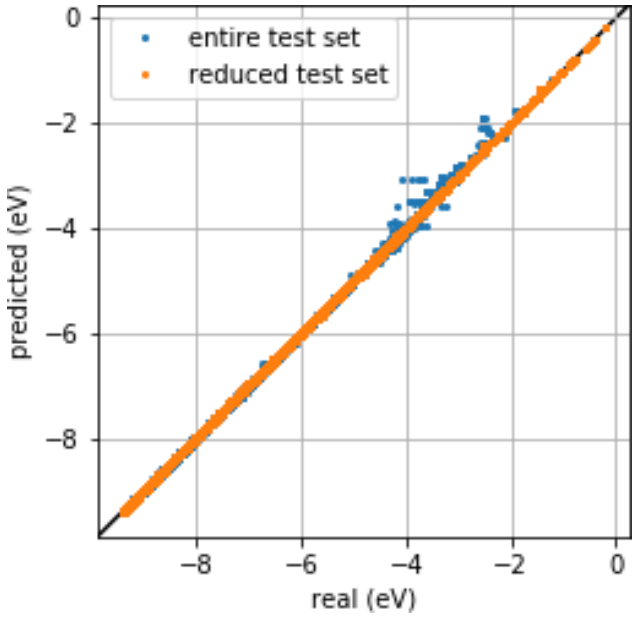


Figure 3.30: Crossplot of the results of the PIP method with Morse proximities with $a = 2\,a_0$ on the test data set.

3.5 Results H_2–H_2, linear regression on PMIs

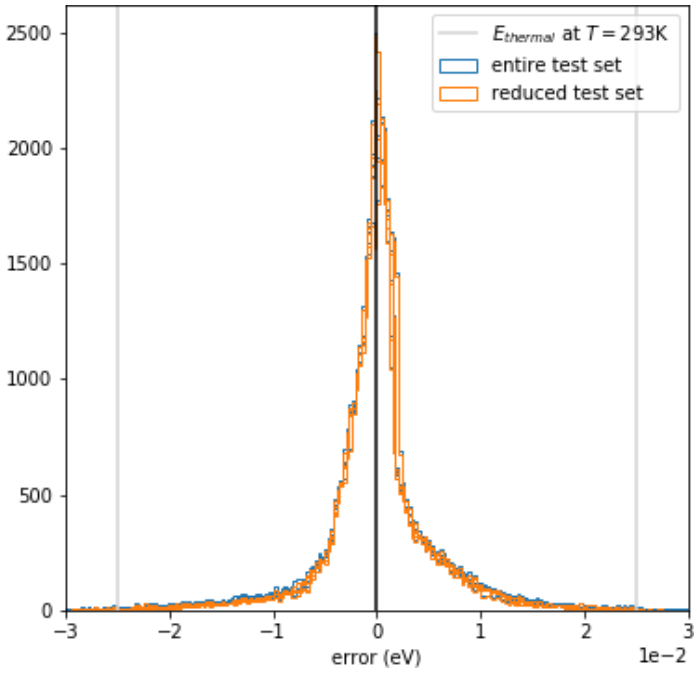
A lot of linear regressions on the PMIs $\mathrm{tr}(\mathbf{P}^k)$ and $\mathrm{tc}(\mathbf{P}^k)$ (described in 3.1.10) were tried, with various types of proximities and parameters (n for proximities A, B and C, a for Morse proximities). The full results are in Appendix E. The results in this appendix are from fits on cross-validation with suspected outliers removed. We also tried to use the first 4 PMIs as a neural network input, but this did not perform any better than a linear regression.

There are a lot of different parameter combinations. The fits were to:

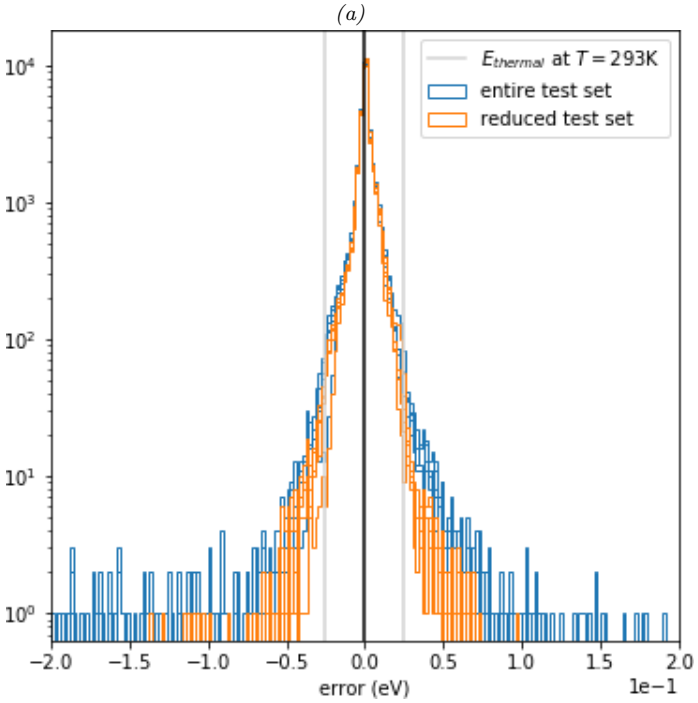
- only the $\mathrm{tr}(\mathbf{P}^k)$ PMIs,
- to the $\mathrm{tr}(\mathbf{P}^k)$ (trace) as well as the $\mathrm{tc}(\mathbf{P}^k)$ (trace complement) PMIs,
- or to combinations of $\mathrm{tr}(\mathbf{P}^k)$ with $k = 2, 3, 4$ (for example $\mathrm{tr}(\mathbf{P}^2)\,\mathrm{tr}(\mathbf{P}^3)$).

Including the trace complements $\mathrm{tc}(\mathbf{P}^k)$ as well helped, but making combinations of the PMIs did not. Table 3.7 shows the best result achieved for each type of proximity. The C proximities performed the best, the Morse proximities also did well. The B proximities did very badly, which was most likely because of numerical issues (the condition number of the matrix with the PMIs was often in the order of 10^{31}), since the value of the B proximities goes to infinity near zero. It was often the case that adding more inputs actually made the fit worse (both the training and validation result), which should be impossible for a linear least squares fit if there are no numerical issues. The Gaussian proximities were also surprisingly bad. It is possible that this is because their gradient gets smaller closer to zero, when in the PES there will be a steep potential wall as two atoms get close. The sine/cosine proximities B were better than expected, given the fact that they also to to infinity near $r = 0$, but did not manage to improve on the result with the proximities C.

This shows that the best settings turned out to be proximities C ((3.52)), with the maximum k equal to 13, $n = [2, 3, 5, 8, 12]$. This gives 125 parameters in total, which is comparable to the 120 parameters in a



(a)



(b) Zoomed out, with logarithmic y-axis to show outliers.

Figure 3.31: Histograms (one per fold) of the results of the PIP method with Morse proximities with $a = 2\,a_0$ on the test data set.

Table 3.7: The best results of the PMI linear regressions on the validation data for each proximity type. For comparison, the RMSE achieved by simply always predicting the mean is also given. [a]using linear combinations of $\mathrm{tr}(\mathbf{P}^k)$ *up to order 3, [b]using* $\mathrm{tr}(\mathbf{P}^k)$ *and* $\mathrm{tc}(\mathbf{P}^k)$*, [c]using only* $\mathrm{tr}(\mathbf{P}^k)$

Type of proximities	Number of coefficients	Max. k	Parameter values n or a (a in bohr)	RMSE training (meV)	RMSE validation (meV)
Proximities B (cutoff=15.9Å) [c]	8	3	[1.5, 2., 2.5, 3.]	292.9 ± 1.1	294.4 ± 4.8
Proximities C (cutoff=10.1Å) [b]	125	13	[2., 3., 4., 6., 9., 12.]	2.8 ± 0.1	3.6 ± 0.4
Morse [b]	155	16	[0.7, 2., 3., 4., 5.]	2.2 ± 0.1	3.6 ± 1.7
Sine/cosine proximities B (cutoff=15.9Å) [c]	120	7	[0., 1., 2., 3., 4., 5., 6., 7., 8., 9.]	10.4 ± 0.2	11.9 ± 1.2
Always predict mean	1	-	-	1817. ± 16.	1815. ± 64.

Table 3.8: Results of the PMI method on the validation data and on the test data. Uses both $\mathrm{tr}(\mathbf{P}^k)$ *and* $\mathrm{tc}(\mathbf{P}^k)$*. Proximities C, a maximum* k *of* 13*,* $n = [2, 3, 5, 8, 12]$*, 125 parameters*

	RMSE on entire test set (meV)	RMSE on reduced test set (meV)
Trained on entire cross-validation set	10.83 ± 0.45	5.09 ± 0.41
Trained on reduced cross-validation set	87.39 ± 19.51	2.71 ± 0.21
		Validation RMSE (meV)
Trained on entire cross-validation set		17.36 ± 8.77
Trained on reduced cross-validation set		3.59 ± 0.26

7^{th} order PIP fit. The 5 fits (from 5-fold cross-validation) with those settings were used on both the entire test set as well as the reduced test set. The results of this are in Table 3.8. Figure 3.32 and 3.33 show a crossplot and histograms, respectively, of the error of the test set, for the fits that were trained on the full cross-validation data.

It shows that in this case, training on a more inclusive data set improves performance on the difficult points, but when testing on a reduced data set it is better to also train on a reduced dataset. At least, for these settings, but these settings were chosen based on their performance on the reduced cross-validation data set, so that may not always be the case.

It shows that on the reduced test set, the performance was actually better than during cross-validation. A possible explanation is the uniform spherical sampling in the test set (see also A.4), while the cross-validation data set oversampled the configuration that are similar to the L-configuration in 2.6. These configurations seem to be more difficult than the others.

Table 3.9 shows almost the same thing as 3.8, but in this case the trace complement for the last proximity matrix (with $k = 13$) is left out, so the number of traces is equal to the number of trace complements. This gives exactly 120 parameters (the same as the PIP method). It shows that the performance does not change much.

The full results in Appendix E show that the PMI method, especially when using both $\mathrm{tr}(\mathbf{P}^k)$ and $\mathrm{tc}(\mathbf{P}^k)$, is very temperamental, in that small changes in the settings can give a very different result, for example when using the best settings mentioned earlier (both $\mathrm{tr}(\mathbf{P}^k)$ and $\mathrm{tc}(\mathbf{P}^k)$ with proximities C, k up

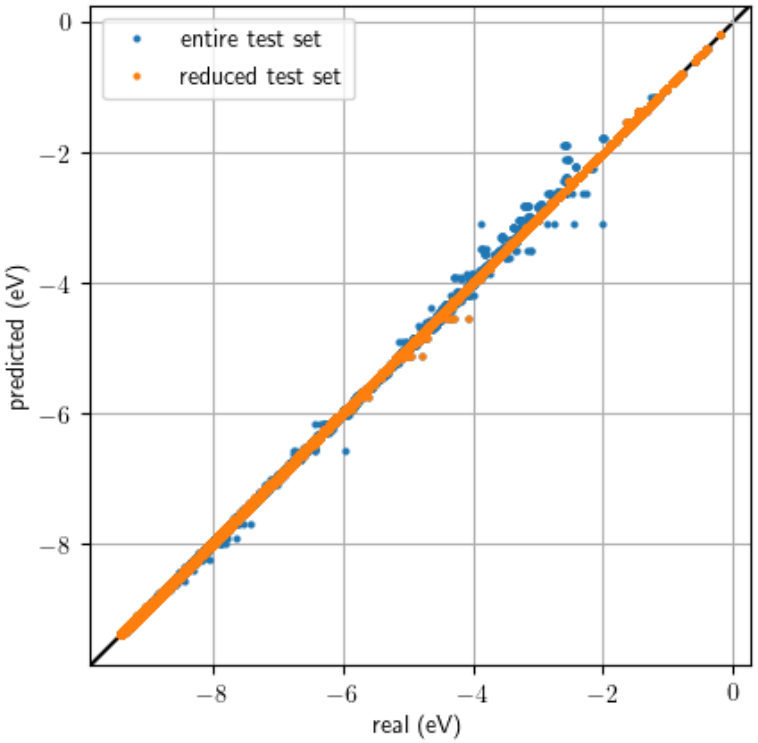
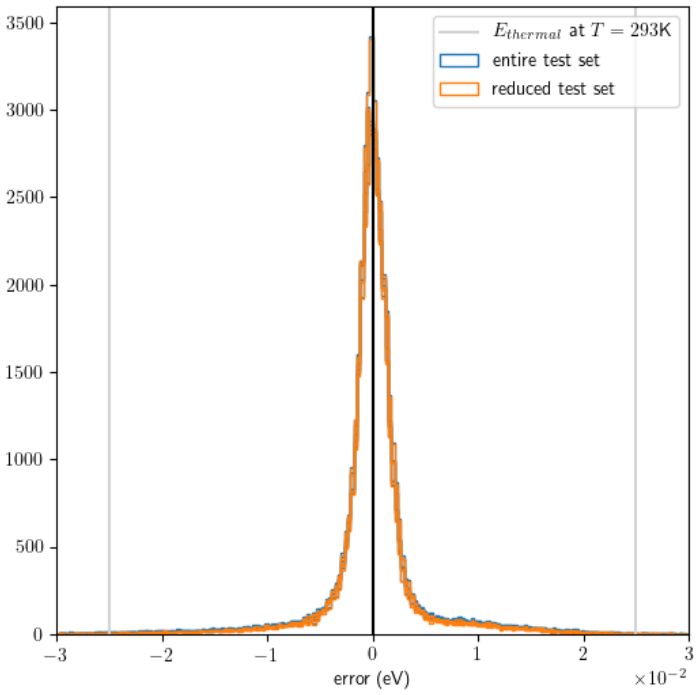


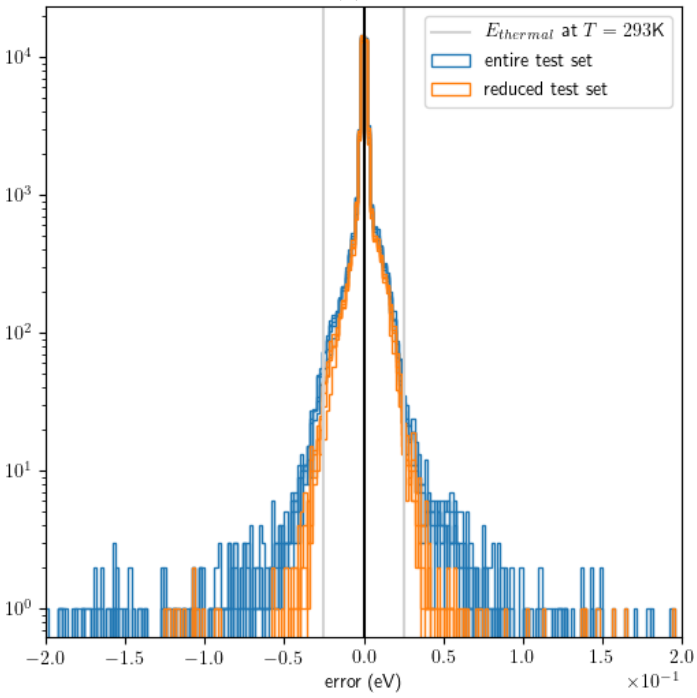
Figure 3.32: Crossplot of the results of the PMI method with the best settings on the test data set.

Table 3.9: Results of the PMI method on validation data of 5-fold cross-validation of fits trained on part of the training set (outliers removed). Uses both $\mathrm{tr}(\mathbf{P}^k)$ *and* $\mathrm{tc}(\mathbf{P}^k)$*, but no trace complement for the maximum* $k = 13$*. Proximities C, a maximum* k *of* 13*,* $n = [2, 3, 5, 8, 12]$*,* 120 *parameters*

	RMSE on entire test set (meV)	RMSE on reduced test set (meV)
Trained on entire cross-validation set	11.15 ± 0.95	5.17 ± 0.38
Trained on reduced cross-validation set	40.08 ± 7.11	3.16 ± 0.19
	validation RMSE	
Trained on entire cross-validation set	16.48 ± 6.05	
Trained on reduced cross-validation set	4.64 ± 0.91	



(a)



(b) Zoomed out, with logarithmic y-axis to show outliers.

Figure 3.33: Histograms (one per fold) of the results of the PMI method with the best settings on the test data set.

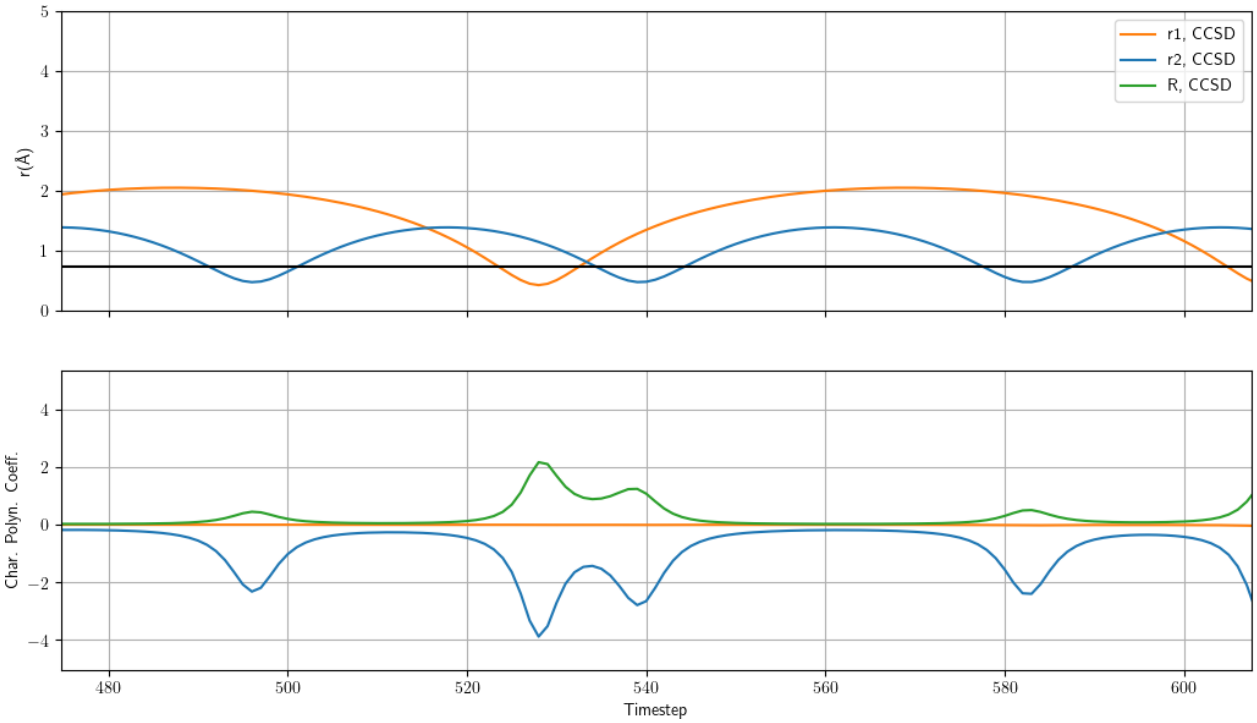


Figure 3.34: The bond lengths r_1 and r_2 and the CPCs (rescaled so they each have a standard deviation of 1, equal to the first 3 PMIs rescaled) over time during the same trajectory as the one discussed in subsections 3.3.2 and 3.3.2.

to order 13 and $n = [2, 3, 5, 8, 12]$), we achieve an RMSE of 3.6 ± 0.4 meV, but using those exact same settings except taking k up to order 12 instead of 13, results in an RMSE of $(1.8 \pm 3.6)\times10^5$ meV. Possible explanations are numerical issues, even though the proximities C are bounded between 0 and 1, or that the method is very sensitive to outliers, even though these results were obtained with the reduced cross-validation set, which already had the worst outliers removed.

Comparison with Table 3.5 for the PIP method shows that the PIP method seems somewhat more robust when it comes to outliers; the PIP method performs better on the entire test set, although for the case where both methods were also trained on the entire cross-validation data set the difference is small. The PMI method seems to work slightly better on the reduced test set, although we experimented more with different settings for that method, so that comparison may not be entirely fair.

Figure 3.34 shows that absolutely nothing special happens near $r_1 = r_2$, which is where the PME method shows the just-barely-avoided eigenvalue crossings.

Chapter 4

Molecular Dynamics

.

4.1 Theory

Molecular dynamics is a computational method used to obtain the trajectories of certain particles over a length of time. The particles are given initial positions and velocities, and allowed to interact from there under the influence of a molecular potential, with $\boldsymbol{a} = f(\boldsymbol{x})$. From these trajectories, macroscopic properties of the system can be determined.

The most common integration method for such trajectories is the velocity Verlet method, which looks as follows [115]

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \mathrm{d}t\, \boldsymbol{v}_n + \mathrm{d}t^2\, f(\boldsymbol{x}_n)/2 \tag{4.1}$$

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \mathrm{d}t\, [f(\boldsymbol{x}_{n+1}) + f(\boldsymbol{x}_n)]\, /2. \tag{4.2}$$

The timescale of the simulation is the most serious bottleneck in molecular dynamics, because the duration of the simulation needs to be long enough to capture the processes of interest, but the time step size is limited by atomic oscillations.

Quasiclassical trajectory (QCT) calculations are a combination of molecular dynamics (which uses classical physics) with a quantum PES. It means the hydrogen atoms move classically on a PES based on quantum chemistry calculations. In other words, the atoms are treated classically, the electrons quantum mechanically. Since the atoms are usually a lot heavier and slower than the electrons, calculating their motion classically is usually a reasonable approximation. However, because hydrogen is so light, this approximation is somewhat limited in accuracy (as mentioned in section 2.1.7). Especially near reaction or dissociation thresholds, quantum effects are important. However, in this case we are mostly interested in the vibrational and rotational energy transfer, where QCT seems to be reasonably accurate [116, 40].

For this part of the project, bunches of molecular dynamics trajectories of H_2–H_2 were calculated. The collision energy is defined as the total kinetic energy of the molecules at the start of a trajectory in a reference frame with a stationary global center of mass, where for each molecule its velocity is taken to be the velocity of its center of mass relative to this global center of mass. Apart from the collision energy, there is also kinetic energy in the motion of the atoms within a molecule, which are vibrational and rotational energy.

4.1.1 Vibrational states

The vibrational energy (relative to the potential minimum) of a vibrational state with quantum number ν of a diatomic molecule is [117, p. 87]

$$E_{\text{vib}} = \omega_e \left(\nu + \frac{1}{2}\right) - \omega_e x_e \left(\nu + \frac{1}{2}\right)^2 + \omega_e y_e \left(\nu + \frac{1}{2}\right)^3 + \omega_e z_e \left(\nu + \frac{1}{2}\right)^4 + \ldots, \tag{4.3}$$

The values of ω_e and $\omega_e x_e$ for H_2 are 546 meV and 15.0 meV, respectively [118]. It is important to note here that these values ω_e do not say anything about the actual classical frequency with which the molecule will vibrate.

CO_2 however, is not a diatomic molecule and therefore there are three vibrational modes that play a role; the symmetric stretching mode (r_a and r_b oscillate with the same phase), the antisymmetric stretching mode (r_a and r_b oscillate with opposite phase) and the bending mode (the internal angle ϕ oscillates). The vibrational energies for CO_2 can be approximated with the following formula [119, 56]:

$$\begin{aligned} E_{\text{vib}} =& \omega_1 \left(\nu_1 + \frac{1}{2}\right) + \omega_2 \left(\nu_2 + 1\right) + \omega_3 \left(\nu_3 + \frac{1}{2}\right) \\ &+ x_{11} \left(\nu_1 + \frac{1}{2}\right)\left(\nu_1 + \frac{1}{2}\right) + x_{12} \left(\nu_1 + \frac{1}{2}\right)\left(\nu_2 + 1\right) + x_{13} \left(\nu_1 + \frac{1}{2}\right)\left(\nu_3 + \frac{1}{2}\right) \\ &+ x_{22} \left(\nu_2 + 1\right)\left(\nu_2 + 1\right) + x_{23} \left(\nu_2 + 1\right)\left(\nu_3 + \frac{1}{2}\right) + x_{33} \left(\nu_3 + \frac{1}{2}\right)\left(\nu_3 + \frac{1}{2}\right) \\ &+ x_{l_2 l_2} l_2^2, \end{aligned} \tag{4.4}$$

The values of ω_i, x_{ij} and $x_{l_2 l_2}$ can be found in Kozák and Bogaerts 2014 [56].

4.1.2 Rotational states

The classical rotational energy of a molecule is the energy in the rotation of the molecule around its center of mass. This is is equal to

$$E_{\text{rot}} = \sum_i \frac{1}{2} I_i \boldsymbol{\omega}_i^2 = \sum_i \frac{1}{2} m_i \boldsymbol{v}_{i,\perp}^2, \tag{4.5}$$

with I_i the moment of inertia of atom i, $\boldsymbol{\omega}_i$ its angular velocity pseudovector and m_i its mass. The velocity of an atom relative to the center of mass is $\boldsymbol{v}_i$, and $\boldsymbol{v}_{i,\perp}$ the component of this velocity perpendicular to the vector $\boldsymbol{r}_i$ that connects the atom to the location of the center of mass $\boldsymbol{r}_{\text{CoM}}$. This is how the rotational energy is extracted from the quasiclassical trajectories.

A diatomic molecule will rotate around an axis that is perpendicular to the molecular axis and goes through the molecule's center of mass. CO_2 is a linear molecule, and will therefore have only one important axis of rotation, which is perpendicular to the molecular axis. However, once the angle ϕ is large, a rotation about the molecular axis can also become significant.

To relate this energy to a rotational quantum state, one can look at the energy of a rotational state with quantum number J. For small J for a diatomic molecule such as hydrogen, this is [41, p. 68]:

$$E_{\text{rot}} = \frac{\hbar^2 J(J+1)}{2I}, \tag{4.6}$$

with I the total moment of inertia of the molecule. Assuming I is constant, we get the rigid rotator approximation and we can introduce the constant B_e:

$$E_{\text{rot}} = B_e J(J+1) \tag{4.7}$$

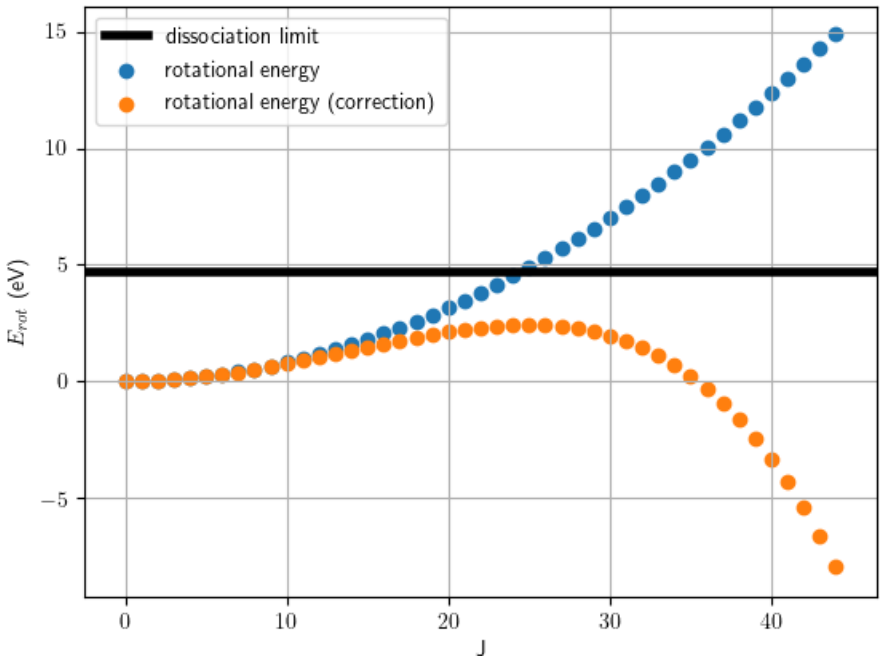


Figure 4.1: The energy of the rotational levels J of hydrogen for the uncorrected ((4.6)) and the corrected energy ((4.8)).

For H_2, the value B_e is 7.55 meV [118]. When adding a correction for higher J, this becomes:

$$E_{\text{rot}} = B_e J(J+1) - D_e J^2 (J+1)^2. \tag{4.8}$$

For H_2, the value of D_e is 5.84×10^{-3} meV [118]. For comparison: for $J = 1$, the correction is only 0.0234 meV, for $J = 10$ it is 70.7 meV. Figure 4.1 shows these energies for increasing J. At $J = 25$, the corrected version starts declining and the polynomial fit is not valid anymore, which is why only $J = 0$ to $J = 25$ are taken into account.

The rigid rotator approximation assumes that the rotational inertia I is constant, meaning they assume the molecule has its equilibrium bond length. This is not the case, since the molecule vibrates. However, since the vibrational state is initialized first, we can calculate I and use (4.6) to determine the rotational energy.

When one wants to calculate cross sections that are thermally averaged, the rotational state J needs to be sampled from a Boltzmann distribution for a certain temperature. Figure 4.2 shows what the Boltzmann distribution looks like for various temperatures for H_2. Appendix A.4 describes how to initialize a diatomic molecule with a certain vibrational and rotational energy. Quantum mechanically speaking, some rotational transitions are spin-forbidden, because they would change the spin state. We ignore this, since we are treating the atoms classically.

4.1.3 Probabilities and cross sections

A cross section is a quantity that expresses the probability of a certain process happening when two particles collide. It can be thought of as the area of the 'target' a particle needs to hit to cause the process of interest to occur. It can be approximated by simulating lots of trajectories of the particles approaching each other

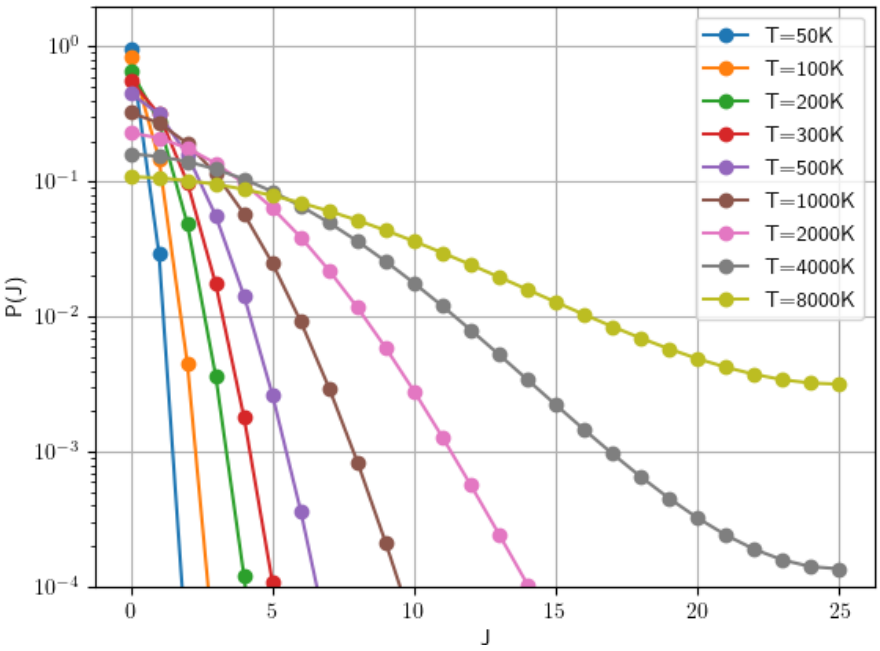


Figure 4.2: The relative population of the rotational states J of H_2 for various temperatures, using the rotational energy in (4.8).

with various impact parameters. In this chapter, several cross sections for dissociation of hydrogen and rovibrational energy transfer in hydrogen are calculated.

Cross sections can be approximated by analyzing the statistics of many randomly sampled trajectories. This means running simulations of trajectories of colliding molecules, while varying the impact parameter b. In the paper by Lombardi et al. [6], 30,000 trajectories were simulated for each unique collision energy and T_{rot} and set of initial vibrational states. The probability of vibrational state transition from v to v' can then be estimated as $P_{vv'} = \frac{N_{vv'}}{N_t}$, with $N_{vv'}$ the number of trajectories where this transition happens, and N_t the total number of trajectories. The cross section is then

$$\sigma_{vv'} = \int_0^{b_{max}} 2\pi b P_{vv'}(b) \mathrm{d}b. \tag{4.9}$$

The cross sections can be used to determine rate constants for the process they describe. These cross sections and rate constants can then be used in simulations involving a lot more molecules. They can be used to model how energy in the system will move to and from translational energy, rotational energy and vibrational energy.

It can be useful to determine cross sections that are thermally averaged in some way, for example the rotational states. This means assuming that these states have already relaxed to equilibrium and there is a rotational temperature T_{rot}. The resulting cross section can be used in a simulation where keeping track of the rotational state is not necessary. The process for setting up a simulation is then simpler; instead of using the Boltzmann distribution of J corresponding to the rotational temperature and then using one cross section for each J, only one thermally averaged cross section needs to be used. It is also possible to thermally average over the collision energy E_{coll}, where it is assumed that the translational motion has relaxed to equilibrium and there is a collisional temperature that is given.

4.2 Methodology

4.2.1 Trajectories using LAMMPS

To compute the trajectories, the program LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) is used [120]. LAMMPS was extended with custom code, which implements a neural network and the analytical gradient (see F for the derivation of the analytical gradient of the potential) of its energy predictions[1]. It was verified that this analytical gradient was the same as the gradient of the network computed by PyTorch's automatic differentiation. The best performing NN for the PME method was used (with the sole exception of the rotational cross section in 4.4, which was done with an older network with a similar performance), which is discussed in Section 3.3. The following is an example of a LAMMPS input file:

```
units           electron
atom_style      atomic
atom_modify     map             yes
boundary        f f f
region          myreg block     -30.0 30.0 &
                                -30.0 30.0 &
                                -30.0 30.0
create_box      1 myreg
create_atoms    1 single  -0.6542561833886333   -0.5150144618469884     -27.41992441975301
create_atoms    1 single  0.6542561833886333    0.5150144618469884      -27.58007558024699
create_atoms    1 single  0.3473961564372681    -0.584578291017627      26.823582416153627
create_atoms    1 single  -0.3473961564372681   0.584578291017627       28.176417583846373


group group1 id 1
group group2 id 2
group group3 id 3
group group4 id 4
mass            *         1.00782503207

velocity group1 set -0.000466166235213183 0.0002762497586835011 -0.0020320742050145262
velocity group2 set 0.000466166235213183 -0.0002762497586835011 0.0020320742050145262
velocity group3 set -4.774679328081884e-06 0.0007171149594321428 -0.0006222022501271932
velocity group4 set 4.774679328081884e-06 -0.0007171149594321428 0.0006222022501271932


pair_style      pml
pair_coeff  * * none

thermo          10
fix             1 all nve

timestep        0.24188843265963555
dump        dump1 all custom 1 /data/fleur/la/dump_dir_6_01/results_xyz/dumpH2_xyz.*.txt x y z
dump        dump2 all custom 1 /data/fleur/la/dump_dir_6_01/results_force/dumpH2_force.*.txt fx fy fz
dump        dump3 all custom 1 /data/fleur/la/dump_dir_6_01/results_v/dumpH2_v.*.txt vx vy vz

run         22 #  1 #


set group group3 x 0.3473961564372681 y -0.584578291017627 z 26.823582416153627
set group group3 vx -4.774679328081884e-06 vy 0.0007171149594321428 vz -0.0006222022501271932
set group group4 x -0.3473961564372681 y 0.584578291017627 z 28.176417583846373
set group group4 vx 4.774679328081884e-06 vy -0.0007171149594321428 vz 0.0006222022501271932
velocity        group1 set -0.010993521189413212 -0.0010408813751451734 0.07621833990444324 sum yes
```

[1]Thank you to dr. ir. Jesper Janssen for doing the majority of the C++ programming for this and a thank you to dr. ir. Jan van Dijk for help with this as well.

```
velocity        group2 set -0.010993521189413212 -0.0010408813751451734 0.07621833990444324 sum yes
velocity        group3 set 0.010993521189413212 0.0010408813751451734 -0.07621833990444324 sum yes
velocity        group4 set 0.010993521189413212 0.0010408813751451734 -0.07621833990444324 sum yes

label loop
variable a loop 500
run     100
print '$(x[1]) $(y[1]) $(z[1]) $(x[2]) $(y[2]) $(z[2]) $(x[3]) $(y[3]) $(z[3])'
next a
print   "next a"
jump SELF loop
```

The line `print '$(x[1]) $(y[1]) $(z[1]) $(x[2]) $(y[2]) $(z[2]) $(x[3]) $(y[3]) $(z[3])'` is important, because when one or more of the atoms end up outside the simulation box, the coordinates of those atoms become `nan`, but LAMMPS will happily continue the trajectory. However, when LAMMPS is then asked to write the coordinates to the screen, it crashes. This is the easiest way to stop a trajectory once one or more atoms are out of range. This means this print statement functions as a check to determine if the simulation is done, which is why it is executed every 100 steps.

To call LAMMPS from Python, the Python script `lammps_NN_trajectories_2H2.py` is used. This script creates LAMMPS input files and uses the Python multiprocessing library to run multiple trajectories in parallel. This script itself also takes an input file, which specifies what kind of trajectories and how many should be run, where to find LAMMPS and also where the results should be written. An example of such an input file is:

```
lammps_command mpirun -np 1 ../src/lmp_g++_openmpi -in  # command to run lammps
lammps_input_dir /home/fleur/LAMMPS_ML/lammps/input     # dir with ML files
lammps_temp_dir /data/fleur/la          # dir to put temporary lammps files
results_dir /data/fleur/results/lammps_results_NN_2H2/  # dir to put results

J1 9  # vibrational quantum number of molecule 1
J2 9  # vibrational quantum number of molecule 2

T_coll 8000  # translational temperature in kelvin

vi1 9      # initial vibrational quantum number of molecule 1
vi2 9      # initial vibrational quantum number of molecule 2

R_init 55    # initial separation between molecules in bohr
R_limit 60   # edge length in bohr of simulation box (is a cube)
b_max 38     # maximum impact parameter in bohr (38 bohr = 20 angstrom)

n_trajs_total 30000  # nr of trajectories to do
n_trajs 10   # nr of trajectories to run simultaneously. Should be less than nr of cores
max_steps 50000 # maximum nr of time steps
dt 10           # time step in hbar/hartree
```

For each set of cross sections, 30,000 trajectories were run. Depending on the number of time steps needed, this took about 4 to 10 hours per cross section. The trajectories for the rotational cross sections were on average about 1800 steps (these trajectories had an initial $E_{\text{coll}} = 1.2\,\text{eV}$, for lower values the atoms move more slowly and therefore need more time steps), and took on average about 15 seconds per trajectory on a computer with an AMD Ryzen Threadripper 2990WX 32-Core Processor @ 3.0GHz and 96GB of memory. We ran 30 trajectories in parallel, so one set of 30,000 trajectories took 4 hours. LAMMPS is not completely threadsafe, so we cannot guarantee that running multiple trajectories at the same time will always work with no problems, but we experienced no problems. (LAMMPS can also parallellize a simulation by dividing the atoms over multiple cores using Message Passing Interface (MPI), but this is not very useful in this case because each simulation will only have 4 atoms. For that reason we chose to run multiple trajectories in

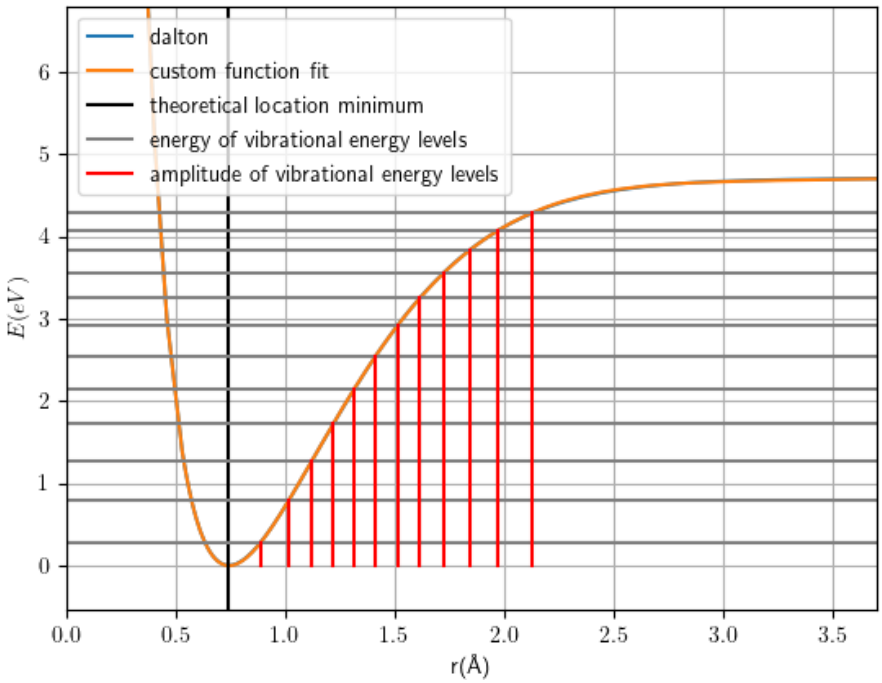


Figure 4.3: Amplitudes of vibrational energy levels of H_2.

parallel instead of parallelizing one trajectory.)

4.2.2 Vibrational state

For the H_2–H_2 system, the energy levels were obtained from (4.3), using the first two terms shown in that equation. This formula is an expansion around the minimum, so it gets less accurate for a higher vibrational quantum number ν and higher order corrections are required.

These energy levels were related to the maximum amplitude of the oscillation. This is the value of r where the potential energy is equal to the vibrational energy, with $r > r_{minimum}$. In Figure 4.3 this r is the value where the dark grey line crosses the potential curve. This is the maximum amplitude of the classical oscillation corresponding to this quantum vibrational level. These amplitudes were determined in the script `H2.py`. For each level, a trajectory calculation (CCSD with basis set aug-cc-pVTZ, same as the data set) was done where the molecule vibrated in place, to determine its period (shown in Figure 4.4). The periods and amplitudes are given in Table 4.1.

The vibrational state in the beginning of the trajectories was initialized by creating a molecule with a bond length equal to this amplitude. This means that the molecule is created with all its vibrational energy in the form of potential energy. This means all molecules start with the same phase. This is especially a problem if both molecules start in the same vibrational state, because then their phase will always be the same when they collide, which ruins the statistics. To avoid this, the configuration of one of the two molecules is reset back to $t = 0$ after a certain delay. This delay is chosen randomly uniform from the period associated with this vibrational level. For CO_2–CO_2 this might be unnecessary, because the random state is very chaotic anyways with three atoms, and the vibrational time scale compared to the time scale of the collision is much smaller for CO_2–CO_2 than for H_2–H_2. In the input script, the first time the command `run` is called, it is used to have the molecules vibrate in place and after that one of the two molecules is reset to

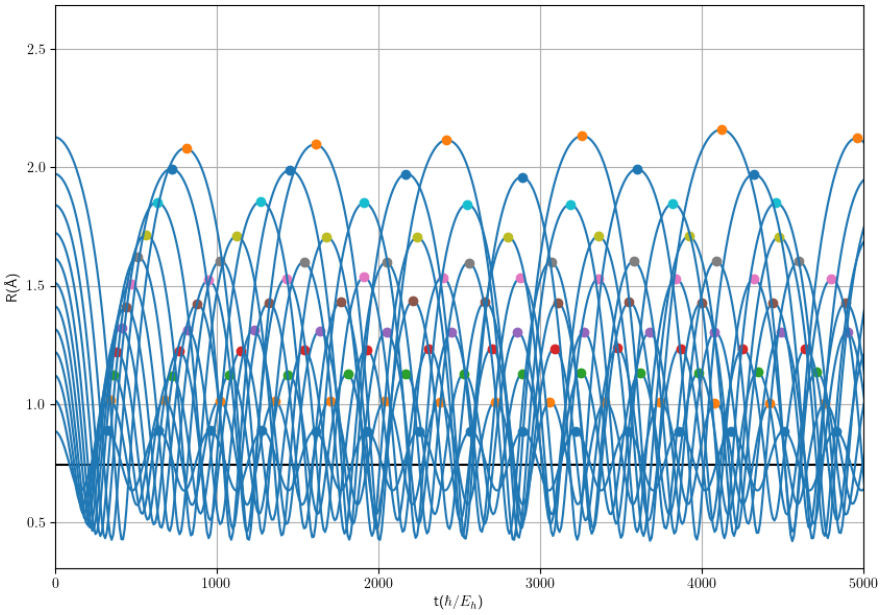


Figure 4.4: Trajectories of one H_2 molecule vibrating in place used to determine the period, for the first 12 vibrational energy levels.

Table 4.1: The quantum vibrational energy levels of a H_2 molecule and their associated classical amplitudes and periods.

ν	Energy (eV)	Max. amplitude (Å)	Period (fs)
0	0.27	0.89	7.79
1	0.78	1.02	8.25
2	1.27	1.12	8.76
3	1.73	1.22	9.32
4	2.15	1.32	9.86
5	2.55	1.41	10.79
6	2.91	1.51	11.58
7	3.25	1.62	12.56
8	3.55	1.72	13.68
9	3.83	1.84	15.09
10	4.07	1.97	17.24
11	4.29	2.13	21.07

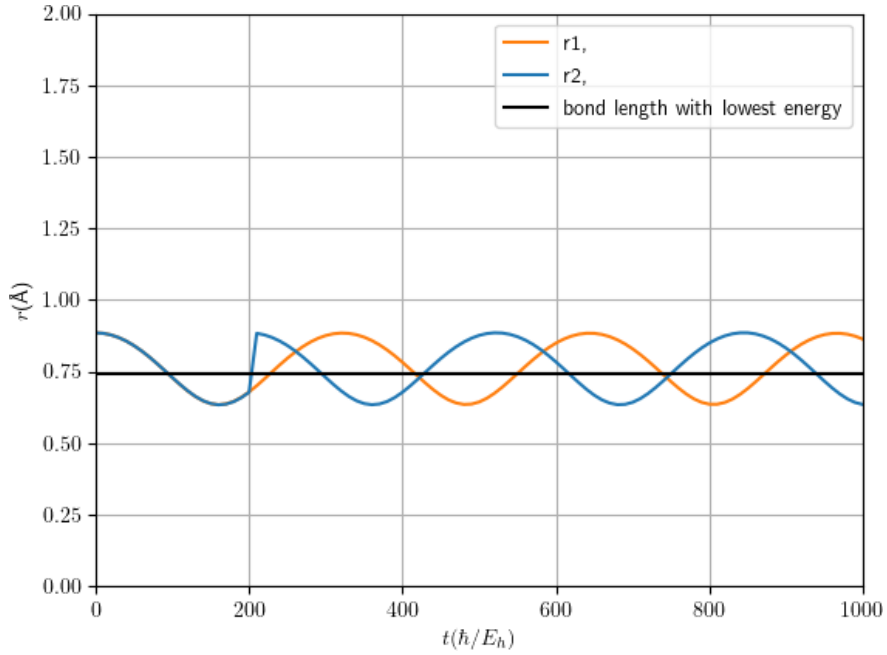


Figure 4.5: The bond lengths during a typical H_2–H_2 trajectory. The configuration of one molecule is reset, to randomize the initial phase. In this trajectory this happens near $t = 200\,\hbar/E_\mathrm{h}$.

its original position, and both molecules are given the extra velocity that makes them collide.

The vibrational state at the end of the trajectories H_2–H_2 was determined by taking the amplitude of the oscillation and picking the quantum number with an energy closest to the potential energy associated with this amplitude. There are more sophisticated methods to do this (Gaussian binning, etc.) which can be investigated in the future.

For a 3-atom (or more) molecule such as CO_2, extracting the vibrational state is a bit more difficult. One possibility is to use normal mode analysis, which is a decent approximation near the bottom of the potential well. For CO_2, there are three coordinates of relevance: $p_1 = r_1 - r_{eq}$, $p_2 = r_2 - r_{eq}$, ϕ. The three modes are: antisymmetric stretching ($p_2 = -p_1$), symmetric stretching ($p_1 = p_2$) and the bending mode (ϕ oscillates). This is only approximately correct near the bottom of the well because of the harmonic approximation. the 'amplitude' of the symmetric mode is then the amplitude of $(p_1 + p_2)/2$, for the antisymmetric this is $(p_1 - p_2)/2$, and the bending mode is just described by ϕ. However, this harmonic approximation is really only accurate near the bottom of the potential well.

4.2.3 Rotational state

To initialize a molecule in a certain rotational state J, using the rigid rotor approximation (which means assuming that the vibrational state is not interfering), we can use (4.7) or (4.8) to get the corresponding rotational energy E_rot. Appendix A.5 then details how to intialize a molecule with this rotational energy. However, the rigid rotor approximation turns out to be pretty bad even with low vibrational quanta. The rigid rotor approximation seems to only be reasonable if the molecules do not vibrate at all, but in our trajectories they do even for $\nu = 0$. Because the vibrational state changes the moment of inertia I, the rotational energy is changed. Since the vibrational state is initialized first, we can calculate I and use (4.6)

to determine the rotational energy corresponding to the desired value of J.

To extract the rotational state, we essentially do the opposite of what we do to intialize a molecule with a certain rotational energy in Section A.5 and extract the rotational energy of each molecule. This rotational energy can then be mapped back to a rotational quantum number by simply determining which rotational energy it is closest to, using (4.7). This turned out to be very imprecise, because the rotational energy shows big oscillations caused by the vibrational state. Another option is also extract the moment of inertia I. Given the values of E_{rot} and I we found, we can then solve (4.6) for J

$$E_{\text{rot}} = \frac{\hbar^2 J(J+1)}{2I} \implies J = -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{4E_{\text{rot}}I}{\hbar}}, \tag{4.10}$$

and round it to the nearest integer. However, this does not include the correction with D_e in (4.8), which means we need to choose to correct for higher J or correct for the influence of the vibrational state.

The code to initialize the rotational state used (4.8), only for extracting the final rotational state was (4.10) used. In the future, modifying the code to also use (4.10) is probably a good idea to improve consistency.

The combination of the vibrational and rotational states of both molecules is often written (ν_1, J_1, ν_2, J_2).

4.2.4 Dissociation

We also calculated some dissociation cross sections for H_2–H_2. These were only to test the procedure, as the CCSD calculations were not meant to be accurate for dissociation (no multireference method and no counterpoise correction, which will be relevant near the dissociation threshold), sampling for NN did not include dissociation, and the QCT approach is not very accurate near dissociation thresholds anyway. The neural network did seem to replicate the potential fairly accurately for dissociated states as well, since the potential is forced to zero. However, dissociation also means higher energy states are possible, including configurations with the atoms very close together and in these configurations the potential may not be accurate at all.

To decide if two atoms formed a molecule together or not at the end of a trajectory, we used a threshold of 3 Å. There are then six possible outcomes of a trajectory:

Type 0 Same molecules at the end.

Type 1 Atom exchange. The molecules exchange an atom.

Type 2 Non-reactive dissociation. One original molecule is left, the other is dissociated.

Type 3 Reactive dissociation. Both molecules are dissociated, one new molecule is formed.

Type 4 Full dissociation. At the end, 4 separate atoms are left.

Type 5 Miscellaneous. Any trajectory outcome that does not fit in any of the other types. This turned out to be trajectories where three H-atoms were close together at the end (distance less than 3 Å).

Figure 4.6 shows an example of each of these outcomes.

Type 5 trajectories are interesting, but the network is extrapolating at that point because the training set contained no configurations with 3 H-atoms together, so these results are completely unreliable. This can also affect the trajectories where one or two molecules dissociate, but when a molecule dissociates, the relevant proximities are forced to zero, which should work even though those configurations are also sparsely sampled. It should also be mentioned that due to a bug in the code (now corrected), for these dissociation cross sections the rotational state of a molecule with $J = 0$ was not initialized as exactly zero, but with a rotational energy of about 15% of one vibrational quantum.

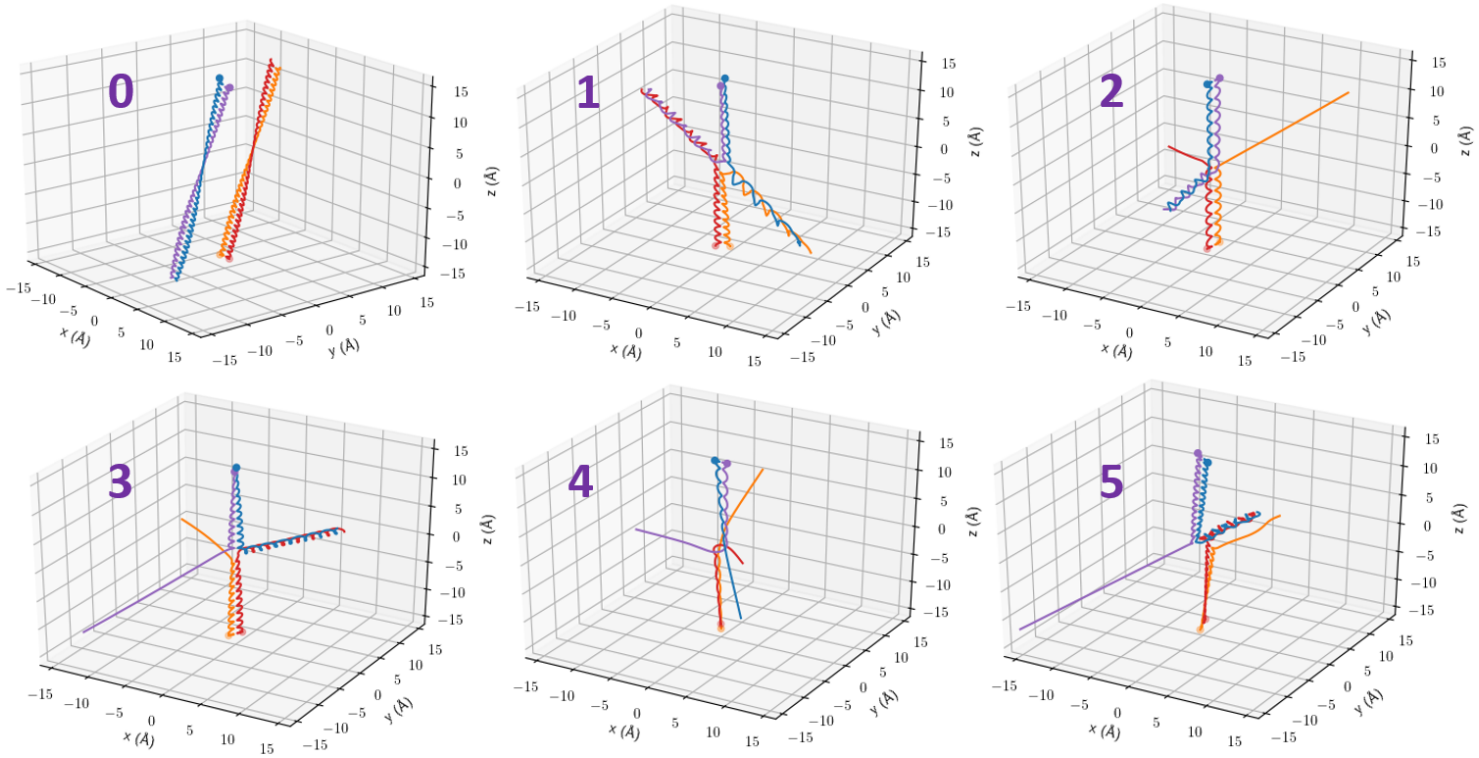


Figure 4.6: The 6 possible outcomes of a trajectory for a dissociation cross section. In each figure, each colored curve is the path of one atom and the starting point of each atom is marked with a dot. The blue/purple molecule starts at the top, moving down and the red/orange molecule starts at the bottom, moving up.

4.2.5 Cross sections

The previous three subsections described how the outcomes of the trajectories were categorized according to their vibrational state, rotational state or dissociation result. These results can then be processed into an actual cross sections, which was done in the script `2H2_crosssections.py`. To do this, (4.9) was calculated numerically, by binning the results based on their impact parameter b, with 50 bins from 0 to b_{max}. This is not a very sophisticated method, and more advanced methods such as Gaussian binning could be more accurate. Varying the number of bins did not change the results much. Another option is to calculate the cross sections directly, without binning, which requires that b is sampled according to the probability density function

$$f(b)\,\mathrm{d}b = \begin{cases} \frac{2\pi b}{\pi b_{max}^2}\,\mathrm{d}b & b \leq b_{max}, \\ 0 & b > b_{max}, \end{cases} \tag{4.11}$$

which essentially means uniform sampling on a disc-shaped target of area πb_{max}^2. The cross section can then be calculated as $\pi b_{max}^2 \frac{N_{vv'}}{N_t}$, which is equal to [size of the target]×[probability of process occurring when hitting this target]. This way of calculating the cross sections is not currently implemented.

Since during training with five-fold cross-validation, five networks were already trained, it made sense to use the five different networks to get an estimate of the accuracy of the cross sections. For the rotational and vibrational cross sections, only one neural network was used and Poisson statistics were used to obtain uncertainties for the results. The Poisson distribution tells us the spread in how often we can expect an event to occur in a certain time interval if it has a constant rate, or, in this case, the number of 'positives' (trajectories where a certain process happens, for example $N_{vv'}$) we can expect given a total number of tries (total number of trajectories N_t). The standard deviation of the Poisson distribution is $\sqrt{N_t}$, and this is used to calculate the uncertainty in the probability $P_{vv'}$ and from that the uncertainty in the cross section.

4.3 Results H_2–H_2 Dissociation cross sections

Figure 4.7 shows the outcomes of one set of 30,000 trajectories, for the initial state $(v_1, J_1, v_2, J_2) = (9, 0, 9, 0)$, at a temperature of 8000 K. Table 4.2 shows the calculated cross sections and compares them to the results of Ceballos et al. [121]. Their results were obtained from trajectories with the Aguado-Suarez-Paniagua PES [8], which has a RMSE of 52 meV. The Aguado-Suarez-Paniagua PES is fitted on ab initio data by Boothroyd et al., which was calculated using the multiple reference (single and) double excitation configuration interaction (MRD-CI) method, with an estimated RMSE of 24 meV.

It is important to note here that the uncertainty for our results in this table is simply the standard deviation over the 5 networks that were trained during cross-validation. So this uncertainty does *not* take into account the Poisson statistics or the errors in the ab initio methods.

Despite the questionable accuracy of our neural network for dissociation purposes, the calculated cross sections are still somewhat similar to the cross sections calculated by Ceballos et al., with the cross sections for atom exchange and reactive dissociation having overlapping uncertainty intervals. Only for non-reactive dissociation is the difference large; our cross section is more than twice as large. It is possible that Ceballos et al. calculated this cross section slightly differently and divided their result by 2, to account for the fact that there are two possible molecules that can split. However, there are also multiple ways an atom exchange reaction or a reactive dissociation can happen, so it would be a strange choice to do this.

The temperature of 8000 K corresponds to an energy of 0.689 eV, which is inside the range of 0.0433 eV–0.868 eV that was used for creating the training data sets. However, the collision energy E_{coll} for these trajectories was sampled from a Boltzmann distribution, so there were also some trajectories with a higher energy. This means the cross sections are not expected to be very accurate. There is also the fact that (4.8), used to initialize the rotational state, uses the rigid-rotor approximation, which will not be very accurate at higher vibrational energies like $v = 9$.

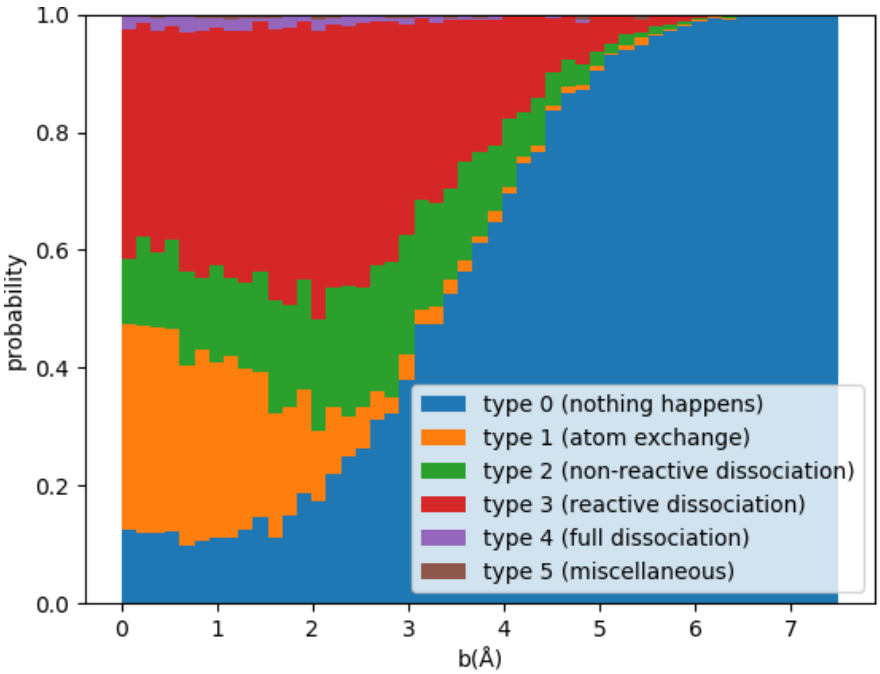


Figure 4.7: The outcomes of 30,000 trajectories with intial quantum numbers $(v_1, J_1, v_2, J_2) = (9, 0, 9, 0)$ *for a translational temperature of* 8000 K*, while varying the impact parameter* b*.*

Table 4.2: The cross sections computed using the neural network potential compared to the results of Ceballos et al. [121], for the initial state $(v_1, J_1, v_2, J_2) = (9, 0, 9, 0)$*, at* 8000 K*.*

		Cross sections (Å^2)	
	Type of trajectory outcome	Our results	Ceballos et al.
1.	Atom exchange	1.68 ± 0.07	1.82 ± 0.09
2.	Non-reactive dissociation	3.19 ± 0.42	1.53 ± 0.09
3.	Reactive dissociation	6.43 ± 0.28	6.64 ± 0.17

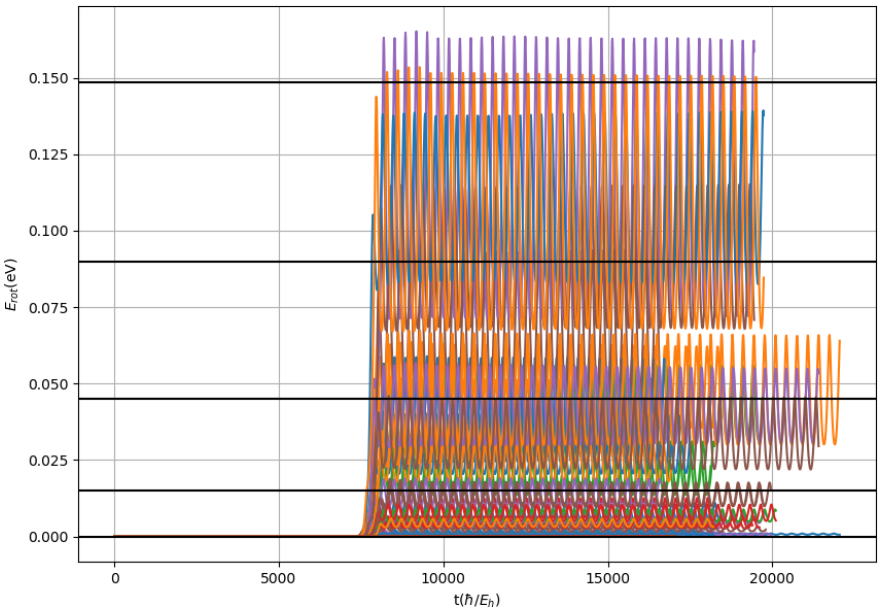


Figure 4.8: The rotational energy of a H_2 molecule over the course of a trajectory in which it collides with another H_2 molecule, for several trajectories. With initial state $(\nu_1, J_1, \nu_2, J_2) = (0, 0, 0, 0)$. $E_{coll} = 1.2\,\text{eV}$.

4.4 Results H_2–H_2 Rotational excitation cross sections

We only attempted one set of cross sections where only rotational excitation was relevant, which were from one bunch of 30,000 trajectories with a collisional energy E_{coll} of 1.2 eV and initial quantum numbers $(0, 0, 0, 0)$.

Extracting the rotational state by extracting the rotational energy and mapping this to the closest J, using (4.8) proved to be difficult because of the strong influence of the vibrational state. Figure 4.9 shows the outcome of the trajectories when the final state is determined this way. Figure 4.8 shows the rotational energy of a H_2 molecule over the course of some trajectories in which it collides with another H_2 molecule. It shows that the rotational energy oscillates strongly. This oscillation has approximately the same frequency as the vibrational state, which indicates the rotational and vibrational state mix, even for the lowest vibrational state $\nu = 0$. This means it is necessary to correct the results for the changing moment of inertia.

When the changing moment of inertia is taken into account, we can estimate J with (4.10). The results of this for some trajectories are in Figure 4.10 and the outcomes of all trajectories are shown in Figure 4.11. This way of determining J results in an estimate for J that is not disturbed by oscillations. The results are compared in Table 4.3. It shows that the two different methods give very different results, with the second method resulting in much smaller cross sections for $J_1 \geq 2$.

The table also shows that using either method to estimate J results in cross sections that are not in agreement with the results of Quéméner & Balakrishnan ([122]), which were based on full quantum calculations. A possible explanation is the fact that $E_{\text{coll}} = 1.2\,\text{eV}$ was higher than the maximum E_{coll} of 0.87 eV during sampling. The fact that the rigid-rotor approximation of (4.8) was still used during the initialization of the trajectory should not make a difference in this case, since the initial rotational states were $J = 0$. Another possible explanation is that we do not take into account which states are spin-forbidden, which Quéméner & Balakrishnan probably do. It is unlikely that we should take into account higher order

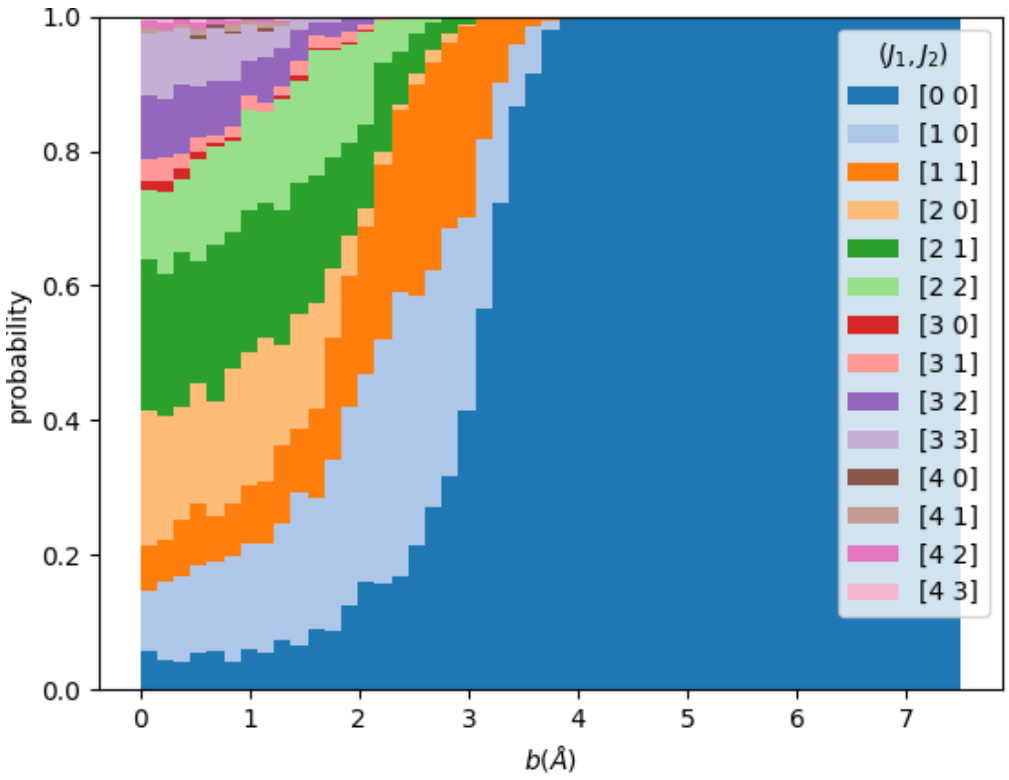


Figure 4.9: The outcomes of 30,000 trajectories with intial quantum numbers $(0, 0, 0, 0)$ *and final quantum numbers* $(0, J_1, 0, J_2)$*, with* $E_{coll} = 1.2\,\mathrm{eV}$*, while varying the impact parameter* b*.* J *was estimated using* (4.8)*.*

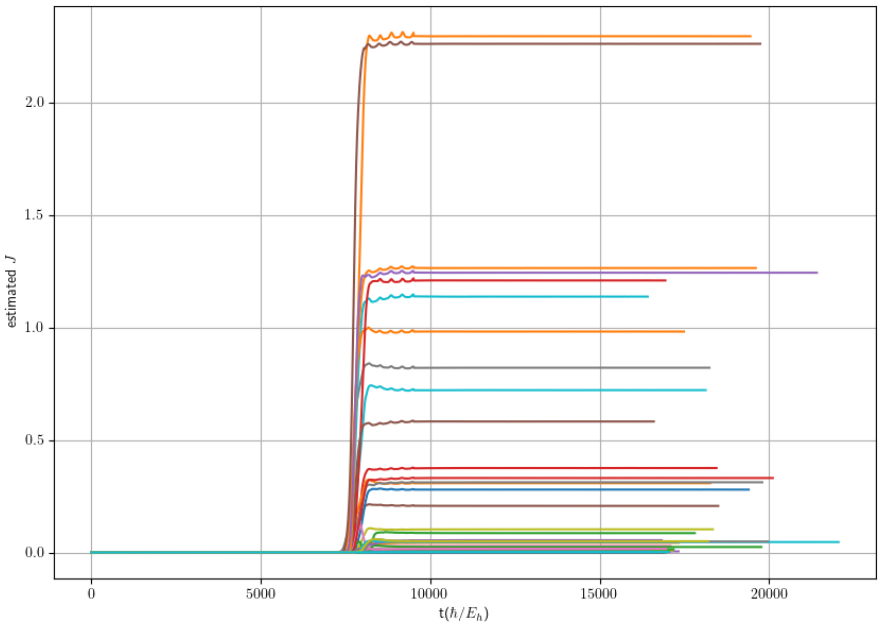


Figure 4.10: Estimated J over the course of several trajectories.

terms in (4.8), since in the vibrational and rotational quantum numbers are still very low.

4.5 Results H_2–H_2 Rovibrational energy transfer cross sections

We only attempted two sets of state-to-state cross sections where both the rotational and vibrational quanta played a role. We used two bunches of 30,000 trajectories each, both with a collisional energy E_{coll} of 0.871 eV and initial quantum numbers $(1, 0, 0, 0)$ and $(1, 0, 0, 1)$, respectively. For these trajectories, transitions to other vibrational quanta ($v = 0$ and $v = 2$) were also possible, although they were too rare to compute a reasonably accurate cross section.

To extract the rotational energy, we used (4.10), which takes into account the changing inertia. The outcomes of all trajectories are shown in Figure 4.12 and 4.13 for initial quantum numbers $(1, 0, 0, 0)$ and $(1, 0, 0, 1)$, respectively.

The computed cross sections are shown in Table 4.4 and 4.5. The cross sections from initial quantum number $(1, 0, 0, 1)$ are compared to results of Dos Santos et al. [123]. Agreement with these results is poor. Once again our results give significantly smaller cross sections.

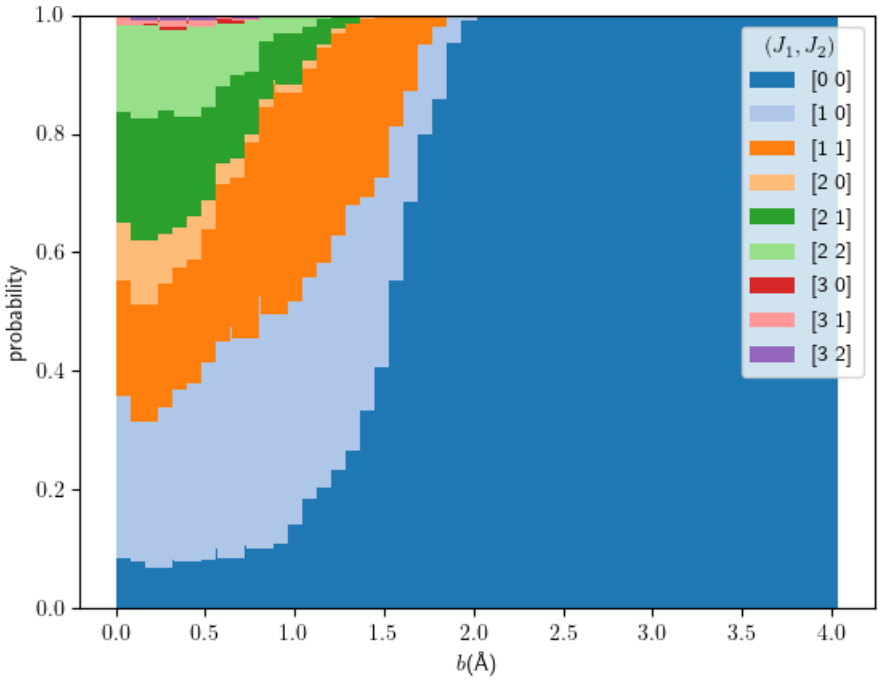


Figure 4.11: The outcomes of 30,000 trajectories with intial quantum numbers $(v_1, J_1, v_2, J_2) = (0, 0, 0, 0)$ *and final quantum numbers* $(0, J_1, 0, J_2)$*, with* $E_{coll} = 1.2\,\text{eV}$*, while varying the impact parameter b. J was estimated using equation* (4.10).

Table 4.3: The cross sections computed using the neural network potential compared to the results of Quéméner & Balakrishnan [122, Fig. 9] (results estimated from Fig. 9 in the paper), with intial quantum numbers $(0, 0, 0, 0)$ *and final quantum numbers* $(0, J_1, 0, J_2)$*, with* E_{coll} *=1.2 eV. (1) Estimating J from closest* E_{rot} *from* (4.8)*, (2) estimating J from* (4.10).

	Cross sections (Å^2)		
(J_1, J_2)	Our results (1)	Our results (2)	Quéméner & Balakrishnan
(1, 0)	2.9 ± 0.24	3.1 ± 0.23	-
(1, 1)	2.1 ± 0.19	2.4 ± 0.19	-
(2, 0)	0.59 ± 0.078	0.11 ± 0.029	2.4
(2, 1)	0.95 ± 0.11	0.44 ± 0.059	-
(2, 2)	0.7 ± 0.082	0.26 ± 0.039	1.9
(3, 0)	$(1.81 \pm 1.0)\times10^{-2}$	$(4.64 \pm 3.0)\times10^{-3}$	-
(3, 1)	$(7.08 \pm 2.3)\times10^{-2}$	$(1.07 \pm 0.6)\times10^{-2}$	-
(3, 2)	0.17 ± 0.033	$(8.17 \pm 4.5)\times10^{-3}$	-
(3, 3)	0.11 ± 0.023	-	-
(4, 0)	$(3.55 \pm 2.6)\times10^{-3}$	-	0.11
(4, 1)	$(1.14 \pm 0.7)\times10^{-2}$	-	-
(4, 2)	$(1.03 \pm 0.6)\times10^{-2}$	-	0.22
(4, 3)	$(9.30 \pm 5.0)\times10^{-3}$	-	-

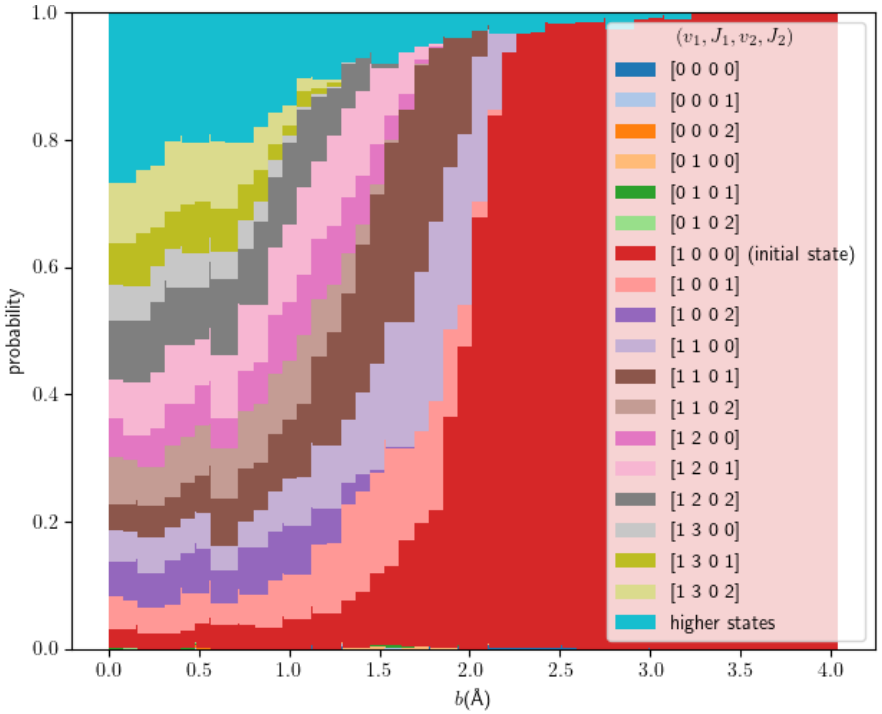


Figure 4.12: The outcomes of 30,000 trajectories with intial quantum numbers $(1, 0, 0, 0)$ *and final quantum numbers* (v_1, J_1, v_2, J_2)*, with* $E_{coll} = 0.87\,\text{eV}$*, while varying the impact parameter b. J was estimated using* (4.8).

Table 4.4: The cross sections computed using the neural network potential, with intial quantum numbers $(1, 0, 0, 0)$ *and final quantum numbers* (v_1, J_1, v_2, J_2)*, with* E_{coll} *=0.871 eV. J was estimated using* (4.10).

(v_1, J_1, v_2, J_2)	Cross sections (Å^2)
(1, 0, 0, 1)	1.4 ± 0.17
(1, 0, 0, 2)	0.35 ± 0.061
(1, 1, 0, 0)	2.2 ± 0.22
(1, 1, 0, 1)	2.3 ± 0.21
(1, 1, 0, 2)	0.48 ± 0.074
(1, 2, 0, 0)	0.58 ± 0.09
(1, 2, 0, 1)	0.91 ± 0.12
(1, 2, 0, 2)	0.59 ± 0.08
(1, 3, 0, 0)	0.12 ± 0.028
(1, 3, 0, 1)	0.17 ± 0.033
(1, 3, 0, 2)	0.25 ± 0.041

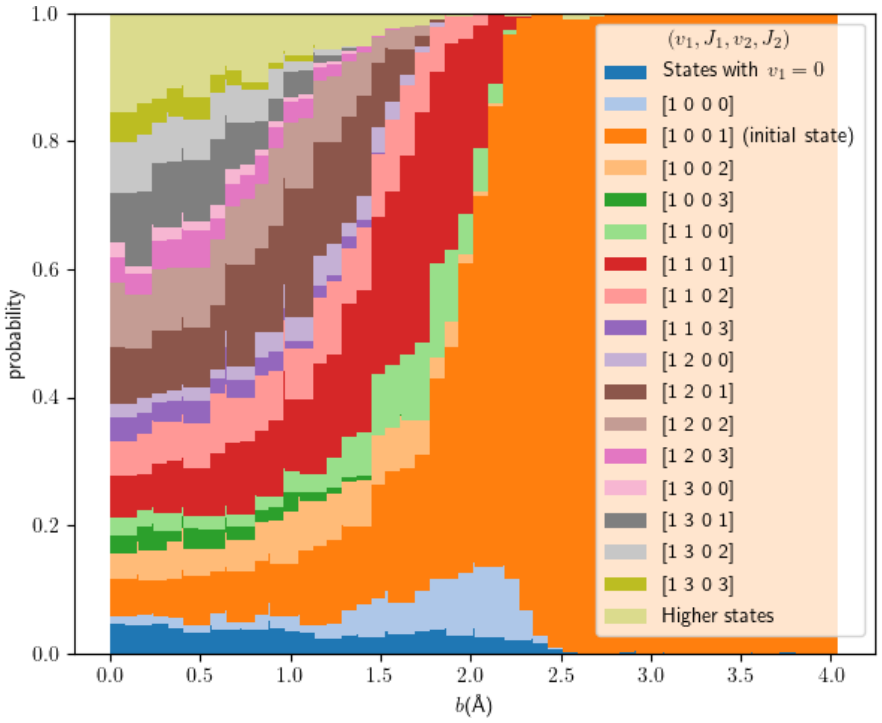


Figure 4.13: The outcomes of 30,000 trajectories with intial quantum numbers $(1, 0, 0, 1)$ *and final quantum numbers* (v_1, J_1, v_2, J_2)*, with* $E_{coll} = 0.87\,\mathrm{eV}$*, while varying the impact parameter* b*.* J *was estimated using* (4.8)*.*

Table 4.5: The cross sections computed using the neural network potential compared to the results of Dos Santos et al. [123, Fig. 2] (results estimated from Fig. 2 in the paper), with intial quantum numbers $(1, 0, 0, 1)$ *and final quantum numbers* (v_1, J_1, v_2, J_2)*, with* E_{coll} *=0.871 eV. J was estimated using* (4.10)*.*

	Cross sections (Å^2)	
(v_1, J_1, v_2, J_2)	Our results	Results Dos Santos et al.
(1, 0, 0, 0)	1.1 ± 0.17	-
(1, 0, 0, 2)	0.69 ± 0.12	-
(1, 0, 0, 3)	0.10 ± 0.032	0.58
(1, 1, 0, 0)	0.94 ± 0.14	-
(1, 1, 0, 1)	2.9 ± 0.26	-
(1, 1, 0, 2)	1.0 ± 0.14	-
(1, 1, 0, 3)	0.14 ± 0.037	-
(1, 2, 0, 0)	0.27 ± 0.063	-
(1, 2, 0, 1)	1.1 ± 0.12	2.3
(1, 2, 0, 2)	0.65 ± 0.095	-
(1, 2, 0, 3)	0.18 ± 0.043	0.58
(1, 3, 0, 0)	$(5.49 \pm 2.1)\times10^{-2}$	-
(1, 3, 0, 1)	0.24 ± 0.043	-
(1, 3, 0, 2)	0.18 ± 0.036	-
(1, 3, 0, 3)	$(7.13 \pm 2.0)\times10^{-2}$	-

Chapter 5

Conclusions and outlook

5.1 Conclusions

Various methods of fitting a full dimensional potential energy surface that is invariant with respect to permutation of identical atoms to ab initio data were discussed, and several of these methods are implemented for the H_2–H_2 system.

The PME NN method gave an RMSE of (19.58 ± 0.66) meV on the entire data set, and (6.50 ± 0.66) meV when possible outliers were removed. Unfortunately crossings or barely avoided crossings are a problem for a smooth PES. This method could still be very promising for applications where a smooth result is not important, such as when comparing the atomization energy of different molecules, as is done by Rupp et al. [21]. When comparing molecules with a different number of atoms, or different species, there is no smoothness needed, since we cannot 'gradually' add or remove an atom. It is also promising for graph neural networks, since a proximity matrix is essentially the adjacency matrix of a complete graph, with the nodes representing the nuclei and the edges representing the internuclear distances. Which means the PME method, as well as the PMI method, can be used to represent any arbitrary undirected graph, which means the possible applications are very broad.

The PIP method gave an RMSE of (10.70 ± 0.33) meV on the entire test set, and (4.37 ± 0.14) meV on the reduced test set. The PMI method using a linear regression gave an RMSE of (10.83 ± 0.45) meV on the entire data set, and (2.71 ± 0.21) meV on the reduced data set. The PMI method is promising; an important advantage of this approach (the PME method also has this advantage) is that it scales favourably to larger systems in comparison to the PIP method. A potential challenge is its apparent instability; small changes in the hyperparameters can cause an extreme difference in accuracy (more than 4 orders of magnitude difference). It seems very sensitive to outliers, a good performance on most of the data points can be disturbed by large errors for very few points. These outliers were especially a challenge for the neural network approach. A possible cause for this is the large dynamic range of the PMIs. A challenge for all these methods are the different scales that show up, the van der Waals well is small (millielectronvolts), the wall goes to electronvolts.

In the end, a potential energy surface from the PME method was used for full-dimensional quasiclassical trajectory simulations that were used to approximate cross sections for dissociation as well as vibrational and rotational energy transfer. The dissociation cross sections were mostly in good agreement with the literature, the rovibrational cross sections were not. There is room for improvement of the mapping of the quantum numbers to a classical state and vice versa.

In conclusion, the PME method gives to good results for a H_2–H_2 system, but cannot guarantee a smooth surface. The PMI method does result in a smooth surface and is a promising way of creating a permutation-

ally invariant potential energy surface, although its stability might be a problem and needs to be investigated more.

5.2 Outlook

There are a lot of possible directions to take this kind of research into the future, and this report only shows a part of the possibilities. There is a lot of literature of similar and also completely different approaches, as machine learning is a hot topic in physics right now. In this section we make some suggestions for future research. We also recommend reading the review paper by Manzhos and Carrington [69], which gives a pretty good overview of machine learning applied to PESs.

For example, in this report, the sampling, the machine learning and the molecular dynamics were treated separately, but a more integrated approach is also possible [124]. For example, DeePMD kit is a package specifically for deep learning applied to interatomic potentials and force fields [125]. It interfaces with TensorFlow, a machine learning framework similar to PyTorch, as well as LAMMPS. For example if we want to adjust the sampling based on how the training of the model is progressing, or to switch from machine learning back to ab initio calculations when a new configuration is encountered during a trajectory.

Another obvious target for future research is the CO_2–CO_2 system, since that was originally the goal of this project. This would also help to figure out how best to apply the PMI method to systems with more than one species.

5.2.1 Quantum Chemistry

For dissociation processes, different ab initio methods (such as multireference methods) are needed, and the counterpoise correction becomes important. Since the eventual goal is to model the dissociation of CO_2 in various vibrational states, this could be a reason to switch to a different method.

A big drawback of using Dalton that we only realized later into the project, is that CCSD calculations are not parallellized, but running multiple Dalton instances at the same time also does not work properly because it writes so much data to disk. This is also not great for an SSD. Most likely, one can realize more efficient sampling using another ab initio program than Dalton, that is either parallellized properly or can have more instances running at the same time, or even uses GPU computing. This page compares various quantum chemistry computer programs and this page lists several open source programs. Some options are CFOUR [126], OpenMolcas[127], Orca [128], LSDalton, PySCF [129] or Dirac [130] (Dirac is also relativistic). PySCF may be especially promising, since it has a Python interface (although the underlying code is compiled C/Fortran code and therefore fast) and can therefore probably be used more easily from a Python script. This is also a way to realize a more integrated approach.

5.2.2 Machine Learning

The sampling used in this report is trajectory sampling, which means using trajectories to sample molecular configurations, that way one obtains a set of states that are actually relevant during a trajectory. It is not perfect, however. The points that can have the biggest influence on the trajectory are often rare outlier points, for example, states where the molecules are very close together are rare, but they have a large impact on the trajectories of the molecules because the repulsion is so strong in those configurations. A combination with quasirandom sampling might help. It may also be a good idea to train a network on broader data than the region of interest; sample slightly beyond the range you're interested in.

A big advantage of the PME and PMI methods discussed in this report is that they scale better to bigger systems than the PIP method, so a natural place to take this research is systems with a lot more atoms. Using proximities with a cutoff then results in a sparse matrix. This can be combined with a method

that sums contributions from atoms (like 3.13), in that case the PME or PMI method could be applied to a matrix with $\boldsymbol{r}_{ik} \cdot \boldsymbol{r}_{ij}$ for each atom i [131, 132].

The PMI method and especially the PIP method were not investigated very thoroughly. For the PIP method it could be interesting to look at the effect of including various proximities (as we do for the PME and PMI methods), perhaps then it is not necessary to go up to high order, which would make this method scale a lot better as well.

There are also more options for possible descriptors to achieve a permutation invariant representation of an atom configuration. There is potential in taking inspiration from machine learning problems dealing with permutation invariance in general, not just in the physics literature. There seems to be surprisingly little overlap between the physics literature dealing with permutation invariance of PESs and the machine learning and mathematics literature on permutation invariance. One example of a paper than *does* take inspiration from machine learning literature is about message-passing neural networks [81], which takes inspiration from graph neural networks.

If not only the energy but also the gradient is obtained from ab initio calculations, this gradient can also be used for training. Nandi et al. [133] report a very impressive RMSE of only 1.1 meV using only 100 data points for CH_5, with configurations sampled from 3 trajectories (which they do not keep separate when dividing the data into training and testing sets, so they may have been helped by correlated data points). They have also made their code available on GitHub.

One thing that became obvious during this project was that the training can take an unusual number of epochs. Manzhos and Carrington [69] suggest using the Levenberg-Marquardt algorithm. This is also known as the damped least-squares (DLS) method and it essentially solves a non-linear least squares problem to determine the best possible update to the parameters for each batch. Combined with Extreme Learning Machines this could lead to faster training [134].

Also promising are neural network 'committees'; these are essentially multiple networks averaged together [135].

Options for using a network pre-trained on a different system may be worth exploring. This means less data is needed for fine-tuning it to a specific system. For example, the Murrell-Guo-Zúñiga potential can be used for both CO_2 and CS_2, although with different parameters. It might be possible to use a net pre-trained on CO_2 data as a starting point for a CS_2 net, which might mean the CS_2 data set does not have to be as big as it would have to be when starting from scratch, although this only works if the constants of the potential are similar.

The least-squares trick described in 3.2.4 was only used for checking convergence in this report, because it caused some bad overfitting. However, if the last layer of the neural network is a lot smaller, for example just 5 neurons, there are most likely not enough parameters for the least squares to cause bad overfitting. That means it might be possible to use this trick to get rid of a bias in the error; the average error will be zero.

5.2.3 Molecular Dynamics

Before using PESs resulting from some fitting procedure, it is a good idea to check the PES for holes, using for example the code CRYSTAL [136].

The current implementation of neural network in LAMMPS is not very flexible, for example changing the activation function requires changing the code and recompiling LAMMPS. It would probably be more convenient to call the network implemented in Python from the LAMMPS input script. It is possible to execute Python code from a LAMMPS script. Another option is to convert a PyTorch model to a neural model that can be executed from C++ with no dependency on Python. This can be achieved by using a Torch Script.

It seems like a good idea to use more of LAMMPS built-in options, such as `rotate` to give a random

rotation to molecules, instead of reinventing the wheel and doing it ourselves.

A more thorough and smart calculation of the cross sections, for example using Gaussian binning and sampling b using (4.11) may give more accurate results, which can help elucidate whether the discrepancies are due to the neural network or some other reason.

Investigating a bigger energy range for H_2–H_2 or higher initial quantum numbers would be interesting, especially ranges where more vibrational energy transfer happens. For the initial conditions in this report, mostly only rotational energy transfer happens.

Bibliography

[1] I. P. on Climate Change, *Climate Change 2014–Impacts, Adaptation and Vulnerability: Part A: Global and Sectoral Aspects: Volume 1, Global and Sectoral Aspects: Working Group II Contribution to the IPCC Fifth Assessment Report*. Cambridge University Press, 2014.

[2] C. B. Field, *Climate change 2014–Impacts, Adaptation and Vulnerability: Regional aspects*. Cambridge University Press, 2014.

[3] T. F. IPCC, 2013 [Stocker, D. Qin, G.-K. Plattner, M. Tignor, S. K. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex, P. M. Midgley, and others], "Climate change 2013: The physical science basis. contribution of working group i to the fifth assessment report of the intergovernmental panel on climate change," *Contribution of working group I to the fifth assessment report of the intergovernmental panel on climate change*, vol. 1535, 2013.

[4] K. Lackner, H.-J. Ziock, and P. Grimes, "Carbon dioxide extraction from air: Is it an option?," tech. rep., Los Alamos National Lab., NM (US), 1999.

[5] A. Fridman, *Plasma chemistry*. Cambridge university press, 2008.

[6] A. Lombardi, N. Faginas-Lago, L. Pacifici, and G. Grossi, "Energy transfer upon collision of selectively excited CO_2 molecules: State-to-state cross sections and probabilities for modeling of atmospheres and gaseous flows," *The Journal of chemical physics*, vol. 143, no. 3, p. 034307, 2015.

[7] S. Carter and J. Murrell, "Analytical two-valued potential energy functions for the ground state surfaces of CO_2 and CS_2," *Croatica Chemica Acta*, vol. 57, no. 3, pp. 355–365, 1984.

[8] A. Aguado, C. Suárez, and M. Paniagua, "Accurate global fit of the h4 potential energy surface," *The Journal of chemical physics*, vol. 101, no. 5, pp. 4004–4010, 1994.

[9] A. Brown, B. J. Braams, K. Christoffel, Z. Jin, and J. M. Bowman, "Classical and quasiclassical spectral analysis of ch 5+ using an ab initio potential energy surface," *The Journal of chemical physics*, vol. 119, no. 17, pp. 8790–8793, 2003.

[10] B. J. Braams and J. M. Bowman, "Permutationally invariant potential energy surfaces in high dimensionality," *International Reviews in Physical Chemistry*, vol. 28, no. 4, pp. 577–606, 2009.

[11] C.-C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE transactions on neural networks*, vol. 6, no. 1, pp. 144–156, 1995.

[12] S. Jung and S. S. Kim, "Control experiment of a wheel-driven mobile inverted pendulum using neural network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 297–303, 2008.

[13] H. M. Kamdar, M. J. Turk, and R. J. Brunner, "Machine learning and cosmological simulations–ii. hydrodynamical simulations," *Monthly Notices of the Royal Astronomical Society*, vol. 457, no. 2, pp. 1162–1179, 2016.

[14] E. P. Van Nieuwenburg, Y.-H. Liu, and S. D. Huber, "Learning phase transitions by confusion," *Nature Physics*, vol. 13, no. 5, pp. 435–439, 2017.

[15] J. Carifio, J. Halverson, D. Krioukov, and B. D. Nelson, "Machine learning in the string landscape," *Journal of High Energy Physics*, vol. 2017, no. 9, p. 157, 2017.

[16] P. Márquez-Neila, C. Fisher, R. Sznitman, and K. Heng, "Supervised machine learning for analysing spectra of exoplanetary atmospheres," *Nature astronomy*, vol. 2, no. 9, pp. 719–724, 2018.

[17] J. P. Coe, "Machine learning configuration interaction," *Journal of chemical theory and computation*, vol. 14, no. 11, pp. 5739–5749, 2018.

[18] J. P. Coe, "Machine learning configuration interaction for ab initio potential energy curves," *Journal of Chemical Theory and Computation*, vol. 15, no. 11, pp. 6179–6189, 2019.

[19] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, "Neural network models of potential energy surfaces," *The Journal of chemical physics*, vol. 103, no. 10, pp. 4129–4137, 1995.

[20] J. Ludwig and D. G. Vlachos, "Ab initio molecular dynamics of hydrogen dissociation on metal surfaces using neural networks and novelty sampling," *The Journal of chemical physics*, vol. 127, no. 15, p. 154716, 2007.

[21] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning," *Physical review letters*, vol. 108, no. 5, p. 058301, 2012.

[22] N. Artrith and A. Urban, "An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for tio2," *Computational Materials Science*, vol. 114, pp. 135–150, 2016.

[23] W. Li, Y. Ando, E. Minamitani, and S. Watanabe, "Study of li atom diffusion in amorphous li3po4 with neural network potential," *The Journal of chemical physics*, vol. 147, no. 21, p. 214106, 2017.

[24] G. Ferré, T. Haut, and K. Barros, "Learning molecular energies using localized graph kernels," *The Journal of chemical physics*, vol. 146, no. 11, p. 114107, 2017.

[25] B. J. Braams and J. M. Bowman, "Permutationally invariant potential energy surfaces in high dimensionality," *International Reviews in Physical Chemistry*, vol. 28, no. 4, pp. 577–606, 2009.

[26] N. S. Ostlund and A. Szabo, *Modern quantum chemistry: introduction to advanced electronic structure theory*. Macmillan, 1982.

[27] U. Kaldor and S. Wilson, *Theoretical chemistry and physics of heavy and superheavy elements*, vol. 11. Springer Science & Business Media, 2013.

[28] F. Jensen, *Introduction to computational chemistry*. John wiley & sons, second ed., 2007.

[29] G. A. Worth and L. S. Cederbaum, "Beyond born-oppenheimer: molecular dynamics through a conical intersection," *Annu. Rev. Phys. Chem.*, vol. 55, pp. 127–158, 2004.

[30] D. R. Yarkony, "Diabolical conical intersections," *Reviews of Modern Physics*, vol. 68, no. 4, p. 985, 1996.

[31] B. F. Curchod and T. J. Martínez, "Ab initio nonadiabatic quantum molecular dynamics," *Chemical reviews*, vol. 118, no. 7, pp. 3305–3336, 2018.

[32] A. Gordon and J. Avron, "Born-oppenheimer approximation near level crossing," *Physical review letters*, vol. 85, no. 1, p. 34, 2000.

[33] M. Bartolomei, M. I. Hernández, and J. Campos-Martínez, "Wave packet dynamics of H_2 (v_1= 8–14)+ H_2 (v_2= 0–2): The role of the potential energy surface on different reactive and dissociative processes," *The Journal of chemical physics*, vol. 122, no. 6, p. 064305, 2005.

[34] D. J. Griffiths and D. F. Schroeter, *Introduction to quantum mechanics*. Prentice Hall, Inc., 1995.

[35] R. J. Boyd, C. Sarasola, and J. M. Ugalde, "Intracule densities and electron correlation in the hydrogen molecule," *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 21, no. 14, p. 2555, 1988.

[36] S. F. Boys, "Electronic wave functions-i. a general method of calculation for the stationary states of any molecular system," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 200, no. 1063, pp. 542–554, 1950.

[37] J. S. Binkley, J. A. Pople, and W. J. Hehre, "Self-consistent molecular orbital methods. 21. small split-valence basis sets for first-row elements," *Journal of the American Chemical Society*, vol. 102, no. 3, pp. 939–947, 1980.

[38] D. E. Woon and T. H. Dunning Jr, "Gaussian basis sets for use in correlated molecular calculations. v. core-valence basis sets for boron through neon," *The Journal of chemical physics*, vol. 103, no. 11, pp. 4572–4585, 1995.

[39] T. Van Mourik, A. K. Wilson, K. A. Peterson, D. E. Woon, and T. H. Dunning Jr, "The effect of basis set superposition error (bsse) on the convergence of molecular properties calculated with the correlation consistent basis sets," in *Advances in Quantum Chemistry*, vol. 31, pp. 105–135, Elsevier, 1998.

[40] E. Carmona-Novillo, M. Bartolomei, M. I. Hernández, and J. Campos-Martínez, "Quasiclassical trajectory study of reactive and dissociative processes in H_2+H_2: Comparison with quantum-mechanical calculations," *The Journal of chemical physics*, vol. 126, no. 12, p. 124315, 2007.

[41] G. Herzberg, *Molecular spectra and molecular structure*, vol. 1. Read Books Ltd, 2013.

[42] A. Watanabe and H. Welsh, "Direct spectroscopic evidence of bound states of $(H_2)_2$ complexes at low temperatures," *Physical Review Letters*, vol. 13, no. 26, p. 810, 1964.

[43] A. Khan, T. Jahnke, S. Zeller, F. Trinter, M. Schöffler, L. P. H. Schmidt, R. Dörner, and M. Kunitski, "Visualizing the geometry of hydrogen dimers," *The Journal of Physical Chemistry Letters*, vol. 11, no. 7, pp. 2457–2463, 2020.

[44] Y. Fukumoto, H. Koizumi, and K. Makoshi, "Location of conical intersections by the pancharatnam connection and the sign-change theorem of longuet-higgins: a model calculation with the h4 potential surface," *Chemical physics letters*, vol. 313, no. 1-2, pp. 283–292, 1999.

[45] A. Boothroyd, P. Martin, W. Keogh, and M. Peterson, "An accurate analytic h 4 potential energy surface," *The Journal of chemical physics*, vol. 116, no. 2, pp. 666–689, 2002.

[46] Y.-S. M. Wu and A. Kuppermann, "Prediction of the effect of the geometric phase on product rotational state distributions and integral cross sections," *Chemical Physics Letters*, vol. 201, no. 1-4, pp. 178–186, 1993.

[47] D. W. Schwenke, "Calculations of rate constants for the three-body recombination of H_2 in the presence of H_2," *The Journal of chemical physics*, vol. 89, no. 4, pp. 2076–2091, 1988.

[48] J. Schaefer and W. Köhler, "Low temperature second virial coefficients of para-H_2 gas obtained from quantum mechanical pair correlation functions," *Zeitschrift für Physik D Atoms, Molecules and Clusters*, vol. 13, no. 3, pp. 217–229, 1989.

[49] P. Diep and J. K. Johnson, "An accurate H_2–H_2 interaction potential from first principles," *The Journal of Chemical Physics*, vol. 112, no. 10, pp. 4465–4473, 2000.

[50] K. Patkowski, W. Cencek, P. Jankowski, K. Szalewicz, J. B. Mehl, G. Garberoglio, and A. H. Harvey, "Potential energy surface for interactions between two hydrogen molecules," *The Journal of chemical physics*, vol. 129, no. 9, p. 094304, 2008.

[51] T. P. Van and U. K. Deiters, "Calculation of intermolecular potentials for H_2–H_2 and H_2–O_2 dimers ab initio and prediction of second virial coefficients," *Chemical Physics*, vol. 457, pp. 171–179, 2015.

[52] A. I. Boothroyd, J. E. Dove, W. J. Keogh, P. G. Martin, and M. R. Peterson, "Accurate abinitio potential energy computations for the h4 system: Tests of some analytic potential energy surfaces," *The Journal of chemical physics*, vol. 95, no. 6, pp. 4331–4342, 1991.

[53] R. J. Hinde, "A six-dimensional H_2–H_2 potential energy surface for bound state spectroscopy," *The Journal of chemical physics*, vol. 128, no. 15, p. 154308, 2008.

[54] X. Huang, D. W. Schwenke, S. A. Tashkun, and T. J. Lee, "An isotopic-independent highly accurate potential energy surface for CO_2 isotopologues and an initial 12c16o2 infrared line list," *The Journal of chemical physics*, vol. 136, no. 12, p. 124311, 2012.

[55] O. L. Polyansky, K. Bielska, M. Ghysels, L. Lodi, N. F. Zobov, J. T. Hodges, and J. Tennyson, "High-accuracy CO_2 line intensities determined from theory and experiment," *Physical Review Letters*, vol. 114, no. 24, p. 243001, 2015.

[56] T. Kozák and A. Bogaerts, "Splitting of CO_2 by vibrational excitation in non-equilibrium plasmas: a reaction kinetics model," *Plasma Sources Science and Technology*, vol. 23, no. 4, p. 045004, 2014.

[57] D. Cappelletti, F. Pirani, B. Bussery-Honvault, L. Gomez, and M. Bartolomei, "A bond–bond description of the intermolecular interaction energy: the case of weakly bound N_2–H_2 and N_2–N_2 complexes," *Physical Chemistry Chemical Physics*, vol. 10, no. 29, pp. 4281–4293, 2008.

[58] M. Bartolomei, F. Pirani, A. Laganà, and A. Lombardi, "A full dimensional grid empowered simulation of the CO_2 + CO_2 processes," *Journal of computational chemistry*, vol. 33, no. 22, pp. 1806–1819, 2012.

[59] J. N. Murrell and H. Guo, "Potential-energy functions for the ground states of CO_2, CS_2 and ocs, and dynamical calculations on the reaction o (1 d)+ cs (1 σ+)→ s (1 d)+ co (1 σ+)," *Journal of the Chemical Society, Faraday Transactions 2: Molecular and Chemical Physics*, vol. 83, no. 4, pp. 683–692, 1987.

[60] J. Zúñiga, A. Bastida, M. Alacid, and A. Requena, "Global potential energy surfaces for the CO_2 and CS_2 molecules," *Chemical physics letters*, vol. 313, no. 3-4, pp. 670–678, 1999.

[61] J. Zúñiga, M. Alacid, A. Bastida, F. J. Carvajal, and A. Requena, "Determination of a potential energy surface for co2using generalized internal vibrational coordinates," *Journal of molecular spectroscopy*, vol. 195, no. 1, pp. 137–146, 1999.

[62] K. Aidas, C. Angeli, K. L. Bak, V. Bakken, R. Bast, L. Boman, O. Christiansen, R. Cimiraglia, S. Coriani, P. Dahle, E. K. Dalskov, U. Ekström, T. Enevoldsen, J. J. Eriksen, P. Ettenhuber, B. Fernández, L. Ferrighi, H. Fliegl, L. Frediani, K. Hald, A. Halkier, C. Hättig, H. Heiberg, T. Helgaker, A. C. Hennum, H. Hettema, E. Hjertenæs, S. Høst, I.-M. Høyvik, M. F. Iozzi, B. Jansík, H. J. Aa. Jensen, D. Jonsson, P. Jørgensen, J. Kauczor, S. Kirpekar, T. Kjærgaard, W. Klopper, S. Knecht, R. Kobayashi, H. Koch, J. Kongsted, A. Krapp, K. Kristensen, A. Ligabue, O. B. Lutnæs, J. I. Melo, K. V. Mikkelsen, R. H. Myhre, C. Neiss, C. B. Nielsen, P. Norman, J. Olsen, J. M. H. Olsen, A. Osted, M. J. Packer, F. Pawlowski, T. B. Pedersen, P. F. Provasi, S. Reine, Z. Rinkevicius, T. A. Ruden, K. Ruud, V. V. Rybkin, P. Sałek, C. C. M. Samson, A. S. de Merás, T. Saue, S. P. A. Sauer, B. Schimmelpfennig, K. Sneskov, A. H. Steindal, K. O. Sylvester-Hvid, P. R. Taylor, A. M. Teale, E. I. Tellgren, D. P. Tew, A. J. Thorvaldsen, L. Thøgersen, O. Vahtras, M. A. Watson, D. J. D. Wilson, M. Ziolkowski, and H. Ågren, "The Dalton quantum chemistry program system," *WIREs Comput. Mol. Sci.*, vol. 4, no. 3, pp. 269–284, 2014.

[63] "Dalton, a molecular electronic structure program, release v2016.2 (2016)." `http://daltonprogram.org`. Accessed: 2020-04-17.

[64] A. Kramida, Yu. Ralchenko, J. Reader, and and NIST ASD Team. NIST Atomic Spectra Database (ver. 5.8), [Online]. Available: `https://physics.nist.gov/asd` [2020, November 23]. National Institute of Standards and Technology, Gaithersburg, MD., 2020.

[65] L. T. Xu, R. L. Jaffe, D. W. Schwenke, and M. Panesi, "The effect of the spin-forbidden co $(1\sigma+)$+ o (3p)→ co2 $(1\sigma g+)$ recombination reaction on afterbody heating of mars entry vehicles," in *47th AIAA Thermophysics Conference*, p. 3486, 2017.

[66] W. Kol/os and L. Wolniewicz, "Accurate adiabatic treatment of the ground state of the hydrogen molecule," *The Journal of Chemical Physics*, vol. 41, no. 12, pp. 3663–3673, 1964.

[67] W. Kolos and L. Wolniewicz, "Improved theoretical ground-state energy of the hydrogen molecule," *The Journal of Chemical Physics*, vol. 49, no. 1, pp. 404–410, 1968.

[68] W. Kołos and L. Wolniewicz, "Improved potential energy curve and vibrational energies for the electronic ground state of the hydrogen molecule," *Journal of molecular spectroscopy*, vol. 54, no. 2, pp. 303–311, 1975.

[69] S. Manzhos and T. Carrington Jr, "Neural network potential energy surfaces for small molecules and reactions," *Chemical Reviews*, 2020.

[70] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[72] S. Lorenz, A. Groß, and M. Scheffler, "Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks," *Chemical Physics Letters*, vol. 395, no. 4-6, pp. 210–215, 2004.

[73] J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," *Physical review letters*, vol. 98, no. 14, p. 146401, 2007.

[74] N. Artrith, A. Urban, and G. Ceder, "Efficient and accurate machine-learning interpolation of atomic energies in compositions with many species," *Physical Review B*, vol. 96, no. 1, p. 014112, 2017.

[75] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons," *Physical review letters*, vol. 104, no. 13, p. 136403, 2010.

[76] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in neural information processing systems*, pp. 3391–3401, 2017.

[77] E. Wagstaff, F. B. Fuchs, M. Engelcke, I. Posner, and M. Osborne, "On the limitations of representing functions on sets," *arXiv preprint arXiv:1901.09006*, 2019.

[78] S. Ravanbakhsh, J. Schneider, and B. Poczos, "Deep learning with sets and point clouds," *arXiv preprint arXiv:1611.04500*, 2016.

[79] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[80] X. Chen, X. Cheng, and S. Mallat, "Unsupervised deep haar scattering on graphs," in *Advances in Neural Information Processing Systems*, pp. 1709–1717, 2014.

[81] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[82] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," *arXiv preprint arXiv:1812.09902*, 2018.

[83] H. Maron, O. Litany, G. Chechik, and E. Fetaya, "On learning sets of symmetric elements," *arXiv preprint arXiv:2002.08599*, 2020.

[84] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds," *arXiv preprint arXiv:1802.08219*, 2018.

[85] Z. Xie and J. M. Bowman, "Permutationally invariant polynomial basis for molecular energy surface fitting via monomial symmetrization," *Journal of Chemical Theory and Computation*, vol. 6, no. 1, pp. 26–34, 2010.

[86] M. Ayouz and D. Babikov, "Improved potential energy surface of ozone constructed using the fitting by permutationally invariant polynomial function.," *Advances in Physical Chemistry*, 2012.

[87] J. Li, Y. Wang, B. Jiang, J. Ma, R. Dawes, D. Xie, J. M. Bowman, and H. Guo, "Communication: A chemically accurate global potential energy surface for the ho+ co→ h+ co2 reaction," 2012.

[88] Y. Paukku, K. R. Yang, Z. Varga, and D. G. Truhlar, "Global ab initio ground-state potential energy surface of n4," *The Journal of chemical physics*, vol. 139, no. 4, p. 044309, 2013.

[89] J. D. Bender, P. Valentini, I. Nompelis, Y. Paukku, Z. Varga, D. G. Truhlar, T. Schwartzentruber, and G. V. Candler, "An improved potential energy surface and multi-temperature quasiclassical trajectory calculations of n2+ n2 dissociation reactions," *The Journal of chemical physics*, vol. 143, no. 5, p. 054304, 2015.

[90] B. Jiang and H. Guo, "Permutation invariant polynomial neural network approach to fitting potential energy surfaces," *The Journal of chemical physics*, vol. 139, no. 5, p. 054112, 2013.

[91] J. Li, Z. Varga, D. G. Truhlar, and H. Guo, "Many-body permutationally invariant polynomial neural network potential energy surface for n4," *Journal of Chemical Theory and Computation*, vol. 16, no. 8, pp. 4822–4832, 2020.

[92] C. Yang, M. Sun, Z. Liu, and C. Tu, "Fast network embedding enhancement via high order proximity approximation.," in *IJCAI*, pp. 3894–3900, 2017.

[93] H. Anton and C. Rorres, *Elementary linear algebra: applications version*. John Wiley & Sons, 2013.

[94] J. C. Gower, "A modified leverrier-faddeev algorithm for matrices with multiple-eigenvalues," *Linear Algebra and its Applications*, vol. 31, no. June, pp. 61–70, 1980.

[95] P. Ren, R. C. Wilson, and E. R. Hancock, "Characteristic polynomial analysis on matrix representations of graphs," in *International Workshop on Graph-Based Representations in Pattern Recognition*, pp. 243–252, Springer, 2009.

[96] P. Ren, T. Aleksić, R. C. Wilson, and E. R. Hancock, "Hypergraphs, characteristic polynomials and the ihara zeta function," in *International Conference on Computer Analysis of Images and Patterns*, pp. 369–376, Springer, 2009.

[97] K. H. Rosen, *Handbook of discrete and combinatorial mathematics (2nd edition)*. CRC press, 2018.

[98] O. F. Inc., "The on-line encyclopedia of integer sequences." `http://oeis.org`, 2020. Accessed: 2020-12-04.

[99] S. S. Skiena, *The algorithm design manual, the second edition*. Springer International Publishing, 2008.

[100] B. D. McKay *et al.*, "Practical graph isomorphism," 1981.

[101] U. Schöning, "Graph isomorphism is in the low hierarchy," in *Annual Symposium on Theoretical Aspects of Computer Science*, pp. 114–124, Springer, 1987.

[102] W. Bosma, J. Cannon, and C. Playoust, "The magma algebra system i: The user language," *Journal of Symbolic Computation*, vol. 24, no. 3-4, pp. 235–265, 1997.

[103] C. Qu and J. M. Bowman, "A fragmented, permutationally invariant polynomial approach for potential energy surfaces of large molecules: Application to n-methyl acetamide," *The Journal of chemical physics*, vol. 150, no. 14, p. 141101, 2019.

[104] R. Conte, C. Qu, P. L. Houston, and J. M. Bowman, "Efficient generation of permutationally invariant potential energy surfaces for large molecules," *Journal of Chemical Theory and Computation*, vol. 16, no. 5, pp. 3264–3272, 2020.

[105] L. Lovász, *Large networks and graph limits*, vol. 60. American Mathematical Soc., 2012.

[106] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[107] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 52–66, 2018.

[108] P. Meltzer, M. D. G. Mallea, and P. J. Bentley, "Pinet: A permutation invariant graph neural network for graph classification," *arXiv preprint arXiv:1905.03046*, 2019.

[109] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[110] G. Levy, "An introduction to quasi-random numbers," *Numerical Algorithms Group Ltd., http://www.nag. co. uk/IndustryArticles/introduction_to_quasi_random_numbers. pdf (last accessed in April 10, 2012)*, p. 143, 2002.

[111] 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.

[112] "Pytorch." `https://pytorch.org/`. Accessed: 2020-04-17.

[113] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[114] A. Kamath, R. A. Vargas-Hernández, R. V. Krems, T. Carrington Jr, and S. Manzhos, "Neural networks vs gaussian process regression for representing potential energy surfaces: A comparative study of fit quality and vibrational spectrum accuracy," *The Journal of chemical physics*, vol. 148, no. 24, p. 241702, 2018.

[115] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters," *The Journal of chemical physics*, vol. 76, no. 1, pp. 637–649, 1982.

[116] D. di Domenico, M. I. Hernández, and J. Campos-Martínez, "A time-dependent wave packet approach for reaction and dissociation in h2+ h2," *Chemical physics letters*, vol. 342, no. 1-2, pp. 177–184, 2001.

[117] M. Capitelli, G. Colonna, and A. D'Angola, *Molecular Partition Function: Vibrational, Rotational and Electronic Contributions*, pp. 79–99. New York, NY: Springer New York, 2012.

[118] K. P. Huber and G. H. Herzberg, *Constants of Diatomic Molecules*. Gaithersburg MD, 20899: National Institute of Standards and Technology, 2020. Accessed: 2020-04-09.

[119] I. Suzuki, "General anharmonic force constants of carbon dioxide," *Journal of Molecular Spectroscopy*, vol. 25, no. 4, pp. 479–500, 1968.

[120] "Lammps molecular dynamics simulator." `https://lammps.sandia.gov/`. Accessed: 2020-04-17.

[121] A. Ceballos, E. Garcia, A. Rodrıguez, and A. Laganà, "A quasiclassical trajectory study of the h2+ h2 reaction," *Chemical physics letters*, vol. 305, no. 3-4, pp. 276–284, 1999.

[122] G. Quéméner and N. Balakrishnan, "Quantum calculations of H_2–H_2 collisions: From ultracold to thermal energies," *The Journal of chemical physics*, vol. 130, no. 11, p. 114303, 2009.

[123] S. F. Dos Santos, N. Balakrishnan, R. C. Forrey, and P. Stancil, "Vibration-vibration and vibration-translation energy transfer in h2-h2 collisions: A critical test of experiment with full-dimensional quantum dynamics," *The Journal of Chemical Physics*, vol. 138, no. 10, p. 104302, 2013.

[124] V. Botu and R. Ramprasad, "Adaptive machine learning framework to accelerate ab initio molecular dynamics," *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1074–1083, 2015.

[125] H. Wang, L. Zhang, J. Han, and E. Weinan, "Deepmd-kit: A deep learning package for many-body potential energy representation and molecular dynamics," *Computer Physics Communications*, vol. 228, pp. 178–184, 2018.

[126] J. F. Stanton, J. Gauss, L. Cheng, M. E. Harding, D. A. Matthews, and P. G. Szalay, "CFOUR, Coupled-Cluster techniques for Computational Chemistry, a quantum-chemical program package." With contributions from A.A. Auer, R.J. Bartlett, U. Benedikt, C. Berger, D.E. Bernholdt, S. Blaschke, Y. J. Bomble, S. Burger, O. Christiansen, D. Datta, F. Engel, R. Faber, J. Greiner, M. Heckert, O. Heun, M. Hilgenberg, C. Huber, T.-C. Jagau, D. Jonsson, J. Jusélius, T. Kirsch, K. Klein, G.M. KopperW.J. Lauderdale, F. Lipparini, T. Metzroth, L.A. Mück, D.P. O'Neill, T. Nottoli, D.R. Price, E. Prochnow, C. Puzzarini, K. Ruud, F. Schiffmann, W. Schwalbach, C. Simmons, S. Stopkowicz, A. Tajti, J. Vázquez, F. Wang, J.D. Watts and the integral packages MOLECULE (J. Almlöf and P.R. Taylor), PROPS (P.R. Taylor), ABACUS (T. Helgaker, H.J. Aa. Jensen, P. Jørgensen, and J. Olsen), and ECP routines by A. V. Mitin and C. van Wüllen. For the current version, see http://www.cfour.de.

[127] I. Fdez. Galván, M. Vacher, A. Alavi, C. Angeli, F. Aquilante, J. Autschbach, J. J. Bao, S. I. Bokarev, N. A. Bogdanov, R. K. Carlson, *et al.*, "Openmolcas: From source code to insight," *Journal of chemical theory and computation*, vol. 15, no. 11, pp. 5925–5964, 2019.

[128] F. Neese, "Software update: the orca program system, version 4.0," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 8, no. 1, p. e1327, 2018.

[129] Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, *et al.*, "Pyscf: the python-based simulations of chemistry framework," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 8, no. 1, p. e1340, 2018.

[130] DIRAC, a relativistic ab initio electronic structure program, Release DIRAC19 (2019), written by A. S. P. Gomes, T. Saue, L. Visscher, H. J. Aa. Jensen, and R. Bast, with contributions from I. A. Aucar, V. Bakken, K. G. Dyall, S. Dubillard, U. Ekström, E. Eliav, T. Enevoldsen, E. Faßhauer, T. Fleig, O. Fossgaard, L. Halbert, E. D. Hedegård, B. Heimlich–Paris, T. Helgaker, J. Henriksson, M. Iliaš, Ch. R. Jacob, S. Knecht, S. Komorovský, O. Kullie, J. K. Lærdahl, C. V. Larsen, Y. S. Lee, H. S. Nataraj, M. K. Nayak, P. Norman, G. Olejniczak, J. Olsen, J. M. H. Olsen, Y. C. Park, J. K. Pedersen, M. Pernpointner, R. di Remigio, K. Ruud, P. Sałek, B. Schimmelpfennig, B. Senjean, A. Shee, J. Sikkema, A. J. Thorvaldsen, J. Thyssen, J. van Stralen, M. L. Vidal, S. Villaume, O. Visser, T. Winther, and S. Yamamoto (available at `http://dx.doi.org/10.5281/zenodo.3572669`, see also `http://www.diracprogram.org`).

[131] H. Weyl, *The classical groups: their invariants and representations*, vol. 45. Princeton university press, 1946.

[132] A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," *Physical Review B*, vol. 87, no. 18, p. 184115, 2013.

[133] A. Nandi, C. Qu, and J. M. Bowman, "Using gradients in permutationally invariant polynomial potential fitting: A demonstration for ch4 using as few as 100 configurations," *Journal of chemical theory and computation*, vol. 15, no. 5, pp. 2826–2835, 2019.

[134] Y.-l. Zhang, X.-y. Zhou, and B. Jiang, "Accelerating the construction of neural network potential energy surfaces: a fast hybrid training algorithm," *Chinese Journal of Chemical Physics*, vol. 30, no. 6, p. 727, 2018.

[135] S. Manzhos, R. Dawes, and T. Carrington, "Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces," *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1012–1020, 2015.

[136] A. Pandey and B. Poirier, "An algorithm to find (and plug)"holes" in multi-dimensional surfaces," *The Journal of Chemical Physics*, vol. 152, no. 21, p. 214102, 2020.

[137] E. F. Beckenbach and R. Weller, *Modern Mathematics for the Engineer: First Series*. Courier Corporation, 2013.

[138] "scipy.spatial.transform.rotation." `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.html`. Accessed: 2020-10-21.

[139] J. Arvo, "Fast random rotation matrices," in *Graphics Gems III (IBM Version)*, pp. 117–120, Elsevier, 1992.

[140] "scipy.stats.maxwell." `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.maxwell.html`. Accessed: 2020-10-21.

[141] "Jmol: an open-source java viewer for chemical structures in 3d." `http://www.jmol.org/`.

[142] "Derivatives of eigenvalues - mathematics stack exchange." `https://math.stackexchange.com/questions/2588473/derivatives-of-eigenvalues`. Accessed: 2020-05-23.

[143] 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.

[144] Z. Xie, B. J. Braams, and J. M. Bowman, "Ab initio global potential-energy surface for h 5+→ h 3++ h 2," *The Journal of chemical physics*, vol. 122, no. 22, p. 224307, 2005.

Appendix A

Coordinate systems

A.1 Coordinates H_2–H_2

To generate initial starting positions for the trajectories, 6 coordinates are used for H_2–H_2, since the system has 6 degrees of freedom (4 atoms, 3 coordinates per atom, minus 6 coordinates for translational and rotational invariance. The coordinates that are used for this system in this report are shown in Figure A.1. The origin is defined to be at the center of mass of the entire system. The z-axis is defined to go through the centers of mass of each molecule. The direction of the y-axis is chosen such that molecule 1 lies in the xz-plane (so the y-coordinate of each atom in this molecule is zero). The angle the hydrogen atoms of molecule 1 make with the z-axis is θ_1. The angle of molecule 2 with the z-axis is θ_2, and the azimuthal angle of molecule 2 is ϕ_2 (there is no ϕ_1, since that would always be zero).

A.2 Coordinates CO_2–CO_2

The coordinates R, r_{a1}, r_{a2}, r_{b1}, r_{b2}, ϕ_a, ϕ_b that are used for the CO_2–CO_2 system in this report are shown in Figure A.2. These coordinates describe the internal state of both molecules (bond lengths r_{i1}, r_{i2} and internal angle ϕ_i) and the distance between their centers of mass R. Since the system has 12 degrees of freedom (6 atoms, 3 coordinates per atom, minus 6 coordinates for translational and rotational invariance) this is not enough to specify a unique configuration, but specific coordinates to describe their orientation were not worked out, since for this project there is no physically relevant information in those coordinates.

The origin is defined to be at the center of mass of the entire system. The z-axis is defined to go through the centers of mass of each molecule.

A.3 Extracting coordinates from Cartesian coordinates

To analyze the results of the trajectories we need the distance between the centers of mass of each molecule R, and the internal bond lengths (r_1 and r_2 for H_2–H_2, r_{a1}, r_{a2}, r_{b1}, r_{b2}, for CO_2–CO_2). For CO_2–CO_2 we need the internal bond angles ϕ_a and ϕ_b as well.

For both systems R is just the distance between the center of mass of each molecule

$$R = \left\| \frac{\sum_i^{N_1} m_i \boldsymbol{x}_i}{\sum_i^{N_1} m_i} - \frac{\sum_j^{N_2} m_j \boldsymbol{x}_j}{\sum_j^{N_2} m_j} \right\|, \tag{A.1}$$

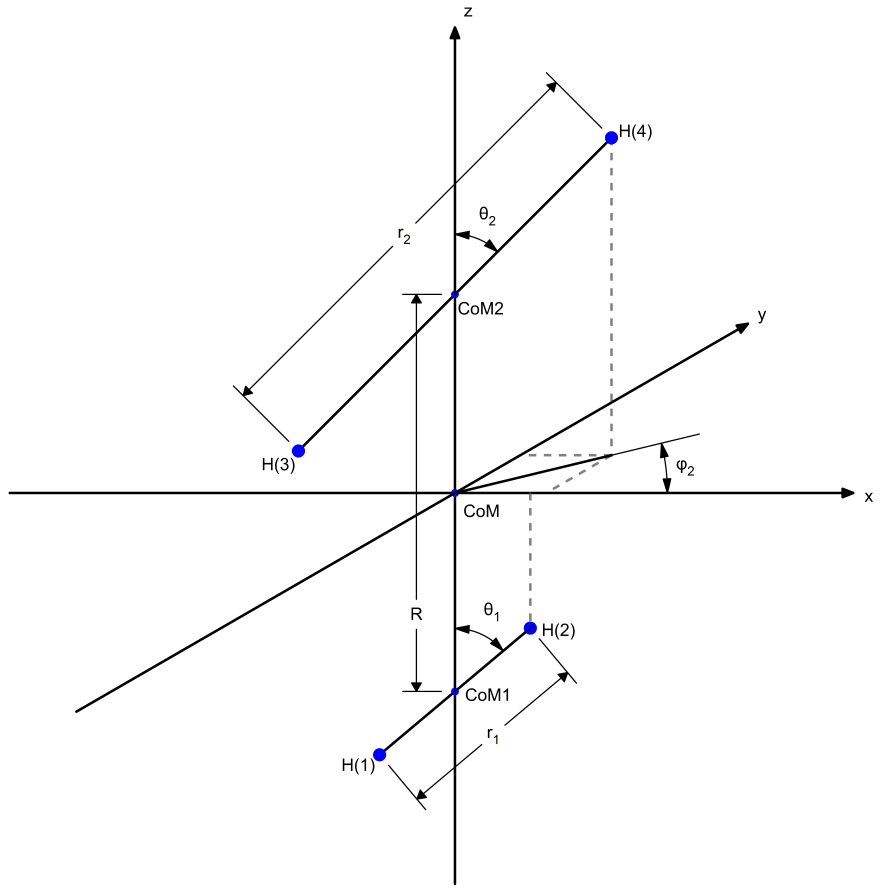


Figure A.1: The coordinates R, r_1, r_2, θ_1, θ_2 and ϕ_2 used to describe the H_2–H_2 system.

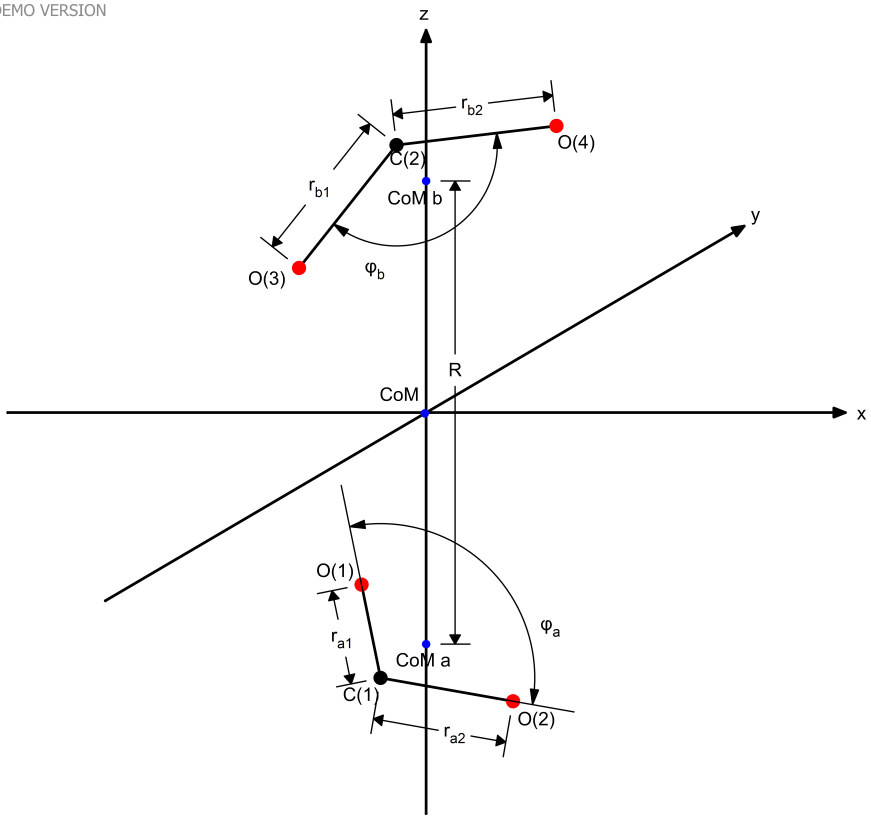


Figure A.2: The coordinates R, r_{a1}, r_{a2}, r_{b1}, r_{b2}, ϕ_a, ϕ_b used to describe the CO_2–CO_2 system.

with i iterating over the N_1 atoms in molecule 1 and j iterating over the N_2 atoms in molecule 2. Any internal bond length r (r_1 and r_2 for H_2–H_2, r_{a1}, r_{a2}, r_{b1}, r_{b2}, for CO_2–CO_2) between atoms i and j is just $||\boldsymbol{x}_i - \boldsymbol{x}_j||$.

The internal angle ϕ of a CO_2-molecule can be calculated using

$$\phi = \arccos\left(\frac{\boldsymbol{r}_{\mathrm{CO},1} \cdot \boldsymbol{r}_{\mathrm{CO},2}}{\|\boldsymbol{r}_{\mathrm{CO},1}\| \, \|\boldsymbol{r}_{\mathrm{CO},2}\|}\right), \tag{A.2}$$

with $\boldsymbol{r}_{\mathrm{CO},1}$ the vector from the carbon atom to the first oxygen atom and $\boldsymbol{r}_{\mathrm{CO},2}$ the vector from the carbon atom to the other oxygen atom. Note that when calculated this way, ϕ is always less than or equal to π.

A.4 Generating a random starting state

To generate a random configuration, the coordinates R, r_1, r_2 (for H_2–H_2) or R, r_{a1}, r_{a2}, r_{b1}, r_{b2}, ϕ_a, ϕ_b (for CO_2–CO_2) can be picked uniformly from a chosen range, but determining the relative orientations of the molecules requires more care. Picking random values for the angles describing their orientations from a uniform distribution, or rotating each molecule randomly around the x-, y- and z-axis is not good enough, because the molecules should have a random orientation that is spherically uniform. See Figure A.3 for an illustration of the problem.

The orientation of a hydrogen molecule can be described by just one vector, for example one from atom 1 to atom 2 $\boldsymbol{r}_{ij}$. A way to generate a random orientation for this molecule is to pick a random direction for this vector from a spherically uniform distribution. One can pick a vector with a random direction by picking each coordinate x, y and z from a normal distribution [137, §12.11]. Fortunately, the SciPy library has a `Rotation` class that has a classmethod `Rotation.random()` that can generate a random rotation [138].

Unless the internal angle is exactly π, the orientation of a CO_2 molecule needs to be described by two vectors, so just picking one random direction is not enough. A uniformly distributed random rotation of any molecule can be generated by a performing a random rotation around the z-axis (the angle can be picked uniformly from $[0, 2\pi]$), and then rotating the north pole (unit vector $\boldsymbol{e}_z$) to a random position [139], for which SciPy's `Rotation.random()` can be used once again.

Generating the velocity of the atoms of each molecule (leaving out the rotational velocity of each molecule for now) needs to take into account the absolute velocity $\|v\|$ and the impact parameter b.

If the desired collisional energy E_{coll} (defined as the total kinetic energy in the reference frame with a stationary center of mass) is given, $\|v\|$ can be calculated as follows. If the two molecules have the same mass m_{mol} (which is the case for both the H_2–H_2 and the CO_2–CO_2 system), the absolute value of their initial velocity $\|v\|$ will be the same and can be calculated from the desired collision energy using

$$E_{\mathrm{coll}} = 2 \cdot \frac{1}{2} m_{\mathrm{mol}} \|v\|^2, \tag{A.3}$$

which means

$$\|v\| = \sqrt{\frac{E_{\mathrm{coll}}}{m_{\mathrm{mol}}}}. \tag{A.4}$$

If we want to calculate cross sections for a given collisional temperature T_{coll}, then $\|v\|$ should be sampled from the Maxwell-Boltzmann distribution that depends on T_{coll}. The Maxwell-Boltzmann distribution looks like this

$$P(v) = \left(\frac{m}{2\pi kT}\right)^{3/2} 4\pi v^2 \exp\left(-\frac{mv^2}{2kT}\right). \tag{A.5}$$

To sample this, the SciPy library has a random variable class `scipy.stats.maxwell` that can generate a random variable from the Maxwell-Boltzmann distribution [140].

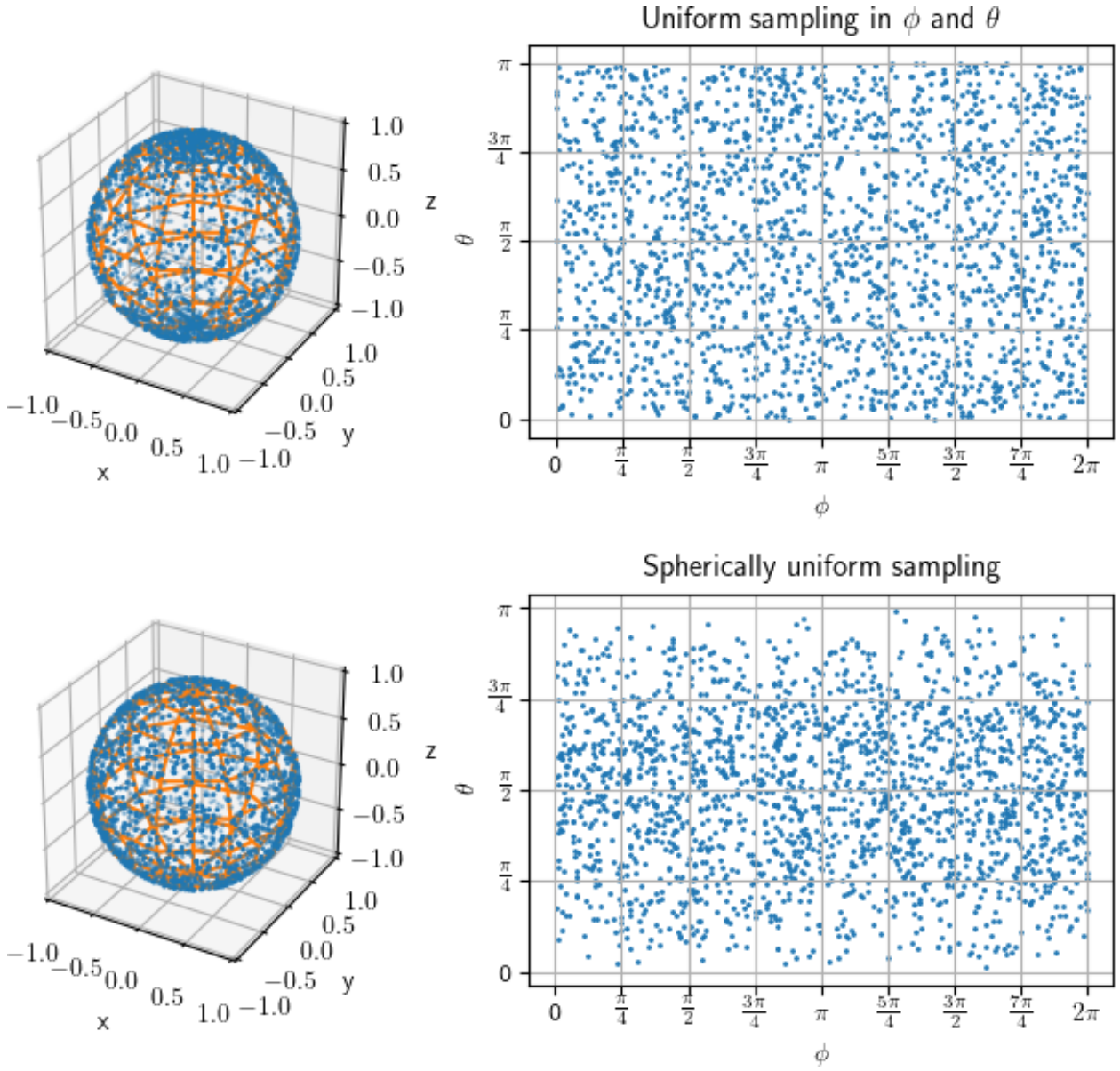


Figure A.3: If the polar angle θ and azimuthal angle ϕ are picked from a uniform distribution, the resulting distribution of orientations will not be uniform. On a sphere, this means relatively more data points near the north and south poles are chosen. For H_2 orientations this means the molecules will disproportionately often be oriented mostly along the z-axis (meaning with small θ).

The polar angle θ_v of the velocity depends on the desired impact parameter b (which can be chosen uniformly from a chosen range)

$$\theta_v = \arcsin\left(\frac{b}{R}\right), \tag{A.6}$$

and the azimuthal angle ϕ_v can be chosen randomly from a distribution $[0, 2\pi]$.

In Cartesian coordinates, this results in the following v for the molecule that moves in negative z-direction in Figure A.1 and Figure A.2

$$v_x = \|v\| \cos\phi_v \sin\theta_v \tag{A.7}$$
$$v_y = \|v\| \sin\phi_v \sin\theta_v \tag{A.8}$$
$$v_z = -\|v\| \cos\theta_v, \tag{A.9}$$

while for the other molecule each component switches its sign. These velocities are the same for all atoms in one molecule, it is only when a rotation is added that the atoms get different velocities.

A.5 Adding rotation

To add a rotation to the initial state of the molecules, we add some extra velocity to the atoms relative to each other. We first need to know the rotational energy E_{rot} to add. If we know exactly what rotational quantum each molecule needs to have, E_{rot} can be taken directly from 4.8.

If instead we want to calculate cross sections for a given rotational temperature T_{rot}, then E_{rot} for each molecule should be sampled from the same rotational energy distribution, which is a Boltzmann distribution that depends on T_{rot}. This means that the cross sections will be thermally overaged over the rotational state [6]. The Boltzmann distribution looks like this

$$P(J) = \frac{e^{-\epsilon_J/kT}}{\sum_{i=1}^{M} e^{-\epsilon_i/kT}}. \tag{A.10}$$

The sum runs over all values of J in (4.8) that have a rotational energy below the dissociation limit. For H_2, this mean up to state $J = 25$

Then a direction for the axis of rotation $\hat{\boldsymbol{\omega}}$ will be chosen, which is the normalized version of the pseudovector representing the angular velocity $\boldsymbol{\omega}$. This direction should be chosen perpendicular to the interatomic axis in H_2, which is possible by generating a random vector and then subtracting the component of this vector parallel to the interatomic axis.

The position of an atom relative to the center of mass of the molecule it is a part of is

$$\boldsymbol{r}_i = \boldsymbol{x}_i - \boldsymbol{x}_{\text{CoM}}, \tag{A.11}$$

with $\boldsymbol{x}_1$ the position of atom i in Cartesian coordinates and $\boldsymbol{x}_{\text{CoM}}$ the position of the center of mass. The component of $\boldsymbol{r}_i$ perpendicular to the axis of rotation is then

$$\boldsymbol{r}_{i,\perp} = -\hat{\boldsymbol{\omega}} \times (\hat{\boldsymbol{\omega}} \times \boldsymbol{r}_i), \tag{A.12}$$

or, equivalently

$$\boldsymbol{r}_{i,\perp} = \boldsymbol{r}_i - \boldsymbol{r}_{i,\parallel} = \boldsymbol{r}_i - (\hat{\boldsymbol{\omega}} \cdot \boldsymbol{r}_i)\,\hat{\boldsymbol{\omega}}. \tag{A.13}$$

The moment of inertia of a molecule consisting of N atoms is

$$I = \sum_{i=0}^{N} m_i ||\boldsymbol{r}_{i,\perp}||^2. \tag{A.14}$$

From the fact that $E_{\text{rot}} = \frac{1}{2} I\omega^2$, the magnitude of ω can then be calculated. The extra velocity on the atoms is then

$$\boldsymbol{v}_{i,\perp} = \boldsymbol{\omega} \times \boldsymbol{r}_{i,\perp}. \tag{A.15}$$

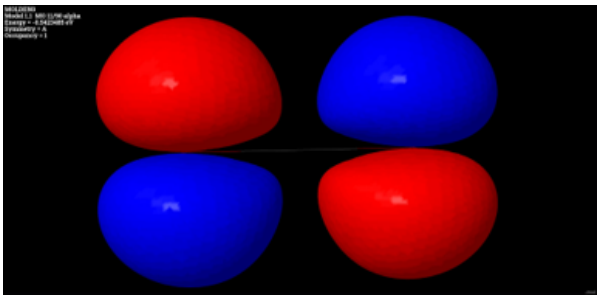
Appendix B

CO_2 peak in ϕ graph

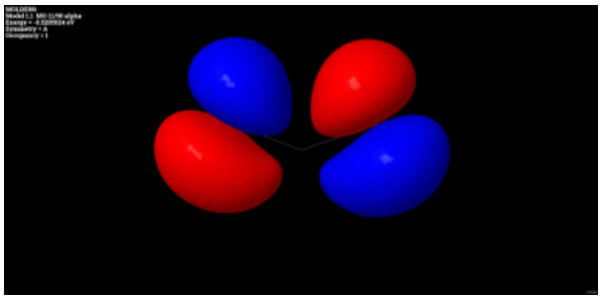
Figure B.1 shows the 3D wave function for the ground state of CO_2, for various values of ϕ. In Figures B.1a, B.1b and B.1c the two oxygen atoms are in anti-bonding mode and repel each other. However, as they get forced closer and closer together, at some point (near $\phi = 76^\circ$) the wave function abruptly changes shape drastically (Figures B.1d, B.1e and B.1f).

Figure B.2 shows the energy of one CO_2-molecule when varying the O-C-O angle ϕ, for various ab initio methods. It shows a sharp peak in the energy for the self-consistent field method (SCF) and the coupled-cluster methods near this point. This is because a partial bond is formed between the two oxygen atoms, which means the configuration is less a carbon dioxide molecule (O=C=O) and more an oxygen molecule (O−O) that happens to have a carbon atom close to it.

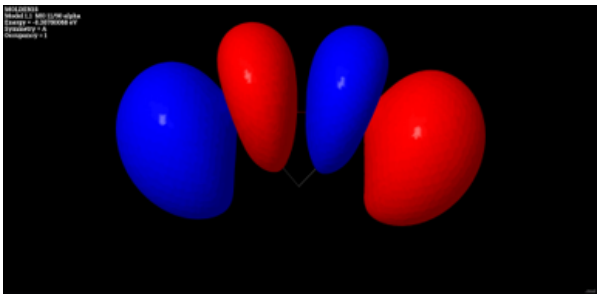
In reality, the real wave function will be some mixture of these states. This means that configurations like these really need a multireference calculation such as the complete active space SCF method (CASSCF) or the complete active space perturbation theory (CASPT2) method to properly describe them, which is why these methods show no peak here. This means that the sharp peak is an unphysical artifact, but the sudden bend in the energy for the CASSCF and the CASPT2 method is most likely physical.



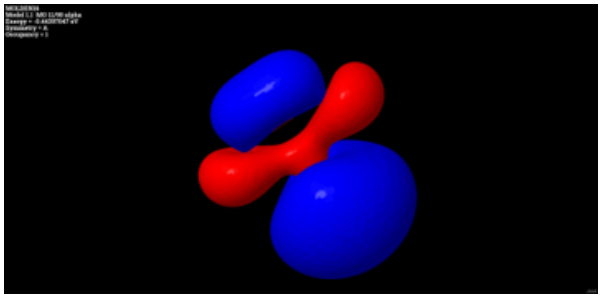
(a) $\phi = 180^{\circ}$



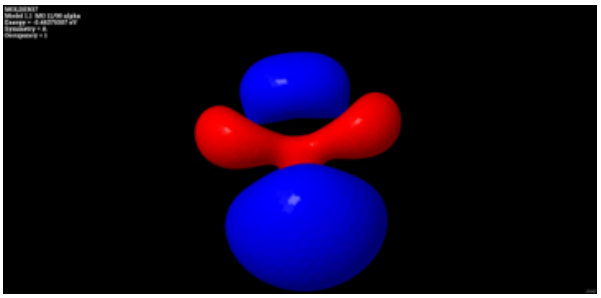
(b) $\phi = 141^{\circ}$



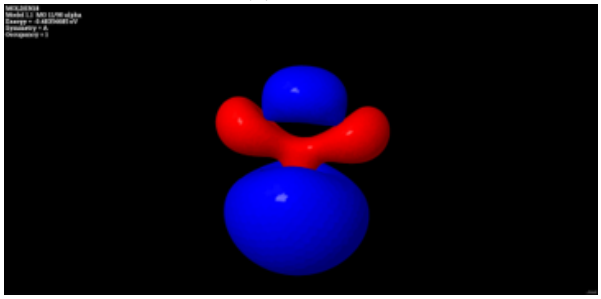
(c) $\phi = 82.5^{\circ}$



(d) $\phi = 76^{\circ}$



(e) $\phi = 69.5^{\circ}$



(f) $\phi = 63^{\circ}$

Figure B.1: Images showing the 3D wave function for the ground state of CO_2, for gradually decreasing ϕ. Images created by dr. Jos Suijker, using CCSD calculations in the program CFOUR[126], visualized using Jmol[141].

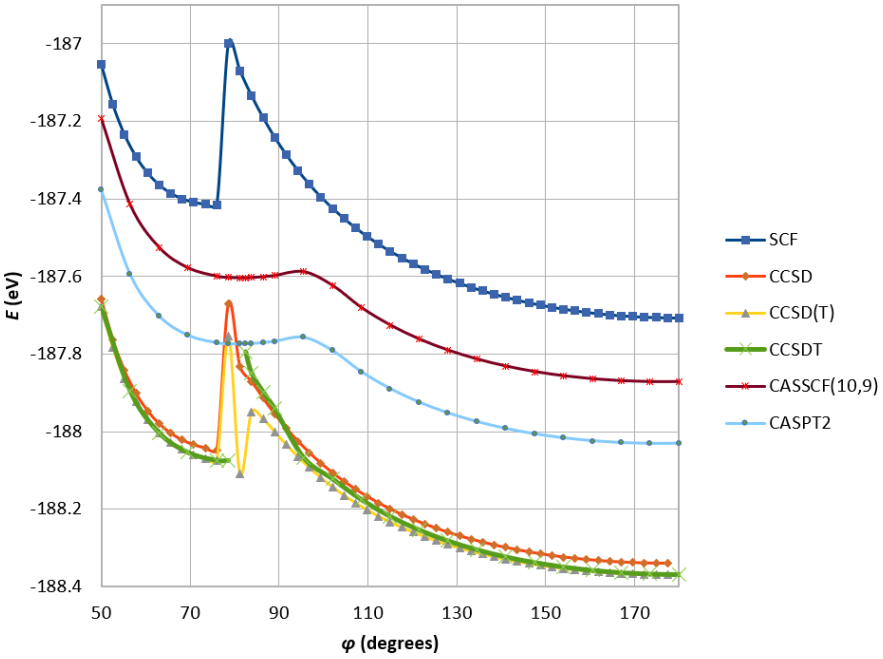


Figure B.2: The energy of a CO_2-molecule when varying the O-C-O angle ϕ, for various ab initio methods. Calculated by dr. Jos Suijker using CFOUR[126] for SCF, CCSD, CCSD(T) and CCSDT, and OpenMolcas[127] for CASSCF and CASPT2.

Appendix C

Data sets

C.1 H_2–H_2

For the first 'cheap' data set of 2 H_2 molecules configurations, CCSD with basis set 6-31G** was used, and the maximum initial kinetic energy was set to 1.2 eV, to prevent the molecules from dissociating. For each molecule, the x, y and z component of the velocity was chosen randomly from a range of 0 – 4375 m/s. 176 trajectories of each 2.000 points were calculated. Each ab initio calculation took approximately 0.4 seconds. The settings for data set 1 are shown in Table C.1.

For the more 'expensive' second data set of 2 H_2 molecules configurations, CCSD with basis set aug-cc-pVTZ was used, but instead of doing all steps of the trajectory with this, most of the trajectory steps were calculated using again the cheaper basis set 6-31G**, and only every 200 steps an expensive calculation was done. This was done because data points that are very similar to other data points are less useful. The settings for data set 2 are shown in Table C.2. Figure C.1 shows the distribution of the energy in the data set and Figure C.1 that of the intermolecular distance R.

The third data set for H_2–H_2 was created using trajectories by LAMMPS. Each trajectory was calculated with the Hinde potential combined with a custom fit for the H-H interaction. 10 configurations from each trajectory were chosen to do an ab initio calculation (basis set aug-cc-pVTZ) with. Unlike data set 1 and 2, in determining the random starting configurations the orientation was spherically uniform. For these trajectories, the molecules were also given an initial rotation, unlike the trajectories of data set 1 and 2. However, due to a bad unit conversion the rotational energy was very small. The settings for data set 3 are shown in Table C.3.

C.2 CO_2–CO_2

For the first CO_2–CO_2 data set, the collisional energy range was the same as in Lombardi [6], which is 1 to 20 kcal/mol (0.043 to 0.87 eV). The rotational temperature in Lombardi is at most 12000K, which is a rotational energy of 1.034 eV (using $E = k_B T$). This was supposed to be the maximum initial rotational energy, but due to a bad unit conversion the rotational energy is smaller than that. The rotation axis was chosen randomly, so the axis is usually not perpendicular to the primary principal axis of inertia of the molecule. The trajectories were calculated using the bondbond and zuniga-murrell-guo potential in LAMMPS. The settings for data set 1 are shown in Table C.4.

For the second CO_2–CO_2 data set, the problem with non-spherical sampling of the initial orientations was solved. The settings for data set 2 are shown in Table C.5.

Table C.1: Settings used and other information of H_2–H_2 data set 1. All points of the trajectories were used. Trajectories were of 2 hydrogen molecules bouncing around in a sphere a couple times. In determining the random starting configurations, the mistake illustrated in Figure A.3 was made; the orientations of the molecules are not spherically uniform.

	H_2–H_2 (dataset 1)
Calculated properties	Energy ground state and its numerical gradient
Nr. of trajectories	176
Total nr. of data points	351,581
Timestep	0.24 fs
Ab initio method	CCSD
Ab initio basis set	6-31G**
Initial velocity (each component)	0 – 4375 m/s
Initial R	3 – 4 Å
Limit on R	3 Å
Initial r	0.5 – 1.5 Å
E_{rot} (eV)	0 (no rotation)
Approx. computation time per data point	0.5 s

Table C.2: Settings used and other information of H_2–H_2 data set 2. Every 200^{th} time step an expensive ab initio calculation (basis set aug-cc-pVTZ) was performed. Trajectories were of 2 hydrogen molecules. In determining the random starting configurations, the mistake illustrated in Figure A.3 was made; the orientations of the molecules are not spherically uniform.

	H_2–H_2 (dataset 2)
Calculated properties	Energy ground state and its analytic gradient
Nr. of trajectories	1770
Total nr. of data points	23,917
Timestep	0.24, 0.48 or 1.2 fs
Ab initio method	CCSD
Ab initio basis set	aug-cc-pVTZ
Energy	0.0433 – 0.868 eV
Initial R	21 – 29 Å
Initial r_1 and r_2	0.5 – 1.5 (2200 data points), 0.42 – 2.1 Å (21717 data points)
Impact parameter b	0 – 20.1 Å (1100 data points), 0 Å (22817 data points)
Stop trajectory if	one or more particles were more than 15.9 Å from the origin or maximum number of steps (1,001 or 2,000 or 20,001) reached
Initial E_{rot} (eV)	0 (no initial rotation)

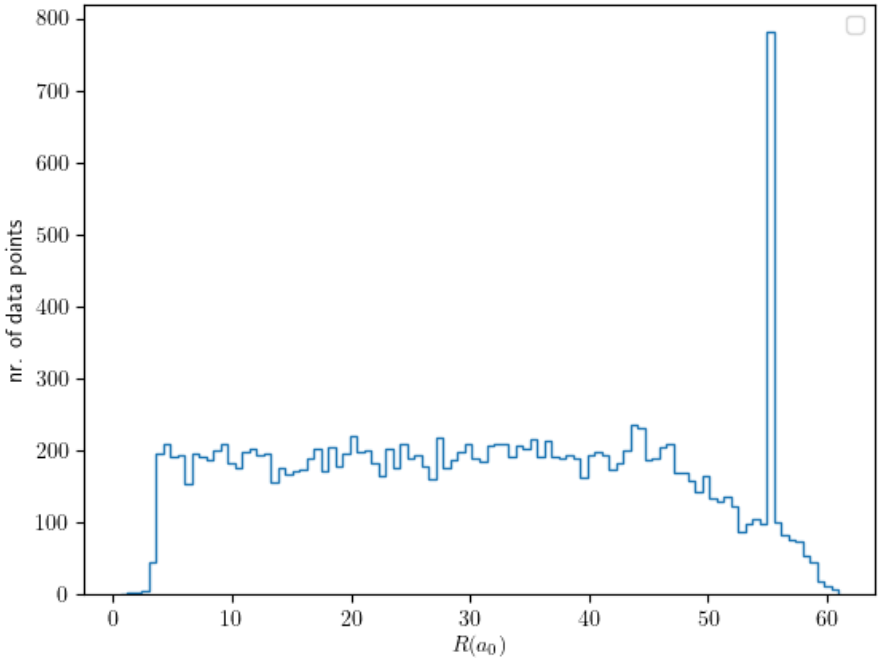


Figure C.1: Distribution of R in the second data set of H_2–H_2 data points.

Table C.3: Settings used and other information of H_2–H_2 data set 3.

	H_2–H_2 (dataset 3)
Calculated properties	Energy ground state
Nr. of trajectories	75,720
Total nr. of data points	756,335
Timestep	0.48 fs
Ab initio method	CCSD (max 50 iterations)
Ab initio basis set	aug-cc-pVTZ
Collision energy	0.043 – 0.87 eV
Initial R	10.6 Å
Initial r_1 and r_2	0.42 – 2.1 Å
Impact parameter b	0 – 1.59 Å
Stop trajectory if	one or more particles were outside the simulation box (cube with sides 13.2 Å centered at the origin) or maximum number of steps (20,001) reached
Initial E_{rot}	0.0252 – 0.509 meV
Approx. computation time per data point	11.5 s

Table C.4: Settings used and other information of CO_2–CO_2 data set 1.

	CO_2–CO_2 (dataset 1)
Calculated properties	Energy ground state
Nr. of trajectories	2564
Total nr. of data points	19445
Timestep	0.48 fs
Ab initio method	CCSD
Ab initio basis set	cc-pVTZ
Energy	0.043 – 0.87 eV
Initial R (Å)	29.1 Å
Initial r_1 (Å)	1.0 – 1.27 Å, 0.96 – 1.6 Å, 0.9 – 1.9 Å, 0.9 – 2.3 Å
Initial r_2 (Å)	1.0 – 1.27 Å, 0.96 – 1.6 Å, 0.9 – 1.9 Å, 0.9 – 2.3 Å
Initial ϕ (rad)	$0.95\pi - \pi$, $0.7\pi - 1.3\pi$, $0.625\pi - 1.375\pi$, $0.6\pi - 1.4\pi$
Impact parameter b (Å)	0 or 0 – 1.59 Å
Stop trajectory if	one or more particles were outside the simulation box (cube with sides 31.8 Å or 37.0 Åcentered at the origin) or maximum number of steps (20,001) reached
Initial E_{rot} (E_h)	0.0252 – 0.509 meV
Approx. computation time per data point	30 min.

Table C.5: Settings used and other information of CO_2–CO_2 data set 2.

	CO_2–CO_2 (dataset 2)
Calculated properties	Energy ground state
Nr. of trajectories	1710
Total nr. of data points	16880
Timestep	0.48 fs
Ab initio method	CCSD
Ab initio basis set	cc-pVTZ
Energy	0.043 – 0.87 eV
Initial R (Å)	31.8 Å
Initial r_1 (Å)	0.96 – 1.6 Å
Initial r_2 (Å)	0.96 – 1.6 Å
Initial ϕ (rad)	0.7π – 1.3π
Impact parameter b (Å)	0 – 1.59 Å
Stop trajectory if	one or more particles were outside the simulation box (cube with sides ... centered at the origin) or maximum number of steps (20,001) reached
Initial E_{rot} (E_h)	0.0252 – 0.509 meV
Approx. computation time per data point	30 min.

Appendix D

Neural network parameters

The hyperparameter search was mostly done on data set 1, with proximities type A.

Table D.1: The accuracy of the neural network for different numbers of nodes in the hidden layer.

nr. of nodes	RMSE (eV)	MAE (eV)
2	$(1.9 \pm 1.3) \times 10^{-1}$	$(1.3 \pm 1.0) \times 10^{-1}$
20	$(1.02 \pm 0.35) \times 10^{-2}$	$(6.2 \pm 1.8) \times 10^{-3}$
200	$(5.3 \pm 1.5) \times 10^{-3}$	$(2.58 \pm 0.23) \times 10^{-3}$
2000	$(5.2 \pm 1.7) \times 10^{-3}$	$(2.18 \pm 0.32) \times 10^{-3}$

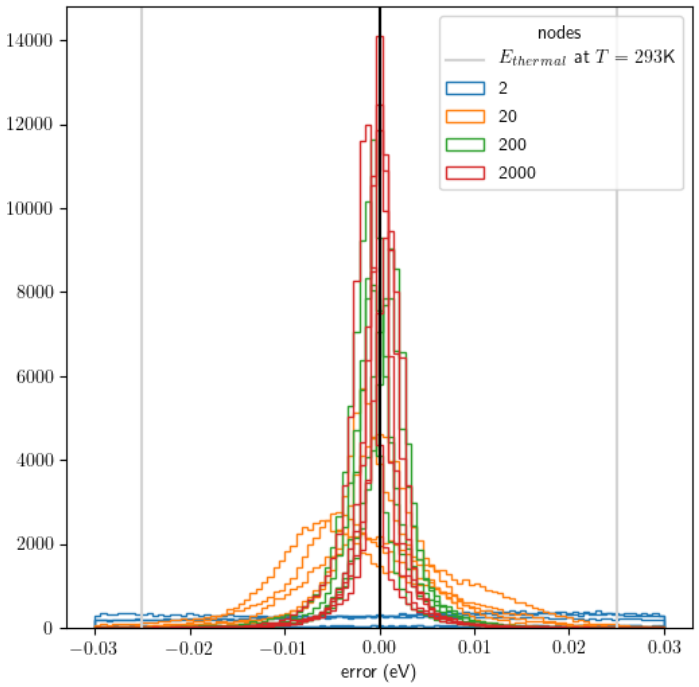


Figure D.1: Error in potential predicted by neural network for different numbers of nodes in the one hidden layer.

Table D.2: The accuracy of the neural network with and without bias parameters.

description	RMSE (eV)	MAE (eV)
no bias	$(1.02 \pm 0.35) \times 10^{-2}$	$(6.2 \pm 1.8) \times 10^{-3}$
with bias	$(6.6 \pm 1.8) \times 10^{-3}$	$(2.9 \pm 0.5) \times 10^{-3}$

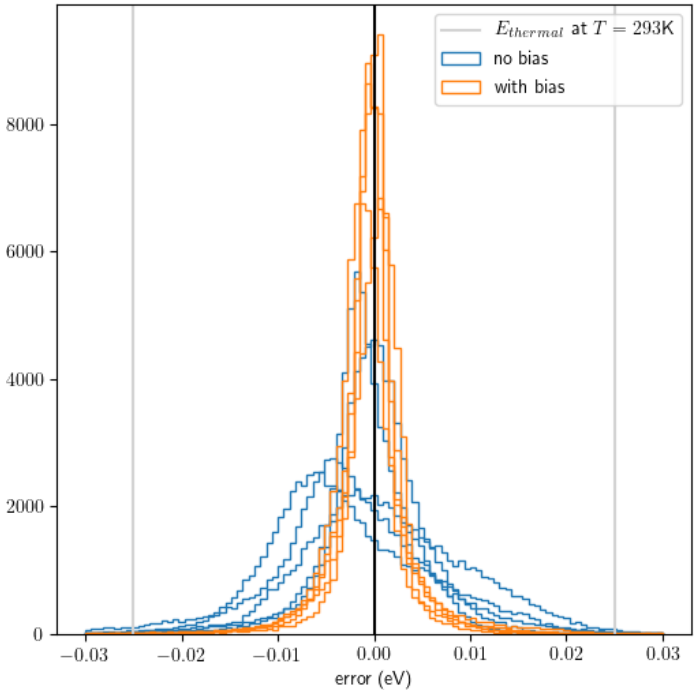


Figure D.2: Error in potential predicted by neural network with and without bias parameters.

Table D.3: The accuracy of the neural network for different activation functions.

description	RMSE (eV)	MAE (eV)
softplus	$(1.02 \pm 0.35) \times 10^{-2}$	$(6.2 \pm 1.8) \times 10^{-3}$
softsign	$(1.3 \pm 0.4) \times 10^{-2}$	$(6.8 \pm 1.0) \times 10^{-3}$

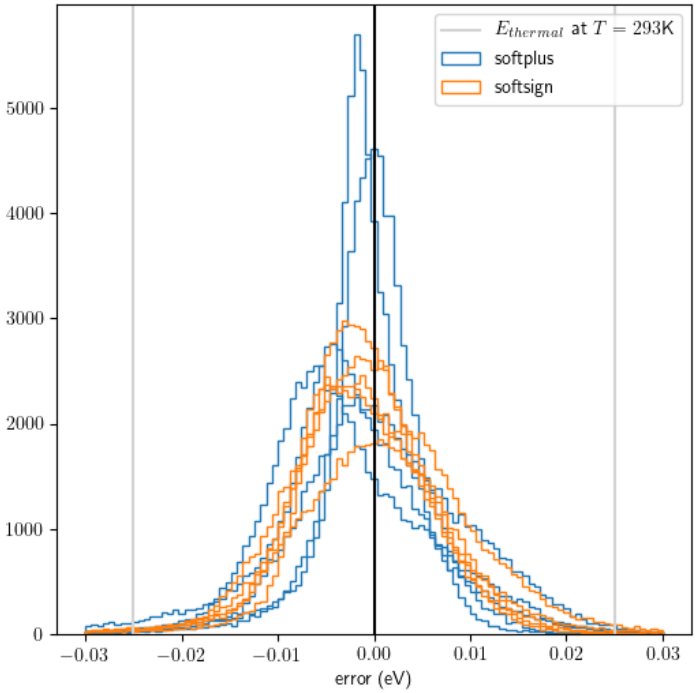


Figure D.3: Error in potential predicted by neural network for various activation functions.

Table D.4: The accuracy of the neural network for different numbers of layers.

hidden layers	RMSE (eV)	MAE (eV)
[20]	$(1.02 \pm 0.35) \times 10^{-2}$	$(6.2 \pm 1.8) \times 10^{-3}$
[20, 20]	$(5.4 \pm 1.7) \times 10^{-3}$	$(2.8 \pm 0.4) \times 10^{-3}$
[20, 20, 20]	$(4.5 \pm 1.2) \times 10^{-3}$	$(2.11 \pm 0.15) \times 10^{-3}$

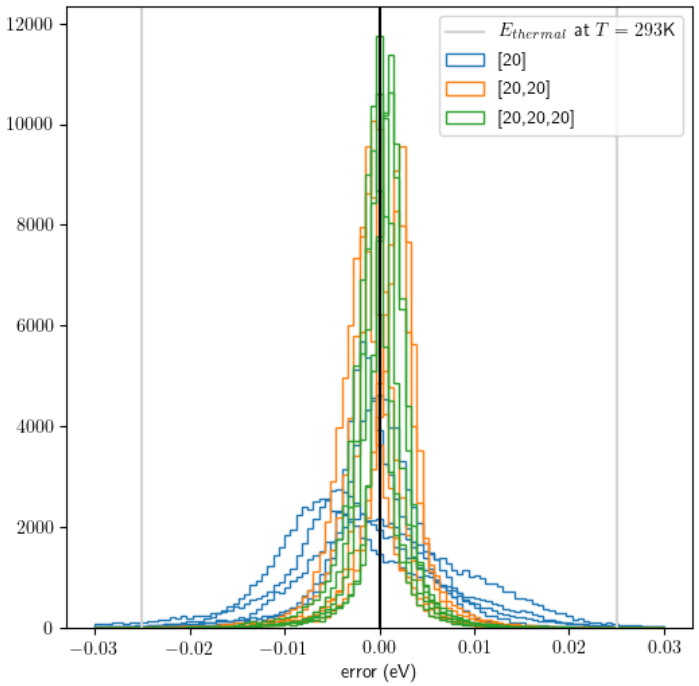


Figure D.4: Error in potential predicted by neural network for different numbers of hidden layers with each 20 nodes.

Table D.5: The accuracy of the neural network for various learning rate schedules.

nr of epochs × learning rate	RMSE (eV)	MAE (eV)
500×10^{-3}	$(6.5 \pm 1.8) \times 10^{-3}$	$(4.7 \pm 1.0) \times 10^{-3}$
250×10^{-3}, 250×10^{-4}	$(5.3 \pm 1.5) \times 10^{-3}$	$(2.58 \pm 0.23) \times 10^{-3}$
250×10^{-3}, 250×10^{-4}, 250×10^{-5}	$(5.2 \pm 1.5) \times 10^{-3}$	$(2.36 \pm 0.13) \times 10^{-3}$

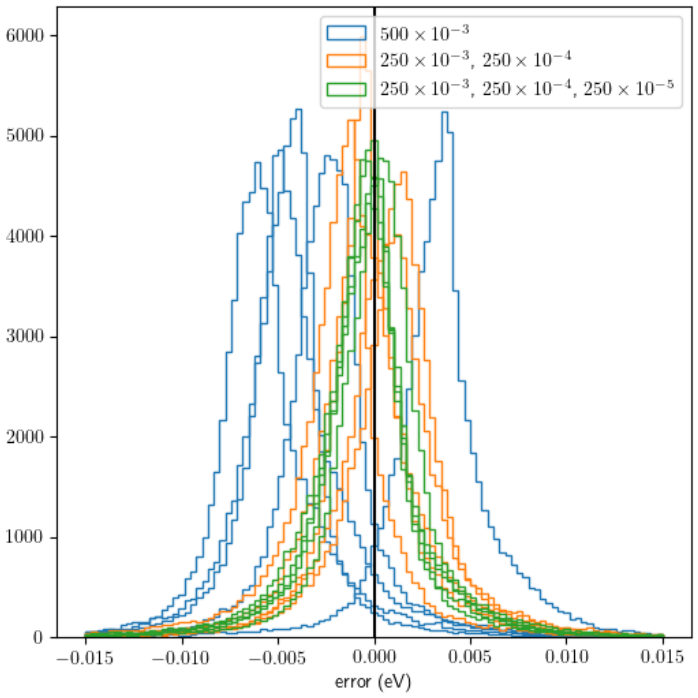


Figure D.5: Error in potential predicted by neural network for various learning rate schedules.

Table D.6: The accuracy of the neural network for different combinations of exponents.

description	RMSE (eV)	MAE (eV)
[1]	$(1.7 \pm 0.5) \times 10^{-1}$	$(1.20 \pm 0.32) \times 10^{-1}$
[2, 1], only 6 eigenvalues	$(9.0 \pm 1.1) \times 10^{-3}$	$(5.5 \pm 0.4) \times 10^{-3}$
[2, 1]	$(1.02 \pm 0.35) \times 10^{-2}$	$(6.2 \pm 1.8) \times 10^{-3}$
[6, 2, 1]	$(7.6 \pm 1.7) \times 10^{-3}$	$(4.3 \pm 0.6) \times 10^{-3}$

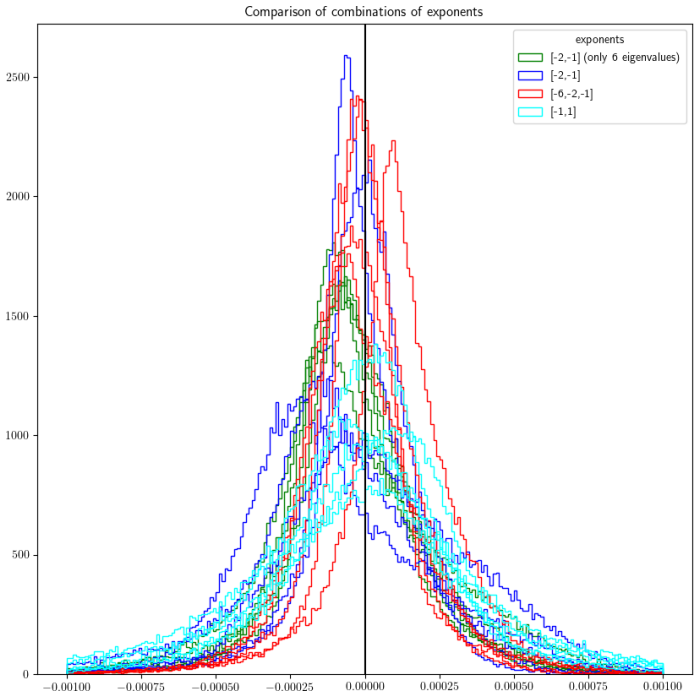


Figure D.6: Error in potential predicted by neural network for different combinations of exponents. (The exponents in the legend still have a minus sign, due to a different definition of the proximities A as r_{ij}^n at the time. Exponents in Table D.6 are correct.)

Table D.7: The accuracy of the neural network with and without scaling the input data to have variance 1. Powers=[1, 2], N1=20, 250 epochs lr=1e-3, 250 epochs lr=1e-4.

description	RMSE (eV)	MAE (eV)
no scaling	$(1.4 \pm 0.6) \times 10^{-2}$	$(8.5 \pm 1.9) \times 10^{-3}$
with scaling	$(1.02 \pm 0.35) \times 10^{-2}$	$(6.2 \pm 1.8) \times 10^{-3}$

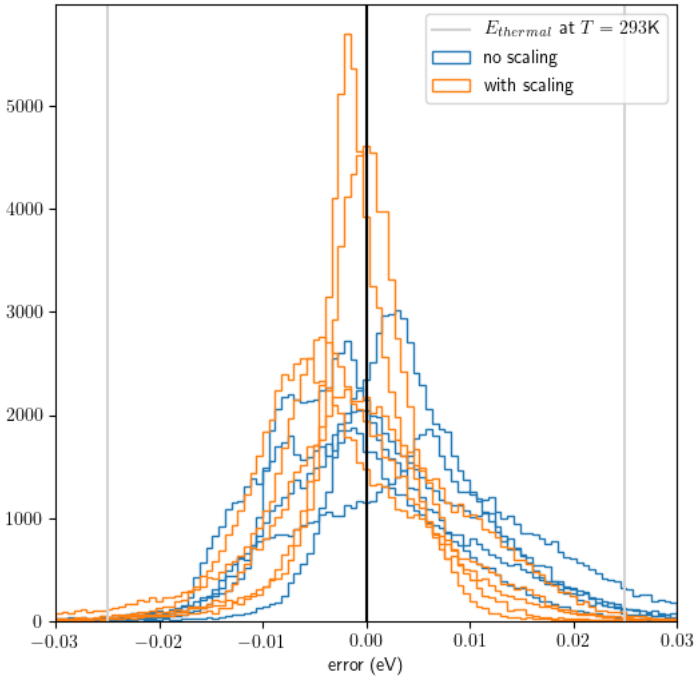


Figure D.7: Error in potential predicted by neural network with and without scaling the inputs to have a variance of 1, for exponents [1, 2].

Appendix E

Linear regression PMI results

E.1 On traces

Table E.1: Results of a linear regression on the $\mathrm{tr}\left(\mathbf{P}^k\right)$ *PMIs.*

Type of proximities	Number of coefficients	Max. k	Parameter values n or a (a in bohr)	RMSE training (meV)	RMSE validation (meV)
Proximities B	9	4	[1.5, 2., 2.5]	654.6 ± 3.2	657.5 ± 12.5
	24	4	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5.]	(6.5 ± 0.0)×10^3	(6.5 ± 0.0)×10^3
Proximities C (cutoff=10.1Å)	24	4	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5.]	62.0 ± 0.6	62.5 ± 2.4
	108	13	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5]	4.6 ± 0.1	5.7 ± 0.5
	120	13	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6.5]	7.7 ± 0.2	8.6 ± 0.5
Proximities C (cutoff=15.9Å)	9	4	[1.5, 2., 2.5]	844.4 ± 6.3	846.2 ± 24.8
	24	4	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5.]	74.0 ± 0.5	74.5 ± 2.1
	30	11	[1.5, 2., 2.5]	100.2 ± 0.9	101.9 ± 3.8
	80	11	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5.]	10.3 ± 0.1	11.1 ± 0.6
	108	13	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5]	7.9 ± 0.1	8.7 ± 0.5
	120	16	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5.]	8.2 ± 0.2	9.5 ± 0.9
	120	13	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6.]	6.8 ± 0.1	7.7 ± 0.5
	120	13	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6.5]	6.0 ± 0.1	7.2 ± 0.7

	120	13	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 6., 7.]	5.5 ± 0.1	7.1 ± 1.0
	120	11	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5 6.5 7. 7.5]	7.0 ± 0.1	7.8 ± 0.5
	120	13	[1.5, 2., 2.5, 3., 3.5, 4., 5., 6., 7., 8.]	5.0 ± 0.1	6.5 ± 1.0
	120	13	[1.5, 2., 2.5, 3., 4., 5., 6., 7., 8., 9.]	4.8 ± 0.1	5.8 ± 0.6
	120	13	[1.5, 2., 2.5, 3., 4., 5., 6., 7., 8., 10.]	4.8 ± 0.1	5.8 ± 0.6
	120	13	[1.5, 2., 2.5, 3., 4., 5., 6., 8., 10., 12.]	4.7 ± 0.1	5.6 ± 0.4
	132	13	[1.5, 2., 2.5, 3., 4., 5., 6., 8., 10., 12., 15.]	4.6 ± 0.1	5.9 ± 0.7
Morse	9	4	[0.2, 0.7, 1.5]	207.7 ± 1.0	208.5 ± 4.1
	12	4	[0.2, 0.35, 0.55, 1.5]	120.6 ± 0.6	121.6 ± 3.0
	12	4	[0.2, 0.4, 0.7, 3.]	157.5 ± 1.4	170.9 ± 22.6
	12	4	[0.2, 0.4, 0.7, 1.5]	89.9 ± 0.5	93.0 ± 3.3
	15	4	[0.2, 0.35, 0.55, 0.8, 1.5]	56.5 ± 0.8	62.4 ± 8.0
	18	7	[0.7, 1., 2.]	41.0 ± 0.5	43.8 ± 4.2
	18	7	[0.2, 0.4, 0.7]	733.5 ± 8.0	761.8 ± 42.8
	30	7	[0.2, 0.4, 0.7, 1., 2.]	27.5 ± 0.5	29.3 ± 2.4
	36	7	[0.2, 0.4, 0.7, 1., 2., 3.]	23.5 ± 0.5	25.2 ± 2.3
	120	21	[0.2, 0.4, 0.7, 1., 2., 3.]	4.3 ± 0.1	18.5 ± 17.4
	120	16	[0.2, 0.4, 0.7, 1., 2., 3., 4., 5.]	4.6 ± 0.1	5.8 ± 1.3
	120	16	[0.1, 0.2, 0.4, 0.7, 1., 2., 3., 4.]	5.1 ± 0.1	6.4 ± 0.5
	140	21	[0.1, 0.2, 0.4, 0.7, 1., 2., 3.]	4.3 ± 0.1	18.5 ± 17.4
	120	13	[0.2, 0.4, 0.7, 1., 2., 3., 4., 6., 9., 13.]	5.9 ± 0.1	7.2 ± 0.9
	120	13	[0.2, 0.4, 0.7, 1., 1.5, 2., 3., 4., 6., 9.]	5.1 ± 0.1	7.3 ± 2.0
Sine/cosine B (cutoff=15.9Å)	18	4	[0, 1, 2]	369.3 ± 3.2	371.6 ± 12.3
	60	11	[0, 1, 2]	174.1 ± 1.8	176.4 ± 6.8
	60	7	[0, 1, 2, 3, 4]	54.4 ± 0.5	56.5 ± 2.3
	120	21	[0, 1, 2]	$(5.7 \pm 0.0)\times10^3$	$(1.3 \pm 1.3)\times10^4$
	120	11	[0, 1, 2, 3, 4, 5]	73.7 ± 0.8	81.7 ± 10.4
	112	8	[0, 1, 2, 3, 4, 5, 6, 7]	16.0 ± 0.3	18.9 ± 3.0

	120	7	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	10.4 ± 0.2	11.9 ± 1.2

E.2 On products of traces

Table E.2: Results of a linear regression on products of $\mathrm{tr}(\mathbf{P}^2)$, $\mathrm{tr}(\mathbf{P}^3)$ *and* $\mathrm{tr}(\mathbf{P}^4)$, *up to a certain order.*

Type of proximities	Number of coefficients	Order	Parameter values n or a (a in bohr)	RMSE training (meV)	RMSE validation (meV)
Proximities B	8	3	[1.5, 2., 2.5, 3.]	292.9 ± 1.1	294.4 ± 4.8
	20	3	[1.5, 2., 2.5, 3., 4., 5., 6., 8., 10., 12.]	(7.0 ± 0.0)×10^3	(7.0 ± 0.1)×10^3
	88	10	[1.5, 2., 2.5, 3.]	(7.0 ± 0.0)×10^3	(7.3 ± 0.7)×10^3
Proximities C (cutoff=10.1Å)	20	3	[1.5, 2., 2.5, 3., 4., 5., 6., 8., 10., 12.]	78.0 ± 0.8	79.8 ± 3.8
	110	7	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 7., 7.5]	15.5 ± 0.3	18.4 ± 0.7
	126	8	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5]	8.8 ± 0.2	10.4 ± 0.9
	130	11	[1.5, 2., 2.5, 3., 3.5]	6.2 ± 0.1	7.4 ± 0.5
	140	8	[1.5, 2., 2.5, 3., 4., 5., 6., 8., 10., 12.]	8.6 ± 0.2	10.6 ± 1.4
	182	11	[1.5, 2., 2.5, 3., 3.5, 4., 4.5]	4.5 ± 0.1	6.9 ± 0.7
	198	10	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5]	4.6 ± 0.1	6.4 ± 0.9
	198	12	[1.5, 2., 3.]	111.9 ± 1.8	158.9 ± 37.8
Morse	50	7	[0.2, 0.35, 0.55, 0.8, 1.5]	24.0 ± 0.4	32.0 ± 10.6
	90	7	[0.2, 0.35, 0.55, 0.8, 1.5, 2., 3., 4., 5.]	16.2 ± 0.3	22.5 ± 7.5
	126	8	[0.2, 0.35, 0.55, 0.8, 1.5, 2., 3., 4., 5.]	9.4 ± 0.3	11.9 ± 2.6
	119	9	[0.2, 0.35, 0.55, 0.8, 1.5, 2., 3.]	9.8 ± 0.2	14.7 ± 2.5
Sine/cosine	80	6	[0, 1, 2, 3, 4]	201.2 ± 1.8	213.1 ± 7.7
	140	8	[0, 1, 2, 3, 4]	127.8 ± 2.0	146.6 ± 10.6
	260	11	[0, 1, 2, 3, 4]	57.3 ± 0.3	85.3 ± 13.5
	264	12	[0, 1, 2, 3]	45.4 ± 0.5	72.6 ± 15.3

E.3 On traces and trace complements

Table E.3: Results of a linear regression on the $\mathrm{tr}\left(\mathbf{P}^k\right)$ *PMIs.*

Type of proximities	Number of coefficients	Max. k	Parameter values n or a (a in bohr)	RMSE training (meV)	RMSE validation (meV)
Proximities C (cutoff=10.1Å)	75	13	[2, 3, 5]	6.2 ± 0.2	(8.0 ± 16.0)×10^6
	91	7	[1.5, 2., 2.5, 3., 3.5, 4., 4.5]	10.4 ± 0.4	(9.6 ± 19.2)×10^3
	100	13	[2, 3, 5, 8]	3.3 ± 0.1	5.7 ± 2.0
	105	11	[2, 3, 5, 8, 12]	4.0 ± 0.2	(1.0 ± 2.1)×10^5
	105	11	[2, 3, 5, 8, 12]	4.0 ± 0.2	(1.0 ± 2.1)×10^5
	105	11	[2, 3, 4, 8, 12]	4.2 ± 0.1	(3.8 ± 7.6)×10^4
	105	11	[1.5, 2., 2.5, 3., 3.5]	6.5 ± 0.1	7.8 ± 0.4
	115	12	[2, 3, 5, 8, 12]	3.5 ± 0.1	(1.8 ± 3.6)×10^5
	125	13	[2, 3, 5, 8, 12]	2.8 ± 0.1	3.6 ± 0.4
	126	11	[2, 3, 4, 6, 9, 12]	3.7 ± 0.1	8.1 ± 5.5
	126	11	[1.5, 2., 2.5, 3., 3.5, 4.]	4.8 ± 0.1	5.7 ± 0.6
	130	7	[1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6.5]	8.3 ± 0.1	10.2 ± 1.6
	130	7	[1.5, 2., 2.5, 3., 4., 5., 6., 8., 10., 12.]	7.4 ± 0.3	(7.4 ± 14.7)×10^3
Morse	120	8	[0.2, 0.4, 0.7, 1., 2., 3., 4., 5.]	6.8 ± 0.1	752.9 ± 1488.2
	124	16	[0.2, 0.4, 0.7, 1.]	24.2 ± 0.2	(4.3 ± 8.5)×10^4
	125	13	[0.7, 2., 3., 4., 5.]	3.4 ± 0.2	(1.2 ± 2.4)×10^5
	135	14	[0.7, 2., 3., 4., 5.]	2.7 ± 0.1	(4.6 ± 9.2)×10^4
	155	16	[0.2, 0.35, 0.55, 0.8, 1.5]	3.8 ± 0.0	17.5 ± 14.7
	155	16	[0.7, 2., 3., 4., 5.]	2.2 ± 0.1	3.6 ± 1.7
	248	16	[0.2, 0.4, 0.7, 1., 2., 3., 4., 5.]	1.9 ± 0.1	(5.4 ± 10.7)×10^7

Appendix F

Derivation analytical gradient

To get the analytical gradient of the neural network potential the entire chain from Cartesian coordinates $x_{ii'}$ (where $x_{ii'}$ is the i'^{th} coordinate of particle i) to the predicted energy E needs to be differentiated. The element $p_{jj'}$ of the proximity matrix $\mathbf{P}(n)$ of type A with the exponent n and the weight factor $\alpha_{jj'}$ is

$$p_{jj'}(n) = \alpha_{jj'} 1/r_{jj'}^n, \tag{F.1}$$

or, if the potential is forced to zero for $r_{jj'} > r_0$ (proximity matrix type B):

$$p_{jj'}(n) = \begin{cases} \alpha_{jj'} \left(\frac{r_0}{r_{jj'}} - \frac{r_{jj'}}{r_0} \right)^n, & \text{if } r_{jj'} < r_0, \\ 0 & \text{if } r_{jj'} > 0, \end{cases} \tag{F.2}$$

in both cases $r_{jj'}$ are the interatomic distances:

$$r_{jj'} = ||\boldsymbol{x}_j - \boldsymbol{x}_{j'}|| = \left(\sqrt{\sum_{h=1}^{3} \left(x_{jh} - x_{j'h} \right)^2} \right), \tag{F.3}$$

which means the diagonal $p_{j=j'}$ is always zero.

Using the fact that

$$\frac{\partial x_{jh}}{\partial x_{ii'}} = \delta_{ji} \delta_{hi'}, \tag{F.4}$$

the derivative of $r_{jj'}$ with respect to a coordinate $x_{ii'}$ for proximity matrix type A is then

$$\frac{\partial p_{jj'}}{\partial x_{ii'}} = -\alpha_{jj'} n (r_{jj'})^{-n-2} (x_{ji'} - x_{j'i'}) \begin{cases} 1 & \text{if } i = j, \\ -1 & \text{if } i = j', \\ 0 & \text{else} \end{cases} \tag{F.5}$$

and for proximity matrix type B

$$\frac{\partial p_{jj'}}{\partial x_{ii'}} = \begin{cases} \alpha_{jj'} n \left(\frac{r_0}{r_{jj'}} - \frac{r_{jj'}}{r_0} \right)^{n-1} \left(-\frac{r_0}{r_{jj'}^2} - \frac{1}{r_0} \right) \frac{x_{ji'} - x_{j'i'}}{r_{jj'}} \begin{cases} 1 & \text{if } i = j, \\ -1 & \text{if } i = j', \\ 0 & \text{else} \end{cases} & \text{if } r < r_0, \\ 0 & \text{if } r > r_0, \end{cases} \tag{F.6}$$

with in both cases the diagonal $\frac{\partial p_{j=j'}}{\partial x_{ii'}}$ also always zero.

ϵ_k, the k^{th} eigenvalue of the matrix $\mathbf{P}(n)$ with elements $p_{jj'}$ is scaled to $e_k = \frac{\epsilon_k - \text{mean}_k}{\sqrt{\text{var}_k}}$, with mean_k the mean of this eigenvalue over the entire training set, and var_k the variance.

The derivative of the eigenvalues with respect to an element of the matrix is [142]:

$$\frac{\partial \epsilon_k}{\partial p_{jj'}} = (v_k \cdot w_j)(v_k \cdot w_{j'}) = v_{kj} v_{kj'}, \tag{F.7}$$

where w_i is the Euclidean basis vector with only a 1 at i and zero everywhere else. with v_{ki} the i^{th} element of the eigenvector v_k of unit length corresponding to eigenvalue ϵ_k. The derivative of the scaled eigenvalue e_k is then

$$\frac{\partial e_k}{\partial p_{jj'}} = \frac{1}{\sqrt{\text{var}_k}} \frac{\partial \epsilon_k}{\partial p_{jj'}} = \frac{v_{kj} v_{kj'}}{\sqrt{\text{var}_k}}. \tag{F.8}$$

The eigenvalues resulting from multiple proximity matrices with different exponents are then combined into a vector $e_{k'}$.

The result of a neural network with L layers and activation function $\phi(x)$ for each neuron is

$$E_{\text{scaled}} = \mathbf{b}^L + \mathbf{A}^L(\phi^L(\mathbf{b}^{L-1}) + \mathbf{A}^{L-1}(...\phi^1(\mathbf{b}^1 + \mathbf{A}^1\phi^0(\mathbf{b}^0 + \mathbf{A}^0 e_{k'})))). \tag{F.9}$$

with the matrix $\mathbf{A}^\ell$ the ℓ^{th} weights connecting the ℓ $^{\text{th}}$ hidden layer (or the input layer for $\ell = 0$) to the $(\ell + 1)$ $^{\text{th}}$ layer (or to the output for $\ell = L$) and $\mathbf{b}^\ell$ the bias parameters for layer ℓ. The derivative of the neural network result E_{scaled} with respect to the vector of eigenvalues $\boldsymbol{e}$ is then (do this efficiently with backpropagation [143]):

$$\frac{\partial E_{\text{scaled}}}{\partial e_{k'}} = \mathbf{A}^L \cdot \phi'(\mathbf{a}_{L-1})\mathbf{A}^{L-1} \cdot \phi'(\mathbf{a}_{L-2})...\mathbf{A}^1 \cdot \phi'(\mathbf{a}_0)\mathbf{A}^0_{:,k'}, \tag{F.10}$$

Here $\mathbf{a}_\ell$ is the activation before applying the activation function (so the intermediate result after applying the matrix $\mathbf{A}^\ell$ and parameters $\mathbf{b}_\ell$; these are already computed during the forward pass, so save them) of the neurons in layer ℓ. Here $\cdot$ stands for matrix multiplication, the other multiplications (for example $\phi'(\mathbf{a}_0)\mathbf{A}^0_{:,k'}$) are element-wise. $\mathbf{A}^0_{:,k'}$ is the k'^{th} column of the matrix $\mathbf{A}^0$. For the softplus activation function $\phi(x) = \ln(1 + e^x)$ (applied element-wise) the derivative is $\phi'(x) = \frac{1}{1+e^{-x}}$.

The actual energy predicted by the net is then $E = E_{\text{scaled}}\sqrt{\text{var}_E} + \text{mean}_E$, which makes

$$\frac{\partial E}{\partial e_{k'}} = \sqrt{\text{var}_E} \frac{\partial E_{\text{scaled}}}{\partial e_{k'}}. \tag{F.11}$$

Here var_E is the variance of E in the training set and mean_E the mean.

Putting this all together, this is the result:

$$\frac{\partial E}{\partial x_{ii'}} = \sum_{k'} \frac{\partial E}{\partial e_{k'}} \left(\sum_{j,j'} \frac{\partial e_{k'}}{\partial p_{jj'}} \frac{\partial p_{jj'}}{\partial x_{ii'}} \right). \tag{F.12}$$

This can be implemented as matrix multiplications.

List of Figures

List of Tables

Glossary

closed walk A sequence of edges in a graph that connects a sequence of nodes, where the first and last node in the sequence are the same. 47, 73

CPC Characteristic Polynomial Coefficient. 83, 149

fold One iteration during crossvalidation. 59, 60

NN Neural Network. 34, 60, 62, 63, 65, 88, 104, 148

open walk A sequence of edges in a graph that connects a sequence of nodes, where the first and last node in the sequence are not the same. 48, 73

PES Potential Energy Surface. 63, 84, 105

PIP Permutationally Invariant Polynomials as described by Braams, Bowman and Xie [144, 25]. 40, 47, 53, 57, 60, 62, 63, 73, 80, 83, 104

PME Proximity Matrix Eigenspectrum. 43, 48, 54, 59, 60, 62, 63, 65, 88, 104, 148

PMI Proximity Matrix Invariant, in this report they are $\mathrm{tr}\left(\mathbf{P}^k\right)$ and $\mathrm{tc}(\mathbf{P}^k)$. 47, 48, 54, 60, 62, 63, 73, 78, 104, 141, 144, 152

QCT Quasiclassical Trajectories. 84, 93

RMSE Root-mean-square error. 24, 63, 74, 80, 95, 135, 141, 143, 144, 151

trace complement The sum of the off-diagonal elements of a matrix. 48, 78