Potential Energy Surfaces: From Force Fields to Neural Networks

Inauguraldissertation

zur

Erlangung der Würde eines Doktors der Philosophie

vorgelegt der

Philosophisch-Naturwissenschaftlichen Fakultät

der Universität Basel

von

Oliver Thorsten Unke

aus Deutschland

Basel, 2019

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät

auf Antrag von:
Prof. Dr. Markus Meuwly
Prof. Dr. Klaus-Robert Müller

Basel, den 23. April 2019

Prof. Dr. Martin Spiess (Dekan)

The road to wisdom? — Well, it's plain
and simple to express:
Err
and err
and err again
but less
and less
and less.

– Piet Hein

Abstract

Oliver Thorsten Unke

Potential Energy Surfaces:
From Force Fields to Neural Networks

Almost a century ago, Paul A. M. Dirac remarked that the Schrödinger equation (SE) contains all that is necessary to describe chemical phenomena. Unfortunately, solving the SE, even approximately, remains a time-intensive task and is possible only for systems containing a few atoms. For this reason, potential energy surfaces (PESs) are used to circumvent the solution of the SE altogether: They estimate the energy of a chemical system by evaluating an analytical function. For example, so-called force fields (FFs) model chemical bonds as "springs", i.e. with harmonic potentials. While this is computationally efficient, it limits the accuracy of FFs. A promising alternative are machine learning (ML) methods, such as kernel ridge regression (KRR) and artificial neural networks (NNs), which allow the construction of a PES without assuming a functional form.

In the first part of this thesis, FFs are described in more detail and the minimal distributed charge model (MDCM) is introduced as a way to increase their accuracy. Applications to several challenging molecules are used to demonstrate the utility of this method. In the second part, KRR is reviewed and ways to improve its computational efficiency for PES construction are discussed. Further, the reproducing kernel Hilbert space (RKHS) toolkit is introduced, which largely automates the construction of efficient and accurate PESs for small systems. In the last part, a brief overview of NNs and their historic development is given. The use of NNs to construct PESs is explored and two alternatives are described in more detail. Both variants are applied to various benchmark datasets in order to demonstrate their versatility and accuracy.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Prof. Dr. Markus Meuwly. The values he taught me by way of example, both socially and academically, will continue to provide me with guidance in my future career and life.

Further, I would like to thank Prof. Dr. Klaus-Robert Müller, who kindly accepted to act as external examiner.

I would also like to express my appreciation for all current and past members of the Meuwly group. Apart from the help they provided, I always enjoyed the small breaks from work we spent together, in particular our insightful and funny discussions at lunch-time.

A special thanks goes to my friend Dmitry Nezlav and colleagues Dr. Jasmine Desmond and Ali Hisham Taher A Razzak for proofreading my thesis and providing helpful suggestions.

Not to be forgotten, I am thankful for my teachers in Gymnasium, who not only provided the foundation to all of my knowledge, but also taught me to be curious. In particular, I would like to thank Frank Braun (Biology), Martin Löw (Physics), Klaus Krepp (Mathematics), and Peter Grinzinger (Chemistry).

My deepest gratitude goes to my parents Christel and Thomas and my brothers Ulf, Thiemo, Aldo, Bautz, and Troll, for their unconditional support.

Finally and above all, this thesis is dedicated to my wife Sarah. Your love and support accompanied me through all difficulties I have faced along the way and you gift me happiness whenever I am sad.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AIM	**A**toms **I**n **M**olecules
ASE	**A**tomic **S**imulation **E**nvironment
BO	**B**orn-**O**ppenheimer
CNN	**C**onvolutional **N**eural **N**etwork
DCM	**D**istributed **C**harge **M**odel
DE	**D**ifferential **E**volution
DFT	**D**ensity **F**unctional **T**heory
ESP	**E**lectro**s**tatic **P**otential
FF	**F**orce **F**ield
GDML	**G**radient-**D**omain **M**achine **L**earning
HDNN	**H**igh-**D**imensional **N**eural **N**etwork
KRR	**K**ernel **R**idge **R**egression
LJ	**L**ennard-**J**ones
LTU	**L**inear **T**hreshold **U**nit
MAE	**M**ean **A**bsolute **E**rror
MD	**M**olecular **D**ynamics
MDCM	**M**inimal **D**istributed **C**harge **M**odel
MEP	**M**inimum **E**nergy **P**ath
ML	**M**achine **L**earning
MLP	**M**ulti**l**ayer **P**erceptron
MMPT	**M**olecular **M**echanics with **P**roton **T**ransfer
MSE	**M**ean **S**quared **E**rror
MTP	**M**ul**tip**ole
NN	**N**eural **N**etwork
PC	**P**oint **C**harge
PCM	**P**oint **C**harge **M**odel
PES	**P**otential **E**nergy **S**urface
QM/MM	**Q**uantum **M**echanics/**M**olecular **M**echanics
RBF	**R**adial **B**asis **F**unction
RKHS	**R**eproducing **K**ernel **H**ilbert **S**pace
RMSE	**R**oot **M**ean **S**quared **E**rror
RNN	**R**ecurrent **N**eural **N**etwork
SE	**S**chrödinger **E**quation
SEpA	**S**quared **E**rror **p**er **A**tom
TI	**T**hermodynamic **I**ntegration

Chapter 1

Introduction

"The general theory of quantum mechanics is now almost complete, the imperfections that still remain being in connection with the exact fitting in of the theory with relativity ideas. These give rise to difficulties only when high-speed particles are involved, and are therefore of no importance in the consideration of atomic and molecular structure and ordinary chemical reactions [...]. The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation."

– Paul A. M. Dirac, 1929 [1]

It is now almost a century ago when Dirac noted[1] that the Schrödinger equation (SE)[2] contains all that is necessary to describe chemical phenomena and processes. Unfortunately, even on modern computers, the SE is too complicated to be solved in closed form but for the simplest systems. To still be able to obtain information about a chemical system, computational and numerical methods have been devised to find approximate solutions.

A widespread approach to simplify the solution of the SE is the Born-Oppenheimer (BO) approximation.[3] It assumes that electronic and nuclear motion are decoupled, neglecting the latter based on the observation that atomic nuclei are more than three orders of magnitude heavier than electrons. The nuclear positions enter the resulting electronic SE only parametrically: From the perspective of the much slower moving nuclei, electrons always remain at their ground state energy and adjust instantaneously to changes in the nuclear positions. Because of this, the electronic energy depends only on the external potential caused by the nuclei, which in turn is fully determined by their positions and nuclear charges.[a]

[a] While some cases are known where the BO approximation breaks down,[4–6] assuming adiabaticity is valid for the vast majority of chemical systems and has been the standard ansatz to describe the interaction between electrons and nuclei since the early days of quantum mechanics.[7]

As a consequence, the BO approximation gives rise to the concept of a potential energy surface (PES): A $(3N-6)$-dimensional[a] hypersurface that returns the energy of a chemical system given the positions (and nuclear charges) of its N nuclei. The dynamics of a molecule are governed by its PES, similar to the way the motion of a ball rolling through a hilly landscape is determined by the topology of the surface it is rolling on. The low energy "valleys" on the PES correspond to stable molecules and a reaction corresponds to the system moving from one valley to another, usually across a "ridge" with higher energy, also known as transition state (see Figure 1.1). Based on this insight, classical molecular dynamics (MD) simulations,[8] where nuclei are assumed to follow Newton's laws of motion[9] according to the forces derived from the PES,[b] have become a standard tool to study reactions and properties of chemical and biological systems.[10]

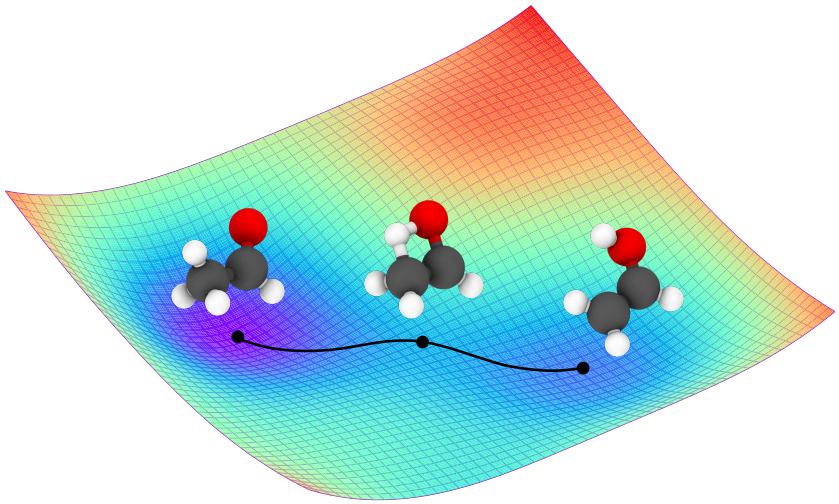


FIGURE 1.1: Two-dimensional illustration of the PES of C_2H_4O (the true PES for this system is 15-dimensional). Regions with low potential energy are drawn in blue and high energy regions in red. The reaction from ethanal (left) to ethenol (right) is depicted.

To extract statistically significant information from MD simulations, many independent (or few very long)[c] trajectories need to be computed, each requiring thousands of energy and force evaluations. As even approximate *ab initio* solutions to the SE are, depending on system size, computationally very demanding or downright

[a] Each of the N nuclei can move freely in three-dimensional space, which results in a total of $3N$ degrees of freedom. However, since translating (in x-, y- or z-direction) or rotating (around the x-, y- or z-axis) the system as a whole does not change the internal energy, six degrees of freedom need to be subtracted.

[b] The forces acting on the nuclei are equal to the negative gradient of the PES with respect to the nuclear coordinates.

[c] According to the ergodic hypothesis,[11] the average of a process over time and the average over a statistical ensemble are equivalent.

impossible, directly solving the SE for the purpose of performing MD simulations is often infeasible. For this reason, the PES is usually approximated by an analytical function, circumventing the problem of solving the SE altogether. The difficulty is then shifted to finding an appropriate function for representing the PES.

A popular approach to construct PESs are force fields (FFs),[12] which fit the parameters of an empirical functional form either to best reproduce *ab initio* reference values, experimental data, such as thermodynamic or spectroscopic observables, or a combination of both. Conventional FFs[13–18] decompose the energy of a system into the sum of bonded terms modelled by harmonic potentials ("springs") and nonbonded terms consisting of the electrostatic interaction of point charges and van der Waals forces, usually represented as a Lennard-Jones (LJ) potential.[19] Polarizable FFs augment this description by explicitly incorporating polarization effects, for example in the form of fluctuating charges[20,21] or polarizable sites.[22,23] Other functional forms, e.g. based on the concept of bond orders,[24–27] also exist but are less common. Due to their simplicity, FFs are computationally efficient and can be applied to large systems, such as proteins or condensed matter.

For the generation of a PES for small systems consisting of just a few atoms, it is often preferable to directly interpolate a set of reference energies, obtained from a pointwise solution of the SE, to construct a continuous functional form. Popular interpolation techniques include the modified Shepard algorithm,[28–30] the moving least squares method,[31–33] permutational invariant polynomials,[34–36] and kernel ridge regression (KRR).[37–39]

In general, an ideal PES should have the following properties (or at least most of them):[40]

- *Accuracy.* The PES should closely approximate *ab initio* solutions to the SE and faithfully reproduce important chemical trends. While a PES without this property could still be used in MD simulations, the resulting dynamics would deliver an erroneous impression of the relevant chemical processes.

- *Efficiency.* The whole purpose of a PES is to circumvent the need for a solution of the SE for increased computational efficiency. As such, an evaluation of the PES should be orders of magnitude faster than solving the SE directly. In particular, the computational complexity of evaluating the PES should scale favourably (linearly or linearithmically) with respect to system size.[a]

- *Reactivity.* The making and breaking of bonds lies at the heart of chemistry. Therefore, the PES should not require any preconceived bonding patterns or

[a] The time complexity of *ab initio* methods to solve the SE is, depending on their accuracy, typically between $O(N^3)$ and $O(N^{10})$,[41] where N is a measure of system size. This unfavourable scaling prevents *ab initio* methods from being applied to large systems such as proteins.

atom types[a] as input, which would make the description of reactions difficult or impossible. The PES should be able to describe all types of systems and bonding situations in an unbiased fashion.

- *Transferability.* The PES should be as predictive as possible, meaning it should be reliable for atomic configurations that were not used for its parametrization. Ideally, it should allow accurate predictions across chemical space for molecules it was not parametrized for.

Most current PESs fulfil only some of the above criteria. For example, while direct interpolation of reference energies (e.g. with KRR) allows accurate predictions and MD simulations at low computational cost, PESs constructed in this way are not transferable to other systems. As such, a new PES needs to be constructed for every system of interest, which is often a laborious and time-consuming process. Additionally, the curse of dimensionality[b] prevents the application of interpolation techniques to large systems altogether.

FFs on the other hand allow transferable predictions across many different systems, usually only restricted to a certain class of compounds (e.g. proteins or carbohydrates). Their functional form is motivated by physical and chemical knowledge, which ensures that bonded interactions between atoms become repulsive at short distances, or electrostatic and dispersion interactions follow the correct asymptotic behaviour. While this fixed functional form makes FFs computationally efficient and allows the simulation of systems containing thousands of atoms, it ultimately limits their accuracy. Additionally, FFs require preconceived bonding patterns and corresponding atom types as input and are thus unable to describe reactions, where the bonding structure and atom types must change. Although remedies for these drawbacks of FFs exist, they usually have a negative impact on some of the desirable properties of FFs. For example, improving the description of electrostatic interactions by introducing polarizability or multipole (MTP) models increases their computational cost.[23] Similarly, while methods to construct reactive FFs for specific systems[42] or processes[43] have become available, they need to be reparametrized for different systems, sacrificing the transferability of FFs, or they lack sufficient accuracy for many applications.[27]

The focus of this thesis is the development of new approaches and improvement of existing methods to construct PESs for both small and large systems.

In chapter 2, conventional FFs are described in detail and an alternative method for describing electrostatic interactions, the minimal distributed charge model

[a] Atom types refer to different variants of atoms for the same element, e.g. a distinction between aliphatic and aromatic carbon atoms.

[b] The number of possible combinations of internal coordinates grows exponentially with the dimensionality of the system, which quickly renders the possibility of performing *ab initio* calculations for all of them intractable.

(MDCM), is introduced. It allows the description of the electrostatic potential (ESP) of a molecule at multipolar quality, while also keeping its computational cost comparable to that of the atom-centred point charge model (PCM) standard in many conventional FFs. The utility of MDCMs is demonstrated by applying the method to several challenging molecules and evaluating its performance.

Chapter 3 reviews KRR and discusses possible ways to make this method computationally more efficient. The reproducing kernel Hilbert space (RKHS) toolkit is introduced, which largely automates the process of generating efficient PESs for small systems. Since PESs obtained by interpolation generally are not transferable (a new PES needs to be constructed for every system of interest), an automatic and fast approach to perform this task with as little human effort as possible is desirable. Given appropriate reference data, the RKHS toolkit allows constructing PESs for small systems in a few minutes.

In chapter 4, a brief overview of the history of artificial neural networks (NNs) how they can be used to construct PESs is given. Two different variants of NN-based PES are introduced and applied to various quantum chemical datasets in order to demonstrate their accuracy and versatility. Provided with sufficient reference data and utilized properly, NNs promise to fulfil many, if not all, of the properties of an ideal PES.

Finally, in chapter 5, the results of the thesis are summarized and possible future research directions are outlined.

Chapter 2

Minimal distributed charge model

In the introduction to this chapter (section 2.1), a detailed description of empirical force fields (FFs) is given. Limitations of their functional form are pointed out and a possible method to improve the description of electrostatic interactions in FFs, the minimal distributed charge model (MDCM), is introduced. Section 2.2 details how MDCMs are constructed, followed by application of the method to various molecules with challenging electrostatic potentials (ESPs) (parameters for all presented models can be found in appendix A). In section 2.3, MDCMs are applied to compute electrostatic interaction energies of water dimers and to reproduce the solvation free energy of bromobenzene. The results are summarized and possible future improvements to the MDCM method are outlined in section 2.4. The results presented in this chapter have been previously published in [44].

2.1 Introduction

In general, conventional force fields (FFs) model the potential energy surface (PES) of a chemical system as a sum over bonded and nonbonded terms[12]

$$E_{\text{pot}} = E_{\text{bonded}} + E_{\text{nonbonded}} \tag{2.1}$$

and the bonded terms E_{bonded} are further decomposed into four different contributions (Eq. 2.2, see Figure 2.1).

$$E_{\text{bonded}} = E_{\text{bonds}} + E_{\text{angles}} + E_{\text{dihedrals}} + E_{\text{impropers}} \tag{2.2}$$

The bonding energy term E_{bonds} is usually modelled by a sum of harmonic potentials running over a list of all atom pairs considered to be bonded (Eq. 2.3).[12]

$$E_{\text{bonds}} = \sum_{\text{bonds}} \frac{k_{\text{bon}}}{2}(r - r_{\text{eq}})^2 \tag{2.3}$$

Here, r is the distance between two bonded atoms a and b (see Figure 2.1A) and the bond strength k_{bon} and equilibrium bond length r_{eq} are parameters specific to the

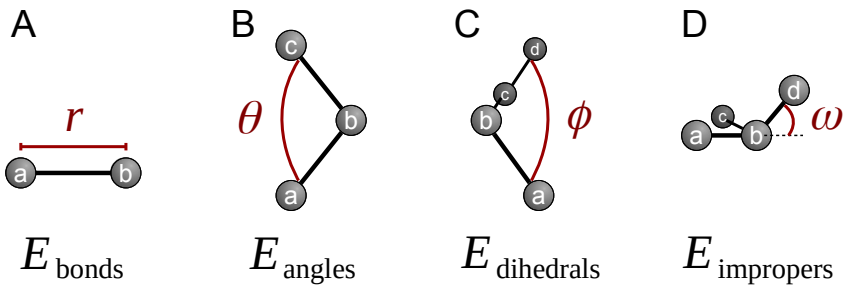


FIGURE 2.1: Overview of different contributions to the bonded energy E_{bonded} (Eq. 2.2) commonly used in FFs. **A**: The bonding energy E_{bonds} (Eq. 2.3) is associated with the distance r between two bonded atoms a and b. **B**: The bending energy E_{angles} (Eq. 2.4) is associated with the bond angle θ spanned by three atoms a, b, and c. **C**: The torsional energy $E_{\text{dihedrals}}$ (Eq. 2.6) is associated with the dihedral angle ϕ between the planes defined by atoms a, b, and c and atoms b, c, and d. **D**: The improper torsional energy $E_{\text{impropers}}$ (Eq. 2.7) is associated with the improper angle ω describing the out-of-plane motion of an atom d from the plane defined by atoms a, b, and c.

combination of atom types[a] involved in the bond. Alternative functional forms for E_{bonds} (e.g. the Morse potential[45]) are used in some FFs but are less common.[12]

The bending energy E_{angles} is associated with the angle between two bonds sharing the same atom, e.g. the angle θ between the bond of atoms a and b and atoms b and c (see Figure 2.1B). It is usually modelled by a sum of harmonic potentials running over all atom triplets describing such an angle (Eq. 2.4).[12]

$$E_{\text{angles}} = \sum_{\text{angles}} \frac{k_{\text{ang}}}{2}(\theta - \theta_{\text{eq}})^2 \tag{2.4}$$

Similar to the bond energy term (Eq. 2.3), the parameters k_{ang} and θ_{eq} correspond to strength and equilibrium bond angle and are specific to the combination of atom types involved in forming θ. In some FFs, the functional form of Eq. 2.4 is replaced by a trigonometric potential (Eq. 2.5).[12]

$$E_{\text{angles}} = \sum_{\text{angles}} \frac{k_{\text{ang}}}{2}\Big(\cos(\theta) - \cos(\theta_{\text{eq}})\Big)^2 \tag{2.5}$$

Many FFs augment the description of the bending energy by adding so-called Urey-Bradley terms[46] in order to improve the fitting to vibrational spectra.[12] These

[a] Atom types refer to different variants of atoms for the same element and depend on their respective bonding pattern. For example, FFs could distinguish aliphatic and aromatic carbon atoms, as well as the hydrogen atoms bonded to them. Different combinations of atom types usually have distinct parameters for the calculation of bonded and nonbonded terms associated with them.

terms follow the same functional form as Eq. 2.3 and are equivalent to adding a pseudo bond between the two external atoms (atoms a and c in Figure 2.1B).

For all cases where four atoms a, b, c, and d are bonded in a row, an additional torsional energy $E_{\text{dihedrals}}$ is added to the bonded terms. It is associated with the dihedral angle ϕ between two planes sharing two common atoms, i.e. the plane defined by atoms a, b, and c and the plane defined by atoms b, c, and d (see Figure 2.1C). The torsional energy is usually represented by a sum over cosine functions running over all atom quartets bonded in a row (Eq. 2.6).[12]

$$E_{\text{dihedrals}} = \sum_{\text{dihedrals}} \frac{k_{\text{dih}}}{2} \left[1 + \cos(n\phi - \delta)\right] \tag{2.6}$$

Here, k_{dih} is a strength parameter, δ is the phase shift of the cosine function, and n is an integer which determines how many minima and maxima the cosine function has in the angular range between 0 and 2π. In order to construct a torsional potential where the minima have different depths, two or more terms with different n can be combined. Similar to bonding and bending energies (Eqs. 2.3 and 2.4), the parameters in Eq. 2.6 are specific to the atom types involved in forming ϕ. Some FFs use alternative functional forms for the torsional energy.[18]

Finally, an additional bonded term is necessary to make sure that certain groups of atoms are biased towards a planar arrangement (e.g. atoms bonded to an sp^2-hybridized carbon atom). These improper torsional terms are required because the ordinary torsional energy (Eq. 2.6) is not sufficient to maintain planarity.[12] The improper torsional energy is usually given by a sum running over all atom quartets which require such a correction (Eq. 2.7).

$$E_{\text{impropers}} = \sum_{\text{impropers}} \frac{k_{\text{imp}}}{2} \left[1 + \cos(2\omega - \pi)\right] \tag{2.7}$$

Here, ω is the improper angle associated with the out-of-plane motion of an atom d from the plane defined by atoms a, b, and c (see Figure 2.1D), the phase shift by π ensures that the minimum of the potential is located at planar configurations and the strength parameter k_{imp} is specific to the atom types involved in the interaction. Sometimes, the functional form of Eq. 2.7 is replaced by a term reminiscent of the bending energy (Eq. 2.4).[12]

In addition to the bonded terms described above, FFs model intermolecular and intramolecular interactions between atoms separated by more than three bonds with nonbonded terms, which are further decomposed into van der Waals and electrostatic contributions (Eq. 2.8).[12]

$$E_{\text{nonbonded}} = E_{\text{vdW}} + E_{\text{electrostatic}} \tag{2.8}$$

Van der Waals interactions between atoms consist of short-ranged repulsive and long-ranged attractive forces. The repulsion is due to the exchange interaction,[a] a quantum mechanical effect related to the Pauli exclusion principle,[47] which becomes active when two electron clouds start to overlap. The attractive component arises due to interactions of spontaneously induced dipoles. Van der Waals interactions are usually modelled with Lennard-Jones (LJ) potentials[19]

$$E_{\mathrm{vdW}} = \sum_i \sum_{j>i} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] \tag{2.9}$$

where the sum runs over all pairs of nonbonded atoms i and j,[b] r_{ij} is the distance between those atoms, and ϵ_{ij} and σ_{ij} are parameters that control the strength of the interaction and the distance for which the balance between repulsive and attractive forces nets zero (the LJ potential reaches a minimum value of $-\epsilon_{ij}$ when $r_{ij} = 2^{\frac{1}{6}}\sigma_{ij}$). In principle, FFs could define different ϵ_{ij} and σ_{ij} for each possible pair of atom types, but for convenience[c] most FFs just define atomic parameters ϵ_i and σ_i for each atom type and the pair parameters are obtained by a combination rule.[48] For example, a popular choice are the Lorentz-Berthelot rules[49,50] given by Eqs. 2.10 and 2.11.[48]

$$\sigma_{ij} = \frac{\sigma_i + \sigma_j}{2} \tag{2.10}$$

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j} \tag{2.11}$$

Contrary to the r^{-6} dependence of the attractive part, the r^{-12} behaviour of the repulsive part of the LJ interaction is not physically motivated and merely chosen for mathematical convenience (like this, the repulsive part of the LJ potential can be obtained by simply squaring the attractive part). Some FFs use alternative functional forms for E_{vdW}, for example the Buckingham potential,[51] which models the exchange interaction with an exponential function.[12]

The remaining contribution to the nonbonded terms is of electrostatic nature and is usually modelled by Coulomb interactions between atom-centred point charges (PCs)

$$E_{\mathrm{electrostatic}} = \sum_i \sum_{j>i} k_e \frac{q_i q_j}{r_{ij}} \tag{2.12}$$

where the sum runs over the same pairs as Eq. 2.9, k_e is the Coulomb constant, q_i and q_j are parametrized partial charges depending on the types of atoms i and j,

[a] The exchange interaction is sometimes also called Pauli repulsion after physicist Wolfgang Pauli (1900–1958).

[b] Atoms are usually considered as nonbonded if they are parts of different molecules, or if they are part of the same molecule, but removed by more than three bonds.[12]

[c] If there are N different atom types, the number of possible pairs scales with $O(N^2)$. Thus, manually determining meaningful parameters for all of them quickly becomes infeasible.

and r_{ij} is the distance between them.

Modelling PESs with empirical FFs offers several advantages. First of all, since the terms involved in describing the potential energy E_{pot} (Eq. 2.1) are all given by simple analytical formulas, evaluating the PES is computationally efficient. This allows the application of FFs to systems involving thousands of atoms, for which a full solution of the Schrödinger equation (SE) is impossible. Further, once parameters for different atom types have been determined, a FF is transferable between all chemical systems involving the same types of atoms.

At the same time, conventional FFs have several limitations. For example, since the evaluation of the bonded terms (Eq. 2.2) relies on the specification of preconceived bonding patterns, they are unable to describe chemical reactions. Additionally, while the simple functional form of FFs is computationally efficient, it ultimately limits their accuracy.

In particular, the electrostatic model using atom-centred PCs (Eq. 2.12) is unable to capture certain features of the electrostatic potential (ESP) outside of molecules.[52] For example, it lacks a realistic description of the ESP around chemically important features such as lone pairs and σ-holes.[53] For this reason, the atom-centred point charge model (PCM) is sometimes replaced with distributed atomic multipoles (MTPs).[52] Whereas in PCMs, the contribution to the ESP by a single atom is given by

$$\text{ESP}(r,\theta,\phi) = \frac{q}{r} \tag{2.13}$$

where q is the partial charge of that atom, in MTP models, the contribution is given by a multipole expansion

$$\text{ESP}(r,\theta,\phi) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} Q_{lm} Y_{lm}(\theta,\phi) R_{lm}(r) \tag{2.14}$$

where the distance r, polar angle θ, and azimuthal angle ϕ describe the position of an arbitrary point in a spherical coordinate system centred around the atom. Here, $l_{\max}$ determines at what term the expansion is truncated,[a] Q_{lm} are expansion coefficients

[a] When the expansion is truncated at $l_{\max} = 0$, Eq. 2.14 reduces to Eq. 2.13 and the MTP model is equivalent to the conventional atom-centred PCM.

that need to be derived from a reference ESP, $R_{lm}(r)$ are radial functions,[a] and $Y_{lm}(\theta,\phi)$ are the real spherical harmonics (see Figure 2.2). They are given by

$$Y_{lm}(\theta,\phi) = \begin{cases} (-1)^m \sqrt{\frac{(2l+1)}{2\pi}\frac{(l-|m|)!}{(l+|m|)!}} P_l^{|m|}(\cos(\theta)) \sin(|m|\phi) & m < 0 \\ \sqrt{\frac{(2l+1)}{4\pi}} P_l^m(\cos(\theta)) & m = 0 \\ (-1)^m \sqrt{\frac{(2l+1)}{2\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos(\theta)) \cos(m\phi) & m > 0 \end{cases} \tag{2.15}$$

where $P_l^m(x)$ are the associated Legendre polynomials (Eq. 2.16)

$$P_l^m(x) = (-1)^m (1-x^2)^{\frac{m}{2}} \frac{\mathrm{d}^m}{\mathrm{d}x^m} P_l(x) \tag{2.16}$$

derived from the ordinary Legendre polynomials (Eq. 2.17).

$$P_l(x) = \frac{1}{2^l l!} \frac{\mathrm{d}^l}{\mathrm{d}x^l} (x^2-1)^l \tag{2.17}$$

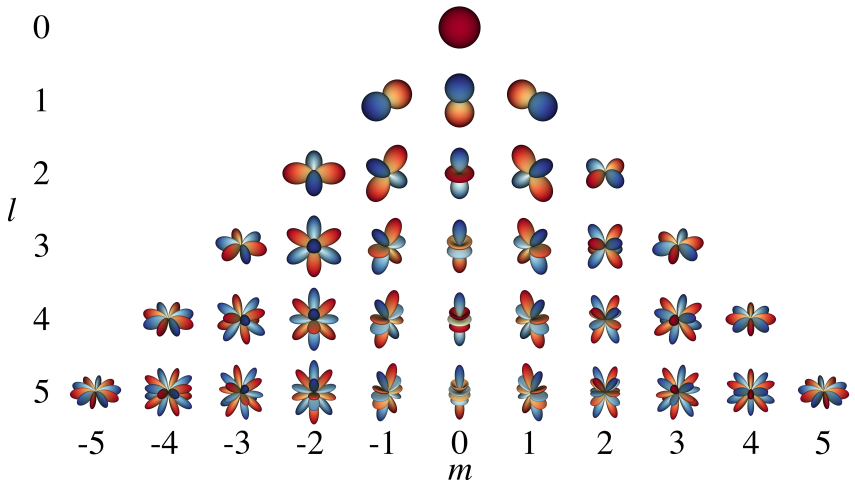


FIGURE 2.2: Visual representations of the real spherical harmonics Y_{lm} (Eq. 2.15) up to $l = 5$. Red and blue portions represent regions where the function is positive and negative, respectively.

While replacing the conventional PCM with MTPs can reduce the error in the ESP between 50–90%,[54] it also increases the computational cost,[55] because additional terms need to be evaluated to compute the electrostatic energy.[53] A computationally more efficient alternative to MTPs is the distributed charge model (DCM),[56] which takes advantage of the fact that the ESP generated by a multipole expansion

[a] The specific form of the radial functions depends on the values of l and m. For further details, refer to the appendix of [53].

(Eq. 2.14) truncated after the quadrupole moment ($l_{\max} = 2$) can equally be represented using an octahedral arrangement of six PCs.[57] However, computing the electrostatic energy with six PCs per atom is still more expensive by a factor of $6^2 = 36$ in comparison to using a single PC per atom.[a] By reducing the number of PCs, the computational efficiency of the DCM could be further improved.

Since any reference ESP can be represented exactly with an infinite number of PCs, there must exist a minimal finite number of PCs to represent it up to a given error threshold. Given a fixed number N of PCs with total charge Q, finding their optimal position $\mathbf{r}_i$ and magnitude q_i to represent a reference ESP can be formulated as an optimization problem which minimizes the root mean squared error (RMSE) between the ESP generated by the N PCs and the reference data:[b]

$$\mathrm{RMSE}(\boldsymbol{\rho}) = \sqrt{\frac{1}{J}\sum_{j=1}^{J}\left(\mathrm{ESP}(\boldsymbol{\rho},\mathbf{x}_j) - \mathrm{ESP}_j^{\mathrm{ref}}\right)^2} \tag{2.18}$$

Here, a total of J reference values $\mathrm{ESP}_j^{\mathrm{ref}}$ at positions $\mathbf{x}_j$ are given, $\boldsymbol{\rho}$ is a parameter vector containing the positions $\mathbf{r}_i$ and magnitudes q_i of all N charges, and $\mathrm{ESP}(\boldsymbol{\rho},\mathbf{x}_j)$ is calculated according to Eq. 2.19.

$$\mathrm{ESP}(\boldsymbol{\rho},\mathbf{x}_j) = \sum_{i=1}^{N}\frac{q_i}{||\mathbf{r}_i - \mathbf{x}_j||} \tag{2.19}$$

Note that $\boldsymbol{\rho}$ is only $(4N-1)$-dimensional instead of $(4N)$-dimensional, since q_N is determined by the total charge Q through $q_N = Q - \sum_{i=1}^{N-1} q_i$ and is therefore not a free parameter in the optimization.

A straightforward way to find a model using a minimal number of PCs for a given error threshold is to first optimize the RMSE for M charges, where M is some modest guess for the smallest number of PCs that are expected to be required, e.g. one per atom (as for a conventional PCM). If the error threshold is not met, the RMSE is optimized for $M+1$ charges and so on, until the results are satisfactory. While the above procedure seems straightforward, even for small molecules and only a single PC for each atom, the dimensionality of the problem is high and minimizing the RMSE is a difficult task where gradient-based approaches, such as the Levenberg-Marquardt method,[58] are prone to becoming trapped in local minima.

Fortunately, the optimization of objective functions (such as Eq. 2.18) is a common problem in many different fields including science and engineering, so a wide array

[a] The computational cost scales quadratically because a double sum over PCs needs to be evaluated in Eq. 2.12.

[b] The reference data is usually obtained from *ab initio* calculations.

of algorithms are available to tackle it.[59] A widely used gradient-free optimization algorithm is differential evolution (DE).[60] While DE uses similar computational steps to other evolutionary algorithms, the members of the current generation of solutions are perturbed by the scaled differences of other randomly selected members of the population, so that no probability distribution has to be assumed to generate offspring.[61]

In the rest of this chapter, the minimal distributed charge model (MDCM) is introduced, which aims to find an optimal arrangement of as few PCs as possible to fit a reference ESP. For this, a fitting procedure based on DE is developed, which can also fit the parameters of a conventional atom-centred PCM and MTPs to a reference ESP. The procedure is applied to four different molecules and the quality of the resulting models is compared. As proxies, water is chosen for being an ubiquitously important molecule, bromobenzene for its complicated ESP with pronounced σ-hole and imidazole as a model for a typical amino acid side chain. In order to test the performance for charged molecules, protonated imidazole is also considered. Using the same number of PCs as a conventional atom-centred PCM, it is shown that MDCMs achieve similar or better quality fits to the reference ESP than MTP models truncated after the quadrupole term ($l_{\mathrm{max}} = 2$). Further, it is demonstrated that MDCMs can reproduce electrostatic interaction energies at the same quality as MTPs by comparing both models to Coulomb integrals for ten water dimers commonly used for probing intermolecular interactions.[62] Finally, the numerical stability of the MDCM in molecular dynamics (MD) simulations and its ability to compute accurate thermodynamic quantities is exemplified by optimizing the nonbonded interactions for calculating the solvation free energy of bromobenzene from thermodynamic integration (TI). The calculated value is in good agreement with the experimental value, which was shown previously not to be achievable by conventional atom-centred PCMs.[63,64]

2.2 Construction of MDCMs

Section 2.2.1 presents the fitting procedure used to construct a MDCM for a given molecule. The *ab initio* calculations for obtaining the reference data are detailed in section 2.2.2. MDCMs were determined for four different molecules and results are presented and discussed in section 2.2.3.

2.2.1 Fitting procedure

Differential evolution (DE) The DE algorithm is the core of the MDCM fitting procedure. In the following, DE is described succinctly. For a more detailed introduction, refer to [60].

The DE algorithm expects an objective function $f(\boldsymbol{\rho})$ to be minimized as input, where $\boldsymbol{\rho}$ is a vector of D parameters.[60] It should be noted that DE is not guaranteed to find the global optimum of $f(\boldsymbol{\rho})$. Rather, DE efficiently explores the available parameter space, focusing on regions where the value of $f(\boldsymbol{\rho})$ is low. A formulation of the standard version of DE in pseudocode is given by Algorithm 2.1.

Algorithm 2.1 Standard form of the DE algorithm. Parameter vectors $\boldsymbol{\rho}_i$ are initialized with randomly chosen parameter values. Symbols such as $\mathbf{u}[j]$ denote the jth entry of vector $\mathbf{u}$.

```
Initialize population {ρ_i, i = 1 ... NP}
G = 0
while not converged and G < G_max do
    for i = 1 ... NP do
        randomly select ρ_a, ρ_b, ρ_c (a ≠ b ≠ c ≠ i) from population
        draw random integer j_rand between 1 and D
        for j = 1 ... D do
            if rand[0,1) < CR or j = j_rand then
                u[j] = ρ_a[j] + F · (ρ_b[j] − ρ_c[j])
            else
                u[j] = ρ_i[j]
            end if
        end for
        if f(u) < f(ρ_i) then
            ρ_i = u
        end if
    end for
    G = G + 1
end while
```

The maximum number of generations $G_{\max}$, the population size NP, the cross over ratio CR and the differential weight F are hyperparameters which are chosen depending on the problem. Increasing NP slows down the rate of convergence but makes the search more global (i.e. increases the chances of finding the global optimum). The differential weight F is usually chosen between 0 and 2, and determines how much the parameters of $\boldsymbol{\rho}_a$ are modified by the difference $\boldsymbol{\rho}_b - \boldsymbol{\rho}_c$ to generate a new trial solution $\mathbf{u}$. The optimal value of F is problem specific (a poor choice usually affects the speed of convergence). The cross over ratio CR controls the probability of choosing new parameters for the trial solution $\mathbf{u}$ versus reusing parameters from the current $\boldsymbol{\rho}_i$. For separable objective functions, $CR < 1$ can lead to faster convergence. The random integer j_{rand} guarantees that at least one parameter in $\mathbf{u}$ is different from $\boldsymbol{\rho}_i$ even when $CR = 0$. The algorithm is terminated either when G reaches $G_{\max}$ or if the population converges to a single solution, but other problem specific convergence criteria are also possible, e.g. when the value of any $f(\boldsymbol{\rho}_i)$ drops below a certain threshold.

The standard DE algorithm is not applicable to constrained optimization problems. A simple modification to allow constraints would be to exclude infeasible solutions

from the population entirely (i.e. reject every trial solution $\mathbf{u}$ that violates the constraints). However, this is usually found to be detrimental to the exploration of parameter space, as often two feasible regions are separated by an infeasible region which needs to be traversed. To allow the necessary (infeasible) intermediate steps, "soft" constraints can be used instead: The objective function f is modified by adding a penalty term, which only becomes active if a solution violates the constraints and gradually increases the bigger the constraint violation is. Unfortunately, it can be very difficult to tune the penalty function such that it neither penalizes constraint violations too harshly nor too leniently and a wrong balance leads to a bias during the search. For this reason, a slightly modified version of DE for constrained optimization was developed,[65] which leaves the objective function untouched. The main modification is to allow infeasible solutions to remain in the population over feasible solutions with a certain probability if their value of the objective function is lower. This probability is gradually decreased, so that the search space can be explored in infeasible regions early on, while finally, the population is driven only towards feasible solutions. For a more detailed description of the method, refer to [65].

To fit MDCMs with DE, the objective function $f(\boldsymbol{\rho})$ is chosen to be the RMSE($\boldsymbol{\rho}$) (see Eq. 2.18) and the parameter vectors $\boldsymbol{\rho}$ contain charge positions and magnitudes. To ensure that the repulsive part of the LJ potential keeps any two charges from approaching arbitrarily close in MD simulations (which would lead to numerical instabilities), the charge positions are constrained to remain in the vicinity of the nuclei.[a] Even though DE is not guaranteed to find the optimal solution, the results show that the best solutions achieve chemical accuracy (1 kcal mol^{-1} e^{-1}) and are therefore suitable for atomistic simulations.

Usually, is is sufficient to initialize $\boldsymbol{\rho}$ randomly (see Algorithm 2.1). For fitting MDCMs with DE however, it was found that such an initialization only gives satisfactory results for up to about seven PCs per molecule. For a larger number of charges, the dimensionality of parameter space is too large and it is therefore difficult to sufficiently cover the search space, unless the population size NP is increased dramatically. Instead of random initialization, it is therefore preferable to initialize the population with meaningful candidate solutions (initial guesses), such as to guide DE towards promising regions of parameter space.

In order to obtain meaningful initial guesses, the molecular ESP is first decomposed into atomic contributions. The atomic ESPs usually are more isotropic and it is reasonable to expect good solutions with few PCs. These "atomic MDCMs" are then used to construct the initial population for fitting an MDCM to the molecular ESP. For this purpose, it is useful to determine several different atomic MDCMs per atom, each with different numbers of PCs, such that a variety of initial guesses can be constructed.

[a] Here, in the vicinity means within 33% of the van der Waals radius[66] of a nucleus.

Determination of atomic ESPs For simplicity, the molecular ESP is decomposed into atomic contributions by constructing an MTP model. As distributed multipoles[52] represent the molecular ESP as a superposition of multipole expansions (see Eq. 2.14) centred on each atom, the separation into atomic contributions is straightforward.

To fit an MTP model, the RMSE (Eq. 2.18) is minimized by DE, where $\boldsymbol{\rho}$ now contains the multipole coefficients Q_{lm} (see Eq. 2.14) and a sum over the atomic multipole expansions (Eq. 2.14) is used to calculate the molecular ESP.[63] Due to the nature of a multipolar expansion, a straightforward way to simplify the fitting problem is to fit MTPs of different rank l separately. This breaks the high-dimensional problem into a series of lower-dimensional steps. Since the quality of a multipole expansion in describing the ESP improves by adding higher-order terms, each subsequent term can be considered to be a correction which reduces the error of the previous lower-order expansion.[63]

This suggests the following iterative fitting procedure: First, the monopole terms ($l = 0$) are optimized, while all other MTP parameters are set to zero. Then, the optimized monopole parameters are kept fixed and only dipole parameters ($l = 1$) are optimized, keeping all higher-order parameters set to zero and so on, until the parameters of the highest-order term $l_{\max}$ are optimized.[63] In principle, it is even possible to simplify the problem further by constraining the molecular dipole, quadrupole, or higher-order moments to match the respective *ab initio* reference values. Here, only the monopole parameters are constrained to reproduce the total molecular charge Q.

It is important to note that the MTP model is only required for obtaining "atomic ESPs" as reference data. Once a decomposition into atomic ESPs is available, the remaining steps in the fitting procedure are independent of how this reference data was generated. Other decomposition procedures to obtain atomic ESPs are possible: Atoms in molecules (AIM),[67] GDMA,[52] iterated stockholder atoms,[68] or the MTPs derived from them[69] could be used to construct the reference data instead.

Construction of atomic MDCMs After atomic ESPs have been obtained, atomic MDCMs are fitted to each of them. Since atomic ESPs are more isotropic than the molecular ESP, fewer PCs are needed to represent them. It was found that one to five PCs give satisfactory results and random initialization of parameters in DE is sufficient to find good solutions. For a large-scale parametrization project involving a range of molecules, it might even be possible to reuse atomic MDCMs between different molecules with the same atom types without refitting to individual atomic ESPs at all.

Generation of initial guesses After atomic MDCMs with different number of charges have been constructed, the question remains of how to generate initial guesses for fitting the molecular ESP. For many molecules, the ESP shows varying degrees of anisotropy around different atoms, so it is meaningful to construct initial guesses accordingly: With increasing anisotropy of the ESP around an atom, the number of PCs placed in its vicinity should also increase. The RMSE difference ΔRMSE between atomic MDCMs with different numbers of PCs to the corresponding atomic ESP is taken as a heuristic guideline. If the atomic ESP is largely anisotropic, the RMSE of atomic MDCMs is expected to decrease significantly with an increasing number of charges, whereas a largely isotropic ESP is expected to be already well represented by a model with fewer charges. A greedy algorithm then decides how to combine atomic MDCMs to different initial guesses: How many charges to use in the vicinity of each atom is determined by a weighted probability that depends on ΔRMSE.

As an example, consider bromobenzene, which features a pronounced σ-hole (see Figure 2.3). It is apparent that the ESP around the bromine atom is more complicated than in the vicinity of other atoms and more PCs are necessary to describe it. Assume an MDCM with 13 PCs should be constructed, which has one additional PC compared to a conventional atom-centred PCM. A completely random assignment would place the additional charge only one out of twelve times in the vicinity of the bromine atom. However, it is intuitively more promising to construct initial guesses in such a way that in most of them the additional charge is placed close to the bromine atom. The greedy procedure automatically ensures that initial guesses are constructed accordingly, because the atomic MDCMs for bromine will show the greatest ΔRMSE upon addition of charges. It was found that as a last step, it is beneficial to further randomly modify magnitudes and positions of the initial guesses by a small amount in order to increase the diversity of the initial population.
Finally, it should be noted that although charges are "assigned" to individual atoms for generating the initial guesses, PCs are not restricted during the DE fit to remain in the vicinity of a specific atom. They can still change their magnitudes and positions as long as this improves the overall RMSE. In the final MDCM, any assignment of PCs to atoms is essentially arbitrary. The algorithm determines all parameters based only on the molecular ESP. Nuclear coordinates are merely used as reference points to maintain charges within the van der Waals volume of the molecule (for numerical stability in MD simulations). However, if necessary for a specific application, assignment of charges can easily be constructed *a posteriori* by a distance criterion.

Summary Since the procedure described above effectively reduces the problem of fitting the molecular ESP to fitting the ESP around individual atoms, it is expected to perform well even for large molecules. Additionally, large molecules are typically divided into subsystems (e.g. amino acid side chains), which further reduces the

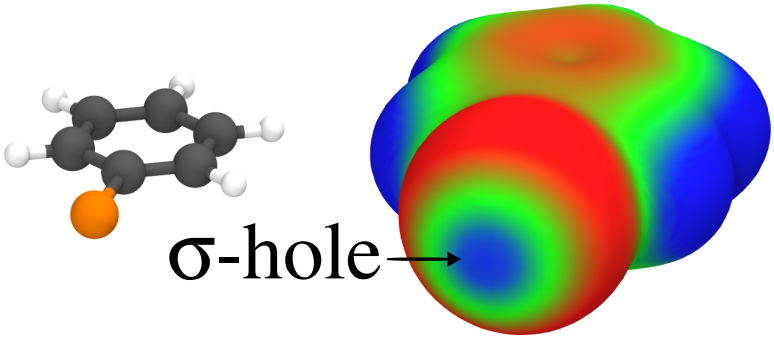


FIGURE 2.3: Depiction of the σ-hole in bromobenzene. **Left**: Ball-and-stick model of bromobenzene (Br: orange, C: black, H: white). **Right**: Visualization of the corresponding ESP. Colours range from red (negative) over green (zero) to blue (positive). The σ-hole is the positive part of the ESP around the predominately negative Br atom.

complexity. The key steps of the fitting procedure are summarised below and the workflow is schematically shown in Figure 2.4.

1. **Generation of atomic ESP reference data.** The molecular reference ESP is decomposed into atomic contributions by iteratively fitting an MTP model to the reference molecular ESP. Atomic ESPs are generated from the resulting multipole parameters. Different decomposition schemes (AIM,[67] GDMA[52] or iterated stockholder atoms[68]) could also be used.

2. **Fitting atomic MDCMs to atomic ESPs.** Atomic MDCMs with different numbers of PCs are fitted to the atomic ESPs from the previous step. Since only few parameters need to be fitted, random initialization is sufficient to find good solutions with DE.

3. **Fitting molecular MDCM to the reference ESP.** Initial guesses are constructed greedily from different combinations of the atomic MDCM solutions obtained in the previous step. After the population has been constructed, DE is used to minimize the RMSE. MDCMs with an increasingly larger number of PCs can be constructed starting from the same atomic MDCMs (the previous step needs to be performed only once) until the desired RMSE threshold or convergence is reached.

2.2.2 *Ab initio* calculations

The *ab initio* calculations needed to obtain the reference ESP were performed using the Gaussian09 suite of software.[70] Data for water was calculated at the CCSD(T)/aug-cc-pVQZ level of theory, all other molecules are treated at the B97D3/aug-cc-pVTZ

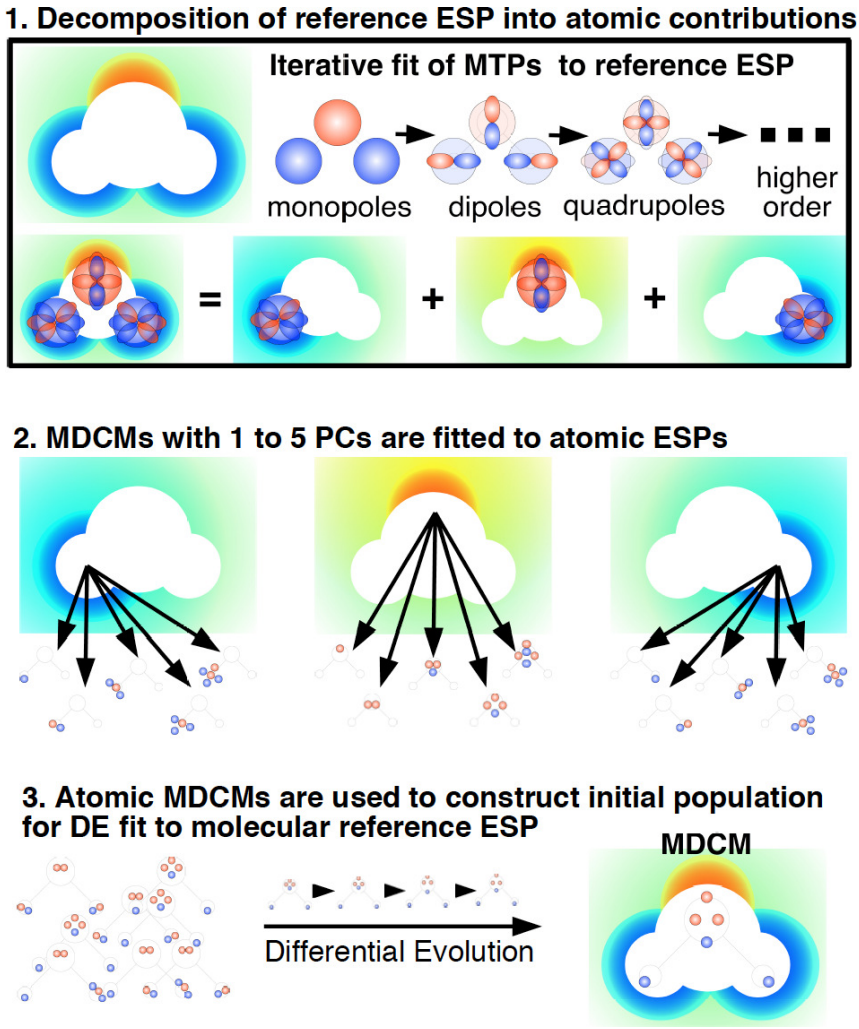


FIGURE 2.4: Schematic representation of the workflow for fitting an MDCM for water. The decomposition in step 1 is a black box (i.e. only its final results are important) that could be replaced by other decomposition methods.

level of theory. The structures of all molecules were optimized and the ESP and density data were extracted on a regular three-dimensional grid using the cubegen utility of Gaussian09. For each molecule, a coarse and a fine grid was constructed, the exact grid specifications are listed in Table 2.1.

TABLE 2.1: Exact grid specifications of *ab initio* reference data for fitting MDCMs for water, imidazole and bromobenzene.

molecule	coarse grid	fine grid
water	0.28 Å $[20 \times 20 \times 20]$ (84 %)	0.07 Å $[80 \times 80 \times 80]$ (82 %)
imidazole	0.43 Å $[22 \times 21 \times 17]$ (85 %)	0.10 Å $[90 \times 83 \times 68]$ (83 %)
bromobenzene	0.59 Å $[23 \times 18 \times 18]$ (90 %)	0.14 Å $[94 \times 74 \times 74]$ (90 %)

The first number in each column gives the grid spacing in Å, the values in square brackets give the number of grid points on each principal axis $[N_x \times N_y \times N_z]$ and the percentage in brackets indicates how many points out of the total are outside the 0.001 au isodensity surface (points inside are excluded both from the fit and the analysis). All grids are centred on the centre of mass of the respective molecule. The grid points for water, imidazole and bromobenzene extend at least up to a distance of 5.3 Å, 6.9 Å and 10.5 Å from the molecular centre, respectively.

2.2.3 Results and discussion

MDCMs were fitted for water, bromobenzene, imidazole, and protonated imidazole using the procedure described in section 2.2.1. To assess the quality of the results, the ESP generated by MDCMs with different number of PCs (denoted as MDCMX, where X is the number of charges) are compared to ESPs generated by MTPs of increasing rank. A maximum rank $l_{\max} = 5$ was chosen for the MTPs, which corresponds to a truncation after the ditriantapole (32-pole) term. Usually, the expansion is truncated much earlier after the quadrupole term ($l_{\max} = 2$).[71] Note that MTPs truncated after the monopole term ($l_{\max} = 0$) are equivalent to a conventional atom-centred PCM. For a comparison independent of the fitting procedure described in section 2.2.1, MTP models obtained from the GDMA program[72] are also included.

Since the computational complexity for the evaluation of the RMSE (Eq. 2.18) scales linearly with the number of reference points J, the RMSE during the fitting is calculated using a coarse grid spacing and the results are evaluated on a fine grid (see Table 2.1). Note that during both fitting and evaluation, only the grid points outside the 0.001 au isodensity surface are considered, which is a good approximation to the Lee-Richards molecular surface.[73] This approach has been used previously[56,74] and eliminates any influence of the ESP "inside" the molecule, which is irrelevant for atomistic simulations.

The primary motivation for fitting MDCMs is to reduce the computational cost of MD simulations while maintaining the accuracy of a multipolar FF. As such, it is instructive to compare the accuracy, i.e. the RMSE, of each model with its associated computational complexity. Unfortunately, it is difficult to compare MTPs and MDCMs directly in terms of wall-clock time, since the computation time strongly depends on the particular cut-off schemes used for the nonbonded interactions as well as the underlying implementation.

For this reason, a complexity measure ζ is introduced, which simply counts the number of additional interaction terms relative to a conventional atom-centred PCM ($\zeta = 1$). A naive multipolar model truncated after the quadrupole term ($l_{\max} = 2$) has up to nine non-zero multipole moments, leading to 81 multipole-multipole interaction terms per atom-atom interaction, so $\zeta = 81$. A more efficient implementation, with diagonalized quadrupole tensor to leave maximally six non-zero multipole moments,[75] has $\zeta = 36$. A DCM with full octahedral charge arrangement has six charges per atom, i.e. also $\zeta = 36$. Because it is possible to construct MDCMs that use fewer PCs than there are atoms in the molecule, they can even have a complexity $\zeta < 1$. Since ζ measures only the number of interaction terms, it should be noted that the actual computational gains of MDCMs compared to multipolar models are likely to be larger than ζ implies due to the additional mathematical complexity of higher-order multipole-multipole interactions compared to the simple charge-charge interactions of an MDCM.[53,56]

The ESP in different regions around the molecule has varying importance depending on the application, so the RMSE and maximum errors are reported not only for the complete grid, but also separately for points close, intermediate and far away from the molecule. The definition of the different regions is that from the Lee-Richards molecular surfaces,[73] where the first interaction belt (up to 1.66 times the van der Waals radii R from atom centres) defines the close, the second interaction belt ($1.66R$ to $2.2R$) defines the intermediate and the third interaction belt (all points farther than $2.2R$) the far region. For a visual representation of the fitting quality, the reference ESP is reported alongside error maps for different models in two-dimensional slices of the grid (see e.g. Figure 2.5).

Water Although water appears to be a simple molecule, it has a comparatively complicated ESP due to the oxygen lone pairs and is therefore a useful test case to evaluate the performance of an MDCM. In addition, water is an ubiquitous molecule in chemistry and biology, which is important for many processes and therefore an important test case. Table 2.2 shows a comparison of the RMSE of the ESP between multipolar models of increasing rank and MDCMs with different numbers of PCs with respect to *ab initio* values. The reference ESP is shown alongside error maps in Figure 2.5. The best MDCMs outperform a multipole expansion truncated after the

quadrupole term ($l_{\max} = 2$). Figure 2.6 shows that MDCMs converge rapidly to the same RMSE value as high-order multipole expansions.

TABLE 2.2: Quality of GDMA, MTPs and MDCMs for water. RMSE and maximum errors (in brackets) for different models are given in kcal mol^{-1} e^{-1}. Results are reported for different regions (close, intermediate, and long range) according to the Lee-Richards molecular surfaces.[73] For MDCMs, the value in brackets denotes how many charges per atom are used.

model	total	close	intermediate	long
GDMA ($l_{max} = 0$)	5.89	8.33 (24.5)	4.15 (8.76)	2.49 (5.08)
GDMA ($l_{max} = 1$)	3.86	5.79 (20.4)	2.26 (7.20)	0.72 (2.49)
GDMA ($l_{max} = 2$)	0.79	1.23 (4.96)	0.34 (1.05)	0.12 (0.33)
GDMA ($l_{max} = 5$)	0.44	0.72 (4.70)	0.57 (0.26)	0.01 (0.06)
MTP ($l_{max} = 0$)	2.21	3.30 (10.9)	1.33 (3.78)	0.49 (1.58)
MTP ($l_{max} = 1$)	0.99	1.52 (8.38)	0.49 (1.78)	0.15 (0.57)
MTP ($l_{max} = 2$)	0.46	0.74 (3.58)	0.11 (0.51)	0.03 (0.10)
MTP ($l_{max} = 5$)	0.36	0.60 (2.79)	0.04 (0.22)	0.02 (0.04)
MDCM3 (1.00)	0.57	0.91 (4.84)	0.19 (0.69)	0.07 (0.22)
MDCM4 (1.33)	0.48	0.77 (4.24)	0.12 (0.54)	0.04 (0.14)
MDCM6 (2.00)	0.39	0.64 (2.11)	0.04 (0.18)	0.01 (0.02)
MDCM9 (3.00)	0.39	0.64 (1.98)	0.04 (0.14)	0.01 (0.03)

Imidazole Many amino acids carry functional groups with demanding electrostatic features. For simulations of protein-ligand or protein-protein interactions, these features need to be reproduced as accurately as possible. Imidazole is a suitable proxy for the histidine side chain and therefore a useful test case to assess the performance of the MDCM for protein side chains. Table 2.3 shows that MDCM6 (0.67 PCs per atom) is sufficient to reach chemical accuracy (1 kcal mol^{-1} e^{-1}) for the ESP. Models with more PCs outperform MTP models truncated after the quadrupole term ($l_{\max} = 2$) and converge rapidly to the quality of a high-order multipole expansion (Figure 2.7). Figure 2.8 demonstrates that the algorithm identifies atoms with more complex ESPs at their surface (here the electronegative nitrogen atoms) and assigns additional charges to them (the close grouping of charges introduces anisotropy).

Protonated imidazole In order to determine the performance of MDCMs for charged molecules, MDCMs were also constructed for protonated imidazole. The performance was found to be similar to neutral molecules (see Table 2.4).

Bromobenzene Bromobenzene has a prominent σ-hole at the bromine atom (see Figure 2.3), an anisotropic feature which cannot be captured by conventional atom-centred PCMs. Previous studies suggest that MTPs are necessary to reproduce

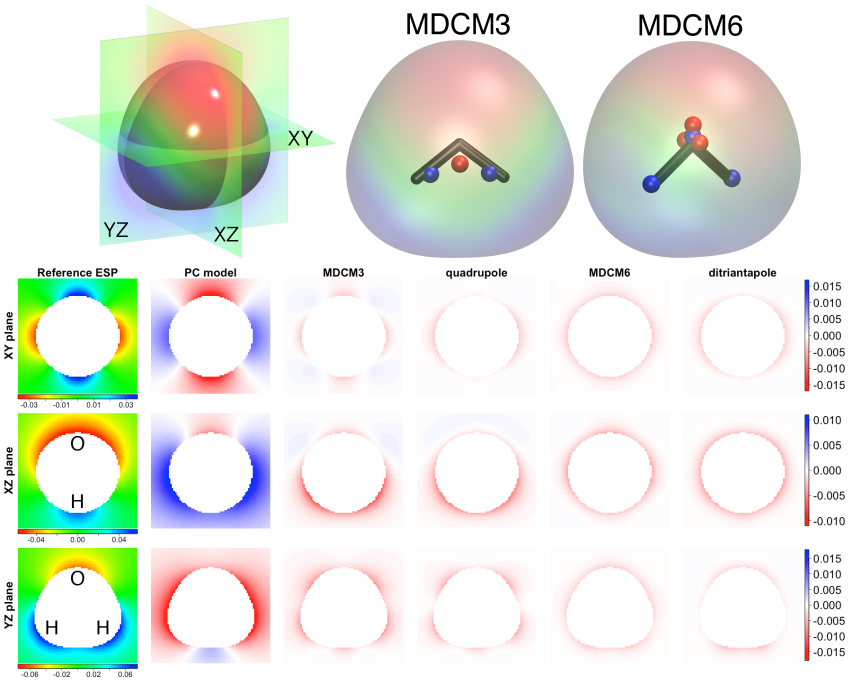


FIGURE 2.5: Error maps for water. **Top**: three-dimensional representation of the 0.001 a.u. isodensity surface along with two-dimensional cuts and visual representations of the MDCMs (negative PCs are shown in red, positive PCs in blue). **Bottom**: Reference ESP (leftmost column, colours encode numerical values; red: negative, green: zero, blue: positive) and error maps for different models. The columns "quadrupole" and "ditriantapole" denote MTP models which are truncated at the respective term ($l_{\max} = 2$ and $l_{\max} = 5$). All values are given in kcal mol^{-1} e^{-1}.

TABLE 2.3: Quality of GDMA, MTPs and MDCMs for imidazole (see Table 2.2).

model	total	close	intermediate	long
GDMA ($l_{max} = 0$)	3.01	4.82 (20.4)	2.70 (10.1)	1.83 (5.56)
GDMA ($l_{max} = 1$)	4.82	8.64 (25.2)	4.23 (15.1)	1.35 (7.12)
GDMA ($l_{max} = 2$)	0.83	1.69 (6.72)	0.44 (1.81)	0.14 (0.59)
GDMA ($l_{max} = 5$)	0.59	1.21 (5.37)	0.29 (1.38)	0.10 (0.44)
MTP ($l_{max} = 0$)	1.33	2.60 (16.5)	0.87 (3.85)	0.35 (1.55)
MTP ($l_{max} = 1$)	0.85	1.74 (12.0)	0.41 (2.05)	0.13 (0.57)
MTP ($l_{max} = 2$)	0.58	1.24 (9.17)	0.18 (1.46)	0.05 (0.29)
MTP ($l_{max} = 5$)	0.39	0.84 (4.35)	0.11 (0.69)	0.05 (0.20)
MDCM6 (0.67)	0.85	1.75 (11.5)	0.42 (2.26)	0.14 (0.67)
MDCM9 (1.00)	0.69	1.43 (9.34)	0.29 (1.51)	0.09 (0.46)
MDCM13 (1.44)	0.50	1.06 (4.59)	0.13 (0.69)	0.04 (0.21)
MDCM16 (1.78)	0.47	1.00 (4.28)	0.11 (0.59)	0.03 (0.19)

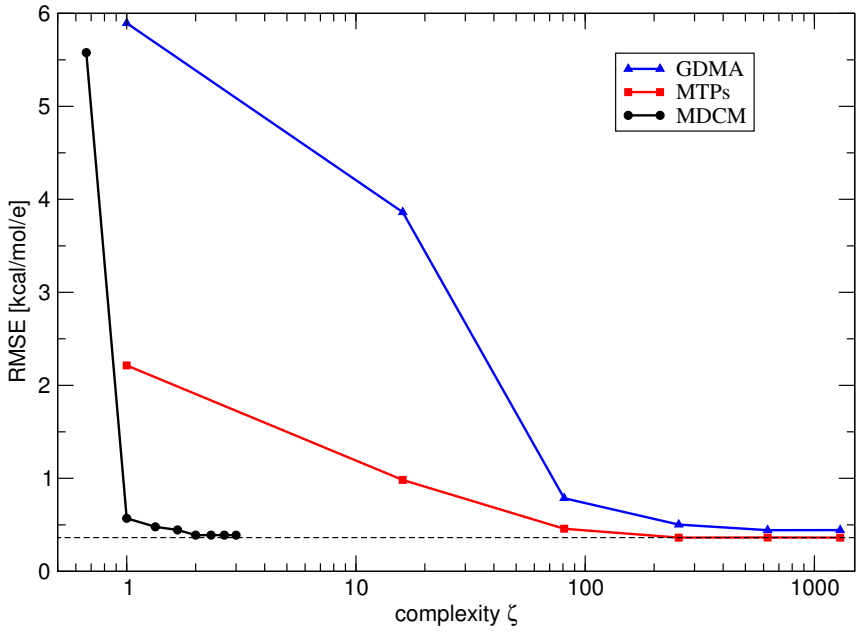


FIGURE 2.6: Convergence of MDCMs and MTPs for increasing model complexity ζ for water. The horizontal dashed line corresponds to the RMSE of a multipole expansion truncated after the ditriantapole (32-pole) term ($l_{\max} = 5$).

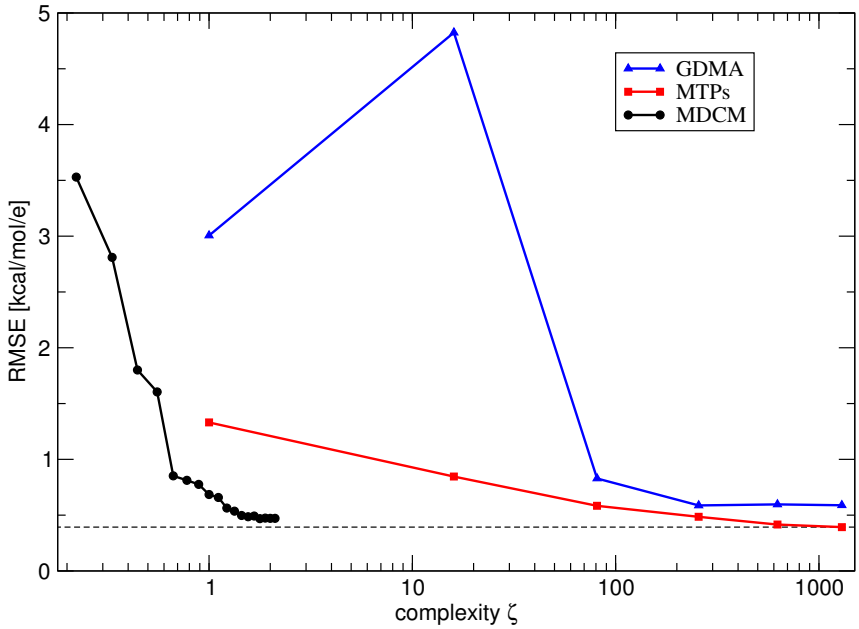


FIGURE 2.7: Convergence of MDCMs and MTPs for increasing model complexity ζ for imidazole (see Figure 2.6).

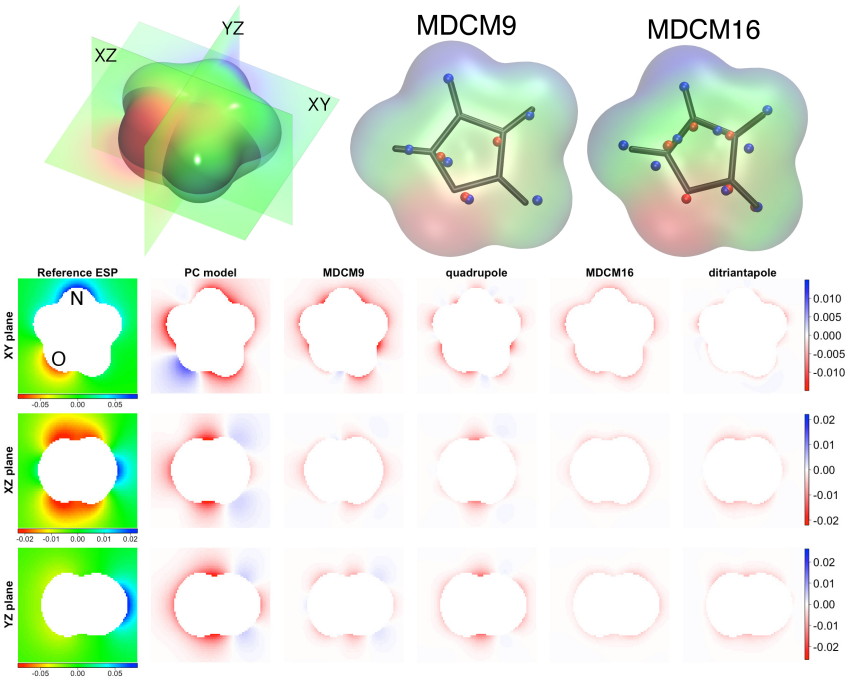


FIGURE 2.8: Error maps for imidazole (see Figure 2.5).

TABLE 2.4: Quality of GDMA, MTPs and MDCMs for protonated imidazole (see Table 2.2).

model	total	close	intermediate	long
MTP (l_{max} = 0)	0.79 (12.0)	1.54 (12.0)	0.40 (2.61)	0.14 (0.48)
MTP (l_{max} = 1)	0.60 (9.61)	1.19 (9.61)	0.26 (1.28)	0.09 (0.35)
MTP (l_{max} = 2)	0.41 (6.21)	0.84 (6.21)	0.01 (0.83)	0.03 (0.18)
MTP (l_{max} = 5)	0.29 (3.95)	0.59 (3.95)	0.08 (0.43)	0.03 (0.11)
MDCM5 (0.50)	0.73 (10.0)	1.44 (10.0)	0.38 (1.94)	0.12 (0.62)
MDCM10 (1.00)	0.49 (6.98)	1.00 (6.98)	0.17 (1.01)	0.06 (0.31)
MDCM13 (1.30)	0.39 (4.41)	0.79 (4.41)	0.11 (0.53)	0.04 (0.19)
MDCM18 (1.80)	0.32 (3.48)	0.65 (3.48)	0.06 (0.28)	0.02 (0.10)

the thermodynamic quantities of bromobenzene in MD simulations.[63,64,76–78] With MDCM12 (one PC per atom), it is possible to reproduce the σ-hole feature (see Figure 2.9). The fitting algorithm correctly identifies the halogen atom as a site requiring additional charges to describe the anisotropy of the σ-hole, and places two charges in close proximity along the C–Br bond axis to create a local atomic dipole and quadrupole moment. More remarkably, even with MDCM10 (fewer PCs than the number of atoms), the σ-hole and molecular ESP are still captured correctly. MDCMs of low computational complexity ζ outperform multipole models truncated after the quadrupole term ($l_{\max} = 2$) and converge rapidly to the same quality as a high-order multipole expansion (see Table 2.5 and Figure 2.10).

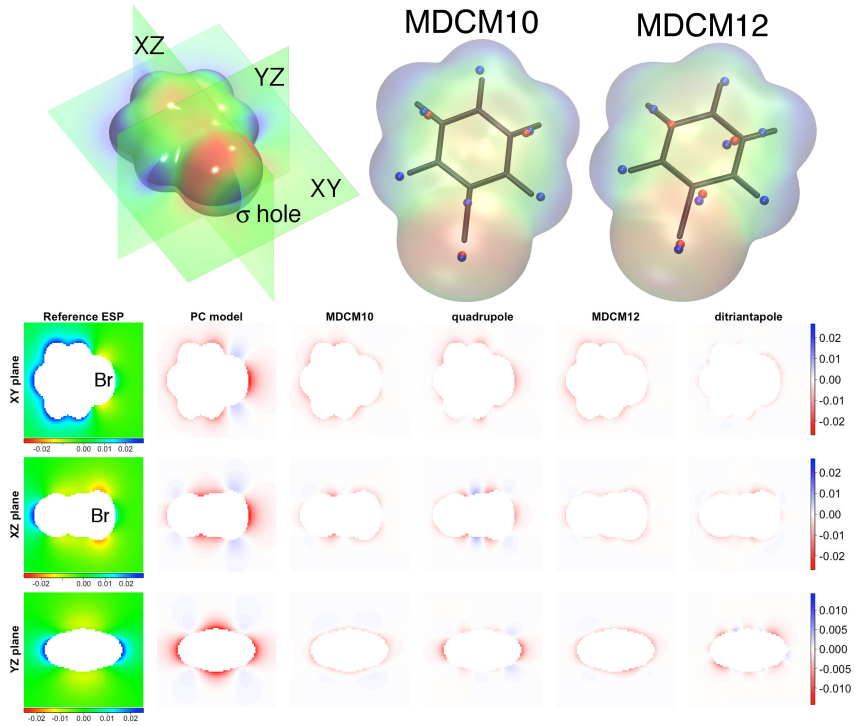


FIGURE 2.9: Error maps for bromobenzene (see Figure 2.5).

The MDCM12 for bromobenzene demonstrates that it is not necessary to include symmetry constraints during fitting, as the symmetry of the ESP is preserved despite the asymmetric charge distribution. This is because, since the reference ESP is symmetric, any asymmetric model ESP would have a large RMSE and be eliminated during the fitting procedure. As an independent check, the molecular dipole moment of this charge distribution was evaluated with the C–Br bond used to define a molecular $z-$axis and the phenyl ring lying in the xz plane, and found to be $\vec{\mu} = (\mu_x, \mu_y, \mu_z) = (-0.002, -0.001, -0.701)$ a.u.

TABLE 2.5: Quality of GDMA, MTPs and MDCMs for bromobenzene (see Table 2.2).

model	total	close	intermediate	long
GDMA (l_{max} = 0)	6.79 (36.3)	13.3 (36.3)	8.14 (17.0)	4.50 (10.4)
GDMA (l_{max} = 1)	3.47 (31.5)	8.00 (31.5)	4.05 (14.5)	1.63 (8.04)
GDMA (l_{max} = 2)	0.57 (8.04)	1.65 (8.04)	0.37 (1.86)	0.08 (0.63)
GDMA (l_{max} = 5)	0.37 (5.16)	1.09 (5.16)	0.21 (1.43)	0.06 (0.52)
MTP (l_{max} = 0)	0.88 (16.5)	2.39 (16.5)	0.77 (4.22)	0.24 (1.68)
MTP (l_{max} = 1)	0.67 (12.2)	1.92 (12.2)	0.43 (2.69)	0.12 (0.97)
MTP (l_{max} = 2)	0.49 (8.45)	1.45 (8.45)	0.23 (1.24)	0.06 (0.33)
MTP (l_{max} = 5)	0.33 (6.34)	0.95 (6.34)	0.17 (0.98)	0.06 (0.28)
MDCM7 (0.58)	0.78 (13.7)	2.08 (13.7)	0.70 (4.98)	0.24 (2.48)
MDCM10 (0.83)	0.42 (7.63)	1.23 (7.63)	0.19 (1.27)	0.05 (0.39)
MDCM12 (1.00)	0.38 (4.89)	1.13 (4.89)	0.16 (1.10)	0.04 (0.34)
MDCM19 (1.58)	0.35 (4.54)	1.04 (4.54)	0.12 (0.93)	0.04 (0.23)

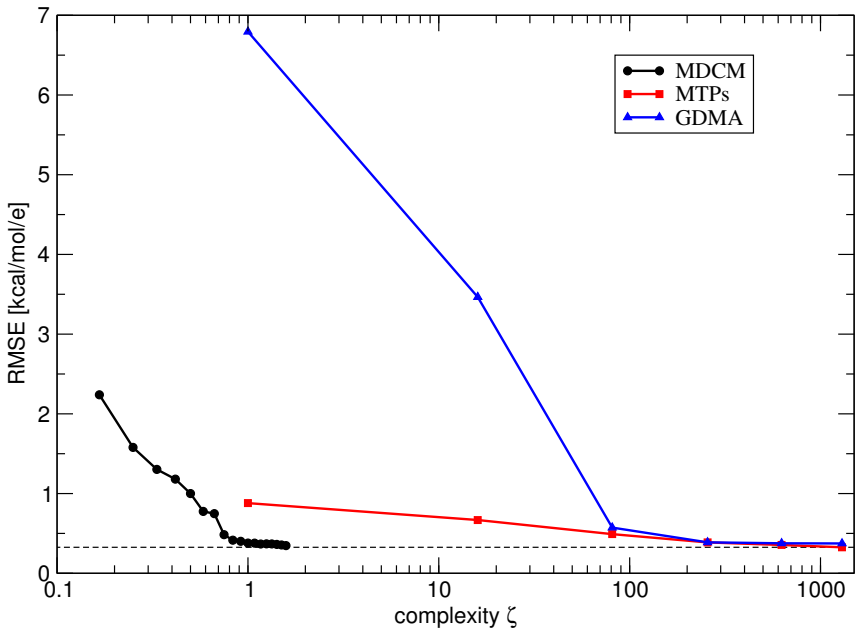


FIGURE 2.10: Convergence of MDCMs and MTPs for increasing model complexity ζ for bromobenzene (see Figure 2.6).

2.3 Applications

For the application in FFs, rather than the ESP, the total interaction energy is usually the quantity of interest. While the performance of the MDCMs in describing the ESP is encouraging, it is instructive to further examine how the differences between reference and fitted ESP affect interaction energies.

As the total interaction energy in conventional FFs consists of electrostatic and van der Waals contributions (see section 2.1), it is not possible to assess the performance of MDCMs in isolation when using the total interaction energy as reference. For a direct comparison with *ab initio* values, appropriate van der Waals parameters have to be determined for a particular MDCM, e.g. by comparing with thermodynamic reference data from experiment. To test the performance of the MDCM electrostatic term in combination with fitted van der Waals parameters, the hydration free energy of bromobenzene is selected as a reference quantity. It has been shown previously that conventional PCMs are insufficient to compute this value, even if LJ parameters are refitted.[79]

For a direct comparison of the electrostatic contribution to the interaction energy, a more suitable reference is the exact Coulomb integral over the *ab initio* electron densities of two frozen monomers (this assumes that second order polarization and charge transfer effects can either be neglected or handled by the remaining FF terms). For this reason, water dimer electrostatic interaction energies are also presented and compared to exact Coulomb integrals.

2.3.1 Methods

Coulomb integrals Reference electrostatic interaction energies for water dimers were calculated using CCSD(T)/aug-cc-pVQZ electron density cube files, with each cube file containing the monomer density in a common orientation. Monomers in each dimer were fixed at the gas phase geometries used to fit their MDCM, as conformational dependence of the ESP has not been considered in building the model. A fine density cube file with 50 points per Å (extending at least 3.5 Å from each nucleus) was generated for each monomer. Within 0.4 a.u. of each nucleus, where the electron density is large and changes quickly, every grid point was considered. Outside of this region, only every fifth grid point was used. The Coulomb integral was then evaluated across the grids using:

$$V_{\text{Coulomb}} = \frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \rho_i V_i \rho_j V_j}{r_{ij}} \tag{2.20}$$

where N_1 is the number of grid points in the cube file of monomer 1, ρ_i is the electron density at point i in the density cube file, V_i is the volume of the cube centred at this point with edge length equal to the spacing between grid points and r_{ij} is the

distance between point i in grid 1 and j in grid 2. The parameters used (grid spacing, extent, the size of the detailed region around the nucleus and the number of points discarded in the low density region) were selected so that V_{Coulomb} had converged to within 0.1 kcal mol^{-1} and the integration error of the charge of each monomer was on the order of 10^{-3} a.u. Due to the large number of grid points required for convergence, this procedure is only possible for small molecules such as water.

Thermodynamic integration (TI) TI was performed with CHARMM[80] using both standard "slow-growth" subroutines (implemented for LJ terms and in a locally modified version of CHARMM for the DCM module), and a procedure previously employed[63] for multipolar simulations, where the standard TI subroutines are not yet implemented. For the latter, simulations for each λ-window are propagated using the scaled Hamiltonian H_λ, but the total energy is recorded for each time step using the product Hamiltonian ($\lambda = 1$). If the energy variance for a trajectory exceeded $k_B T$, λ-windows were divided into smaller parts and the process repeated. For more details, refer to [63].

Solvation free energies were calculated for bromobenzene solvated in a box of 514 pre-equilibrated water molecules with periodic boundary conditions. The system was equilibrated for 80 ps with a 1 fs time step at 298 K at constant pressure and temperature. Ten independent 120 ps simulations were run per λ-window and averaged. Bond lengths involving hydrogen atoms were constrained using the SHAKE algorithm[81] and all nonbonded interactions were evaluated using shifted interactions with a cut-off of 12 Å.

2.3.2 Results and discussion

Water dimer interactions The ten water dimers of Tschumper *et al.*[62] (see Figure 2.11) were chosen as test systems that explore the water dimer PES, including many challenging short-range interactions. The electrostatic interaction energy was calculated for MDCM3, MDCM6, ESP-fitted MTPs truncated after the quadrupole term ($l_{\text{max}} = 2$), and for the popular TIP3P water model.[82] The performance of each model is compared to the electrostatic component of the interaction energy obtained by performing a numerical Coulomb integral (Eq. 2.20). The results show (see Figure 2.11 and Table 2.6) that MDCMs outperform the TIP3P model and their results resemble those of MTP models.
Quantitative measures for the quality of the models (see Table 2.7) show that MDCM3 and the MTP model truncated after the quadrupole term ($l_{\text{max}} = 2$) give similar results, whereas MDCM6 (four charges for oxygen, one for each hydrogen atom) outperforms the MTP model. The coefficient of determination R^2 measures how well the data is modelled by the best straight line fit (see Figure 2.11).
In Table 2.7, the RMSE is relative to the best straight line fit. The Pearson correlation coefficient[83] r directly measures the correlation between Coulomb integral and model

TABLE 2.6: Electrostatic interaction energies in kcal mol^{-1} for the water dimers depicted in Figure 2.11.[62] MDCM3 and MDCM6 are compared to an MTP model truncated after the quadrupole term ($l_{\max} = 2$) and values for the popular TIP3P model.[82]

dimer	MDCM3	MDCM6	MTP	TIP3P	Coulomb integral
1	−7.14	−6.86	−6.50	−6.68	−8.43
2	−6.24	−5.87	−5.58	−5.86	−7.17
3	−6.30	−5.71	−5.47	−7.06	−6.71
4	−6.00	−5.48	−5.68	−5.79	−6.80
5	−5.58	−4.94	−5.26	−5.35	−6.05
6	−5.80	−4.71	−5.20	−5.58	−5.67
7	−4.29	−3.67	−3.70	−4.50	−4.51
8	−1.60	−1.46	−1.39	−1.25	−1.75
9	−4.26	−3.66	−3.63	−5.17	−4.40
10	−3.05	−2.41	−2.46	−4.10	−2.72

TABLE 2.7: Quantitative measures for the performance of different electrostatic models for their ability to reproduce electrostatic interaction energy. The closer r and R^2 are to 1 and the lower the RMSE (given in kcal mol $^{-1}$), the better the model.

model	R^2	r	RMSE
MDCM3	0.9694	0.9846	0.2831
MDCM6	0.9970	0.9985	0.0866
MTP	0.9741	0.9870	0.2484
TIP3P	0.7965	0.8925	0.6968

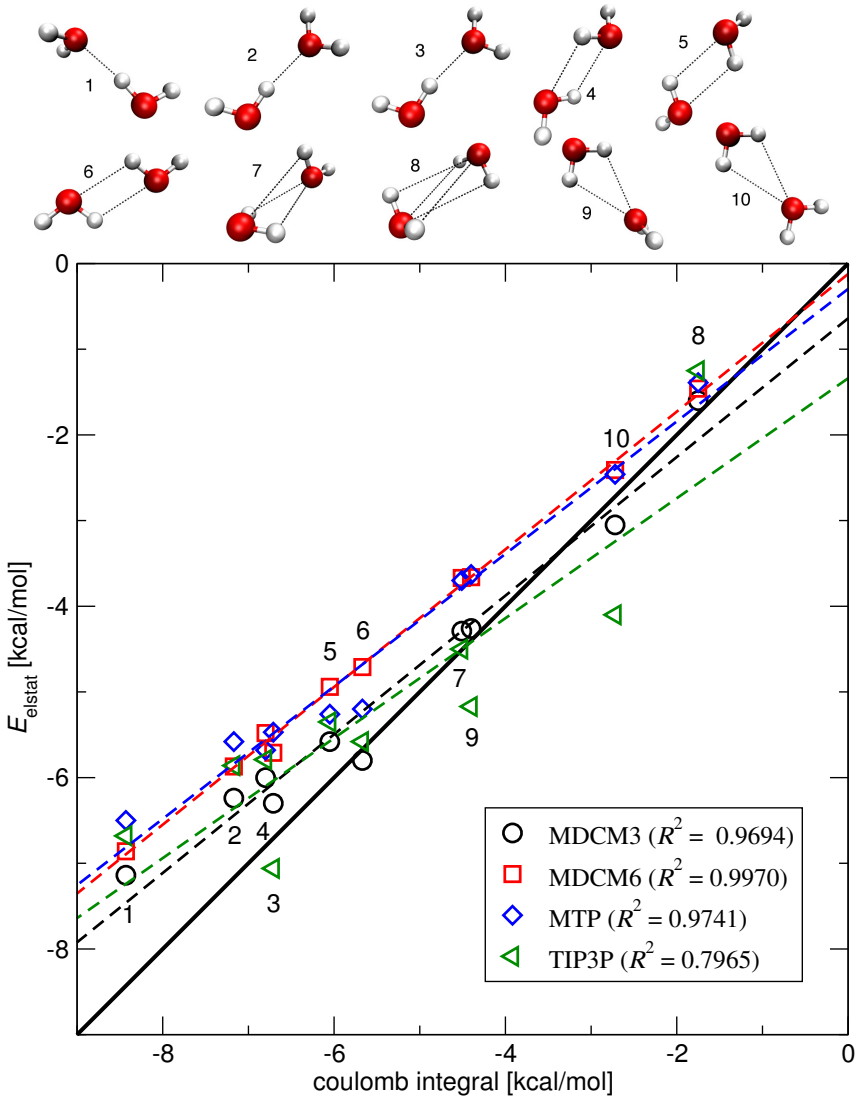


FIGURE 2.11: **Top**: Depiction of the water dimers[62] used to test the performance of various models in predicting the electrostatic interaction energy. **Bottom**: Correlation of the Coulomb integral and interaction energy computed with different models. The bold diagonal black line indicates perfect correlation and the dotted lines are linear fits to the data (the values are given in Table 2.6).

electrostatic interaction energies. The lower correlation coefficients of TIP3P[82] demonstrate that while the description of the dimers is still meaningful, the model does not capture relative energies of different dimer orientations accurately. The high correlation coefficient of MDCM6 in particular shows that the dimer electrostatic interaction energy is much better described. The systematic underestimation of the electrostatic interaction energy by MDCMs and MTP models with respect to the full Coulomb integral represents the well-known penetration energy that arises from overlapping electron densities in short-range interactions.[84] Correcting for this effect is a possible future improvement for MDCMs and MTPs, although MDCMs could also be used directly to replace multipole moments in FFs that already contain explicit correction terms.[85]

Comparison of MDCMs to other off-centre water models Recently, it was suggested that the shortcoming of multiple older water models are mainly due to substantial inaccuracies in the description of higher-order multipole moments.[86] It is therefore instructive to also compare charge positions and magnitudes of different MDCMs to other off-centre water models. Figure 2.12 compares charge positions of several MDCMs and other water models from the literature (the parameters for all models are given in Table 2.8).

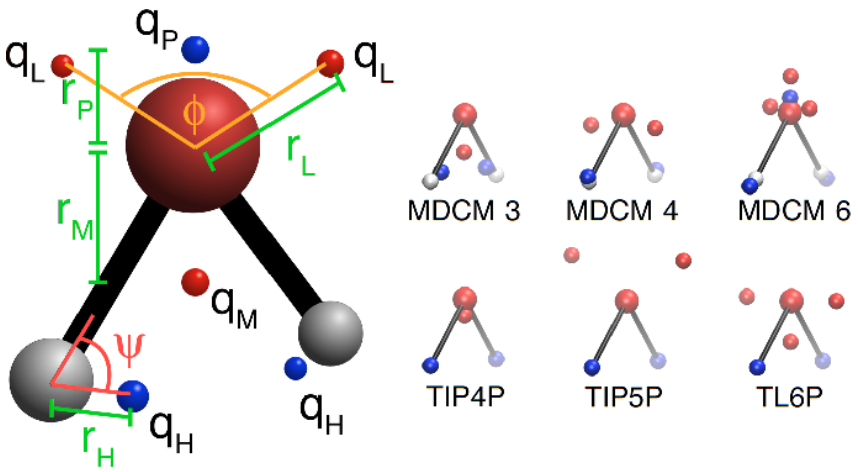


FIGURE 2.12: Comparison of different MDCMs to popular water models with similar numbers of charges. The scheme on the far left shows a unified representation, which describes all of the depicted models.

Interestingly, even though water is known to locally arrange in tetrahedral structures, the placement of negative charges suggests that the electric field around water is best described using a "T-shape" arrangement with the oxygen atom instead.[87] This feature is reproduced by the MDCMs automatically from the molecular reference ESP alone, see for example MDCM4 in Figure 2.12.

TABLE 2.8: Parameters (charges in e, angles in ° and distances in Å) for different MDCMs and popular water models (see Figure 2.12).

model	q_H	r_H	q_M	r_M	q_L	r_L	q_P	r_P	ϕ	ψ
MDCM3	1.26	0.25	−2.52	0.35	—	—	—	—	—	20.19
MDCM4	0.47	0.05	—	—	−0.47	0.34	—	—	216.38	273.26
MDCM6	0.32	0.14	−1.40	−0.34	−2.12	0.21	5.00	0.13	154.43	192.98
TIP4P[87,88]	0.52	0.00	−1.04	0.15	—	—	—	—	—	—
TIP5P[88]	0.24	0.00	—	—	−0.24	0.70	—	—	109.47	—
TL6P[86]	0.51	0.00	−0.57	0.38	−0.22	0.46	—	—	175.00	—

Solvation free energies for bromobenzene For computing the hydration free energy from MD simulations, MDCM10 was chosen, as it is comparable in quality to an MTP model truncated after the quadrupole term ($l_{\max} = 2$). Note that a conventional PCM for this molecule would use twelve charges (one for each atom). LJ parameters were fitted using the Fitting Wizard[79] and are listed in Table 2.9.

TABLE 2.9: Optimized LJ parameters for MDCM10 for bromobenzene (*) compared to values reported in the literature.[63,64,78] CA and HA are the atom types of carbon and hydrogen atoms in aromatic rings.

	atom type	σ [Å]	ϵ [kcal mol^{-1}]
*	CA	2.32	−0.05
	HA	1.07	−0.01
	Br	2.94	−0.20
[63, 64]	CA	1.78	−0.08
	HA	1.07	−0.01
	Br	2.05	−0.46
[78]	CA	2.82	−0.07
	HA	1.92	−0.03
	Br	2.75	−0.47

The computed solvation free energy of $\Delta G_{solv} = -1.55 \pm 0.07$ kcal mol^{-1} is within 0.1 kcal mol^{-1} of the experimental value (−1.46 kcal mol^{-1}).[89,90] It has been argued previously that σ-holes are not accurately represented by a conventional atom-centred PCM[76,77] and it was shown that they generally have an error of about 1 kcal mol^{-1}, i.e. the magnitude of the hydration free energy itself, whereas MTPs are able to reproduce the experimental value accurately.[63,64] In conclusion, MDCM10 with optimized LJ parameters performs comparably to a MTP model, but at considerably reduced computational cost.

2.4 Conclusion and outlook

The MDCM is a viable and efficient off-centre PC representation for faithfully describing the molecular ESP, for example for applications in MD simulations. The results show that positioning PCs away from atom centres can significantly improve the quality of the resulting ESP. Using the same number of charges, MDCMs reduce the error by between 48–74% compared to a conventional atom-centred PCM. If more charges than atoms are added, the error even reduces by 61–82%, which is similar to MTPs.[54]

The best MDCMs all outperform a multipole expansion truncated after the quadrupole term ($l_{\max} = 2$) and are sometimes even better in quality than a multipole expansion truncated after the octopole term ($l_{\max} = 3$). At the same time, they usually use fewer than two PCs per atom and are therefore by about a factor of 10 or higher more computationally efficient than a DCM,[56] while having the same advantages over MTPs in MD simulations. Remarkably, for imidazole and bromobenzene, it is even possible to find an MDCM of multipolar quality using fewer PCs than there are atoms.

In principle, ESPs of arbitrarily high quality should be possible by adding additional PCs in MDCMs. Nonetheless, as Figures 2.6, 2.7, and 2.10 show, a typical MDCM seems to converge somewhere between the quality of a multipole expansion truncated after the $l = 2$ and $l = 4$ terms. A possible explanation is that charge positions are constrained to be close to atom centres, whereas for the guaranteed convergence of infinitely many PCs to a reference *ab initio* ESP, arbitrary charge positions might be necessary. Since the remaining errors usually concern regions close to the 0.001 au isodensity surface, it is possible that MDCMs could be further improved by adding a penetration energy term.[84] The fact that error maps of MDCMs (see Figures 2.5, 2.8, and 2.9) reveal that ESPs are almost exclusively too negative around the molecular surface suggests that such a correction could be beneficial.[a]

MDCMs for large numbers of charges sometimes show asymmetric charge distributions, although the molecular symmetry would suggest a symmetric distribution (see MDCM12 for bromobenzene in Figure 2.9). The resulting ESP however can still be symmetric (see Figure 2.9) and asymmetric charge distributions are therefore of no concern for most practical purposes. If slight asymmetries should be problematic for some applications, solutions can always be symmetrized *a posteriori*. However, it was found that this has a slightly negative impact on the quality of the solution. Nonetheless, future improvements to the MDCM fitting method itself could take advantage of molecular symmetry to further reduce the number of free parameters

[a] The penetration effect is destabilizing here due to the penetration of a positive probe charge used to calculate the ESP, rather than the interpenetration of two negative electron density clouds in intermolecular interactions.

during fitting, similar to methods that exploit symmetry when fitting MTPs.[54] One possible strategy would be to identify symmetry related points in the reference ESP and to construct only a subset of charges directly, while additional charges could be obtained indirectly by applying the appropriate symmetry operations.

In the present work, only comparatively small molecules with challenging features in the molecular ESP were considered. As the fitting procedure is based on local information, extensions to larger molecules are straightforward. However, for molecules with several metastable minima (e.g. acrolein) it has been found that neglecting the conformational dependence of MTPs or averaging them over different conformers renders them worse than conventional atom-centred PC models.[54] It is likely that MDCMs are affected similarly, and charges would need to depend on the internal geometry as a countermeasure.

In conclusion, MDCMs are a possible way to improve empirical FFs. It was demonstrated that when using an MDCM for describing electrostatic interactions, FFs can be parametrized with accuracies for the hydration free energy within 0.1 kcal mol^{-1} of the experimental value. This opens the way for large scale parametrization of molecular building blocks for high accuracy and efficient condensed phase atomistic simulations. The developed fitting procedure based on DE is not only able to generate MDCMs, but also parameters for MTPs and conventional atom-centred PCMs.

Acknowledgements

I would like to thank Dr. Krystel El Hage for helpful discussions regarding TI and Dr. Mike Devereux for performing the calculations with CHARMM and Coulomb integrals.

Chapter 3

Reproducing kernel Hilbert space toolkit

This chapter starts with a small summary of linear regression and introduces the so-called "kernel trick", which can be used to apply linear methods to nonlinear data (section 3.1). The resulting kernel ridge regression (KRR), also known as reproducing kernel Hilbert space (RKHS) method, is discussed as a way to construct potential energy surfaces (PESs). In section 3.2, two algorithms are described to make constructing and evaluating PESs with KRR computationally more efficient. These methods rely on reference data being arranged in a grid structure, but workarounds in cases where the data is incomplete are described in section 3.3. Special kernels, which are particularly useful for constructing PESs, are described in section 3.4. In section 3.5, a six-dimensional toy problem is considered as a test case for which traditional KRR would be difficult to apply due to a large amount of reference data.

The methods discussed in this chapter are implemented in a Fortran90 code, the RKHS toolkit, which largely automates the construction of efficient PESs for small molecular systems. It is available from `https://github.com/MeuwlyGroup/RKHS` (a tutorial on how to use the RKHS toolkit for PES construction is given in appendix B). The results presented in this chapter have been previously published in [91].

3.1 Introduction

One of the most simple methods to model underlying trends in data is linear regression.[92] Given a set $\{(y_i; x_i)\}_{i=1}^{N}$ of N data points, a linear relationship between x and the dependent variable $y = f(x)$ is assumed, i.e. it is modelled as

$$\widetilde{f}(x) = \widetilde{y} = cx + b \tag{3.1}$$

and the so-called least squares solution for the parameters c and b is obtained by minimizing the mean squared error (MSE) between Eq. 3.1 and the data.[a] This leads to the following solutions for c and b

$$c = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2} \tag{3.3}$$

$$b = \bar{y} - c\bar{x} \tag{3.4}$$

where $\bar{x}$ and $\bar{y}$ are the mean values of x_i and y_i, respectively.[b] As an example, consider the dataset shown in Figure 3.1, for which it is reasonable to assume a linear relationship between x and y. Once the parameters are determined, Eq. 3.1 can be used to estimate the value of y where no data is available.

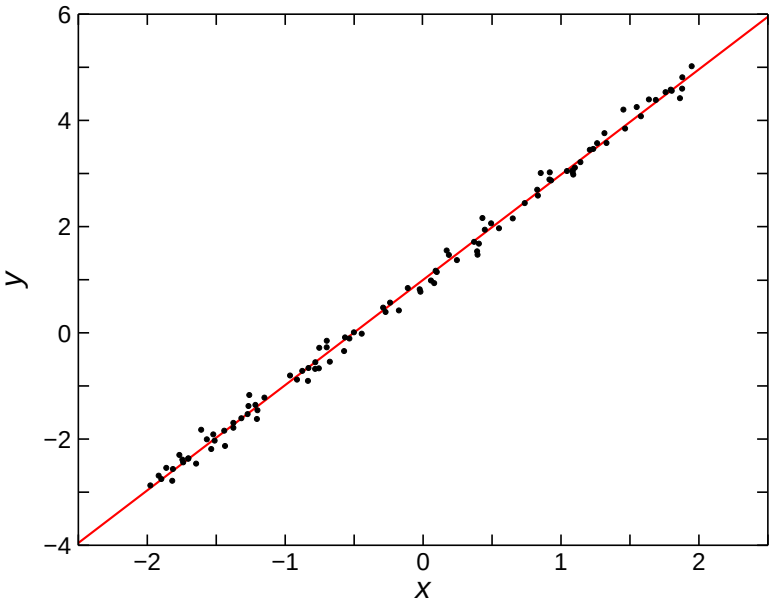


FIGURE 3.1: Linear regression line (red) for a dataset $\{(y_i; x_i)\}_{i=1}^{N}$ (black dots). The trend in the data is well explained by a linear relation.

[a] The MSE is given by

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widetilde{y}_i)^2 \tag{3.2}$$

where $\widetilde{y}_i = \widetilde{f}(x_i)$ is computed according to Eq. 3.1.

[b] The mean value $\bar{x}$ is given by

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.5}$$

This concept can also be extended to model linear relationships for higher-dimensional data of the kind $\{(y_i; \mathbf{x}_i)\}_{i=1}^{N}$, where each value of y depends on D different variables $\mathbf{x} \in \mathbb{R}^D$.[92] As for the one-dimensional case, a linear relationship

$$\widetilde{f}(\mathbf{x}) = \widetilde{y} = \mathbf{c} \cdot \mathbf{x} + b = \sum_{d=1}^{D} c^{(d)} x^{(d)} + b \tag{3.6}$$

is assumed, where $x^{(d)}$ and $c^{(d)}$ are the dth entries of the vectors $\mathbf{x} = [x^{(1)} \ \cdots \ x^{(D)}]^\mathsf{T}$ and $\mathbf{c} = [c^{(1)} \ \cdots \ c^{(D)}]^\mathsf{T}$. The parameters $c^{(d)}$ and b are obtained by a least squares solution of the system of equations given by

$$\begin{bmatrix} x_1^{(1)} & \dots & x_1^{(D)} & 1 \\ x_2^{(1)} & \dots & x_2^{(D)} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_N^{(1)} & \dots & x_N^{(D)} & 1 \end{bmatrix} \begin{bmatrix} c^{(1)} \\ \vdots \\ c^{(D)} \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{3.7}$$

which is equivalent to Eqs. 3.3 and 3.4 in the one-dimensional case.

Unfortunately, many data sets are characterized by a nonlinear relationship between independent and dependent variables. Applying linear methods directly to such problems leads to models which exhibit large errors when predicting unknown values. However, it is often possible to still obtain a linear model that performs well by first mapping the inputs to a higher-dimensional space. In this so-called feature space, the formerly nonlinear problem can become linear, provided that a suitable mapping is chosen.[93] As an example, consider the following linear classification problem:[a] In Figure 3.2A, blue and red data points are not linearly separable, i.e. it is impossible to draw a straight line that separates them. However, when the data is mapped to a higher-dimensional feature space in Figure 3.2B, blue and red data points become linearly separable.[b]

While it is possible to manually identify a feature space that linearises the toy problem shown in Figure 3.2, for real data, it is often difficult to find a suitable mapping. This is where the so-called "kernel trick"[94–96] proves to be beneficial. It allows one to operate in an implicit high-dimensional (or even ∞-dimensional) feature space without explicitly computing the coordinates of data in that space. Rather, the inner product between the images of the data in the feature space are computed directly. Since many linear methods can be reformulated in terms of inner

[a] Linear classification is similar to linear regression, but instead of finding a line that is as close as possible to a given dataset (see Figure 3.1), the task is to find a line that separates two classes of data.

[b] Linear separability in three dimensions means that the data can be separated by a plane. In general, data in a D-dimensional space is linearly separable if a $(D-1)$-dimensional hyperplane can be drawn to separate the classes.

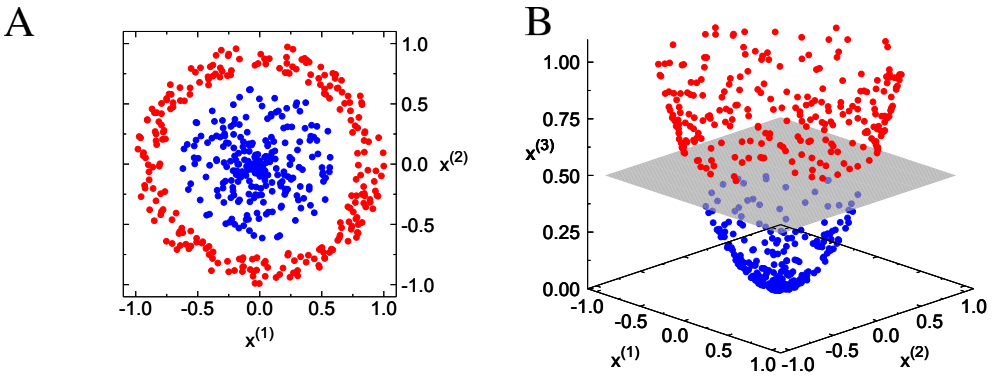


FIGURE 3.2: **A**: The blue and red points with coordinates $(x^{(1)}, x^{(2)})$ cannot be separated by drawing a straight line. **B**: By defining a suitable mapping $\phi : (x^{(1)}, x^{(2)}) \mapsto (x^{(1)}, x^{(2)}, x^{(3)})$, blue and red points can be separated by a plane at $x^{(3)} = 0.5$ (grey).

products,[41] the kernel trick allows to apply them to nonlinear data.

In order for the kernel trick to be applicable,[97] the feature space must be a so-called reproducing kernel Hilbert space (RKHS).[a] Every RKHS is associated with a kernel function $K(\cdot,\cdot) = \langle \phi(\cdot), \phi(\cdot) \rangle$, where $\phi : X \mapsto F$ is the mapping from input space X to feature space F. In other words, evaluating $K(\cdot,\cdot)$ is equivalent to evaluating the inner product $\langle \phi(\cdot), \phi(\cdot) \rangle$ in the feature space. However, when the kernel function is evaluated, the mapping $\phi(\cdot)$ never needs to be explicitly computed. In fact, it is not even necessary to know $\phi(\cdot)$, nor the structure of the feature space F, rather it is sufficient that they exist. It can be shown that every positive definite function is a proper kernel function associated with an RKHS.[99]

Many different kernel functions are possible, for example, a popular choice for input data of any dimension is the Gaussian kernel[b] given by

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2} \tag{3.8}$$

[a] An RKHS is a Hilbert space H of functions where all evaluation kernels are continuous linear functionals[98] In simple terms, this means that if two functions $f, g \in H$ are close in norm, i.e. $\|f - g\| < \epsilon$, where ϵ is small, the two functions are also pointwise close, i.e. for all x, $|f(x) - g(x)|$ is small, too.

[b] The feature space associated with the Gaussian kernel is ∞-dimensional,[41] which allows it to linearise most problems.

where γ is a hyperparameter that controls the width of the Gaussian. It is also possible to construct a kernel function for D-dimensional inputs as product over one-dimensional kernels $k(x, x')$, i.e.

$$K(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^{D} k^{(d)}(x^{(d)}, x'^{(d)}) \tag{3.9}$$

Once a kernel function is selected, the representer theorem[100] guarantees that any dataset $\{(y_i; \mathbf{x}_i)\}_{i=1}^{N}$ without contradictory entries[a] can be modelled as

$$\widetilde{f}(\mathbf{x}) = \widetilde{y} = \sum_{i=1}^{N} c_i K(\mathbf{x}, \mathbf{x}_i) \tag{3.10}$$

where the coefficients c_i are chosen to satisfy the linear relation

$$y_j = \sum_{i=1}^{N} c_i K_{ij} \tag{3.11}$$

with $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Therefore, the coefficients c_i are obtained by solving the system of linear equations $\mathbf{Kc} = \mathbf{y}$ (Eq. 3.12).

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1N} \\ K_{21} & K_{22} & \dots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{3.12}$$

As the kernel matrix[101,102] $\mathbf{K}$ is symmetric and positive definite by construction, the efficient Cholesky decomposition[103] can be used to solve Eq. 3.12. Once the coefficients c_i have been determined, the value of y for an arbitrary point $\mathbf{x}$ can be estimated using Eq. 3.10. This method is known as kernel ridge regression (KRR).[b]

Since KRR reproduces the reference data exactly (see Eq. 3.11), it is prone to overfit to noise potentially present in the training set. For this reason, it is sometimes beneficial to regularize the solution of Eq. 3.12 by adding a small positive constant λ to the diagonal of $\mathbf{K}$,[104] such that the coefficients satisfy

$$y_j = \sum_{i=1}^{N} c_i \left(K_{ij} + \lambda \delta_{ij} \right) \tag{3.13}$$

[a] This means that the dataset does not contain any entries where $\mathbf{x}_i = \mathbf{x}_j$ but $y_i \neq y_j$.

[b] KRR is the "kernelised" equivalent of linear regression (notice the similarity between Eqs. 3.6 and 3.10 and Eqs. 3.7 and 3.12).

instead of Eq. 3.11 (here, δ_{ij} is the Kronecker delta[a]). This method can also be used to obtain a solution to Eq. 3.12 in cases where the kernel matrix $\mathbf{K}$ is ill-conditioned. However, when the training data is of high quality and nearly noise free, it is desirable that KRR reproduces the reference values exactly and no regularization should be used.

The application of RKHS theory in the context of computational chemistry is straightforward: Recall that potential energy surfaces (PESs) are nothing else than functions which return the energy of a chemical system given the position of its constituting nuclei. As such, provided that a high quality dataset of reference energies obtained by solving the Schrödinger equation (SE) is available, Eq. 3.10 can be used to model the corresponding PES.[37,39] For an example, consider the one-dimensional PES shown in Figure 3.3, for which only a few training examples are needed to construct an accurate model with KRR.

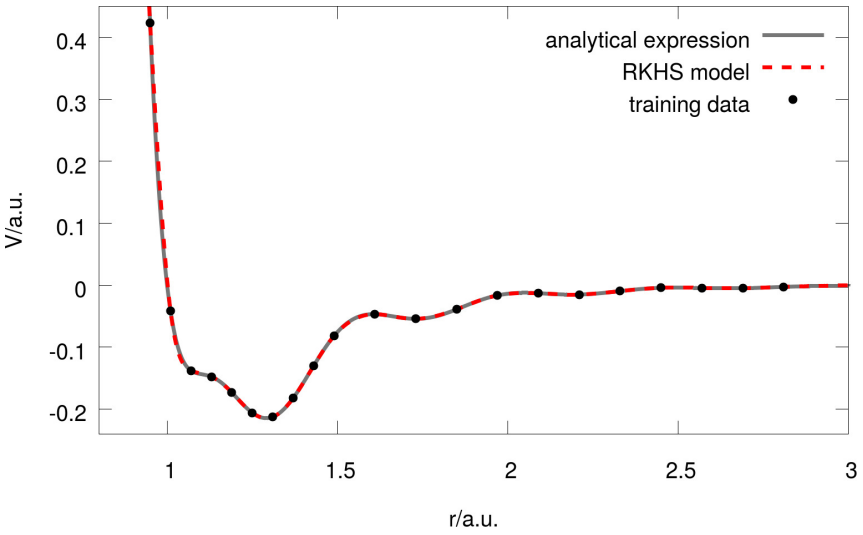


FIGURE 3.3: KRR applied to the function $V(r) = \left(\frac{1}{r^9} - \frac{1}{r^6}\right)\left(\cos^2(7r) + 1\right)$. The RKHS model function $\widetilde{f}(r)$ (red dashed line, see Eq. 3.10) constructed from the training samples (black dots) is virtually identical to the analytical expression (grey line).

For the application of a PES in molecular dynamics (MD) simulations, forces are typically required. Fortunately, derivatives of $\widetilde{f}(\mathbf{x})$ of any order can be calculated analytically by replacing the kernel function $K(\mathbf{x},\mathbf{x}')$ in Eq. 3.10 with its corresponding derivative.[b]

[a]The Kronecker delta is $\delta_{ij} = 1$, when $i = j$ and $\delta_{ij} = 0$, when $i \neq j$. It is named after mathematician Leopold Kronecker (1823–1891).

[b]The sum rule of differentiation states that the derivative of a sum of functions is the sum of derivatives of these functions.

An appealing feature of KRR is that the model function $\widetilde{f}(\mathbf{x})$ (Eq. 3.10) can be systematically improved by using more reference data. Unfortunately, the RKHS method has two major drawbacks when the training set size N becomes large:

1. Since the Cholesky decomposition used to solve Eq. 3.12 scales with $\mathcal{O}(N^3)$, the calculation of the coefficients c_i can become prohibitively time consuming for large training sets. However, since the coefficients need to be only calculated once, this is only problematic for extremely large N ($\gg 10^5$).

2. A more severe drawback is that the evaluation of the model function $\widetilde{f}(\mathbf{x})$ requires a sum over all training samples and thus scales as $\mathcal{O}(N)$ (see Eq. 3.10). If only few evaluations of Eq. 3.10 are required, linear scaling is acceptable. However, for an RKHS representation of a PES used in MD simulations, many thousands of evaluations are typically needed during the course of the dynamics and the linear scaling directly translates to a linear increase in simulation time.

Fortunately, there exist methods to resolve both drawbacks,[38] provided that the training data is arranged in a regular grid, which is often the case when calculating *ab initio* reference data for small systems.[a] However, the algorithms can be difficult to implement efficiently in code and previous implementations have been worked out only for specific examples.[38] For this reason, the RKHS toolkit was developed, which implements algorithms for general cases and largely automates the process of PES construction. It is freely available from `https://github.com/MeuwlyGroup/RKHS`, is published under the MIT license and can be downloaded, modified, and used in other projects for free.

In the following sections, details on fast methods for calculating the expansion coefficients c_i and evaluating the model function $\widetilde{f}(\mathbf{x})$ are given and corresponding algorithms are provided in pseudocode. Further, it is described how the methods can be adapted to work with reference data which is not arranged in a regular grid. Next, an overview of important one-dimensional kernel functions with useful properties for PES construction is given. With these functions, it is possible to construct kernel functions optimized for a particular problem according to Eq. 3.9. Finally, the RKHS toolkit is applied to a high-dimensional toy problem with large reference dataset, for which standard KRR (see Eqs. 3.10 and 3.12) is infeasible.

[a] For example, consider a chemical system consisting of three atoms, for which the configuration is fully specified by three internal coordinates. Often, reference data is obtained by performing *ab initio* calculations for all possible combinations of a set of values for each of these coordinates systematically. The resulting dataset has a regular grid structure.

3.2 Overview of fast RKHS algorithms

3.2.1 Fast calculation of the coefficients

If the training set is constructed by scanning through combinations of sets of $N^{(d)}$ points in each dimension, the resulting data automatically possesses a multi-dimensional grid structure. Note that $N^{(d)}$ can have a different values for each dimension d. The total size N of the training set in this case is simply

$$N = \prod_{d=1}^{D} N^{(d)} \tag{3.14}$$

Further, if the multi-dimensional kernel $K(\mathbf{x},\mathbf{x}')$ can be written as a product of one-dimensional kernels (Eq. 3.9) the kernel matrix $\mathbf{K}$ (see Eq. 3.12) can be expressed as a tensor product of $N^{(d)} \times N^{(d)}$ matrices $\mathbf{k}^{(d)}$

$$\mathbf{K} = \mathbf{k}^{(1)} \otimes \mathbf{k}^{(2)} \otimes \cdots \otimes \mathbf{k}^{(D)} \tag{3.15}$$

where the entries of the matrices $\mathbf{k}^{(d)}$ are given by $k_{ij}^{(d)} = k^{(d)}(x_i^{(d)}, x_j^{(d)})$. Due to the properties of tensor products it is then possible to calculate the coefficients c_i from the Cholesky decompositions of the smaller $N^{(d)} \times N^{(d)}$ matrices $\mathbf{k}^{(d)}$.[105] Since typically, $N^{(d)} \ll N$, the cost of these decompositions is negligible and the coefficients c_i can be computed at essentially the cost of a matrix vector multiplication.[105]

The full procedure for the fast calculation of coefficients is given in pseudocode in Algorithm 3.2. Here, $\mathbf{l}^{(d)}$ is the lower and $\mathbf{u}^{(d)}$ the upper triangular matrix of the Cholesky decomposition $\mathbf{k}^{(d)} = \mathbf{l}^{(d)}\mathbf{u}^{(d)}$. The vectors $\mathbf{z}_{\text{out}}$ and $\mathbf{z}_{\text{in}}$ store intermediate results. For vectors, $x^{(i)}$ is the ith component of vector $\mathbf{x}$ and for matrices, M_{ij} is the entry at position (i,j) of matrix $\mathbf{M}$. Note that matrix equations other than Eq. 3.12 can be solved with this algorithm as well, provided that the matrix has tensor product form and the decomposition of the one-dimensional matrices into lower and upper triangular matrices is possible. Another useful application is for example the calculation of (parts of) the inverse matrix of $\mathbf{K}$, as is needed in Eq. 3.27 (see section 3.3).

3.2.2 Fast evaluation of the model function

The methods described in this section were originally developed and derived by Hollebeek *et al.* and are given in a condensed form for completeness. However, they were never implemented in code for a general case. For further details on their derivation, refer to [38].

As was mentioned earlier, usually it is necessary to sum over all N training samples for a single evaluation of the model function $\widetilde{f}(\mathbf{x})$ (see Eq. 3.10). For particular

Algorithm 3.2 Fast calculation of coefficients for the kernel matrix $\mathbf{K}$ in tensor product form ($\mathbf{K} = \mathbf{k}^{(1)} \otimes \mathbf{k}^{(2)} \otimes \cdots \otimes \mathbf{k}^{(D)}$). The coefficients c_i for data points i need to be initialized to the reference value y_i before the algorithm starts. Pseudocode for the procedures FORWARD_SUBSTITUTION and BACKWARD_SUBSTITUTION are given in Algorithm 3.3.

$n_{\text{left}} = 1$ ▷ begin forward substitution part of algorithm
$n_{\text{right}} = N$
for $d = 1 \dots D$ **do** ▷ loop over dimensions
  $n_{\text{right}} = n_{\text{right}} / N^{(d)}$ ▷ integer division
  $base = 0$
  for $k = 1 \dots n_{\text{left}}$ **do**
    for $j = 1 \dots n_{\text{right}}$ **do**
      $i = base + j$
      for $l = 1 \dots N^{(d)}$ **do**
        $z_{\text{in}}^{(l)} = c_i$
        $i = i + n_{\text{right}}$
      end for
      FORWARD_SUBSTITUTION($\mathbf{l}^{(d)}$, $\mathbf{z}_{\text{out}}$, $\mathbf{z}_{\text{in}}$)
      $i = base + j$
      for $l = 1 \dots N^{(d)}$ **do**
        $c_i = z_{\text{out}}^{(l)}$
        $i = i + n_{\text{right}}$
      end for
    end for
    $base = base + n_{\text{right}} \times N^{(d)}$
  end for
  $n_{\text{left}} = n_{\text{left}} \times N^{(d)}$
end for

$n_{\text{left}} = 1$ ▷ begin backward substitution part of algorithm
$n_{\text{right}} = N$
for $d = 1 \dots D$ **do** ▷ loop over dimensions
  $n_{\text{right}} = n_{\text{right}} / N^{(d)}$
  $base = 0$
  for $k = 1 \dots n_{\text{left}}$ **do**
    for $j = 1 \dots n_{\text{right}}$ **do**
      $i = base + j$
      for $l = 1 \dots N^{(d)}$ **do**
        $z_{\text{in}}^{(l)} = c_i$
        $i = i + n_{\text{right}}$
      end for
      BACKWARD_SUBSTITUTION($\mathbf{u}^{(d)}$, $\mathbf{z}_{\text{out}}$, $\mathbf{z}_{\text{in}}$)
      $i = base + j$
      for $l = 1 \dots N^{(d)}$ **do**
        $c_i = z_{\text{out}}^{(l)}$
        $i = i + n_{\text{right}}$
      end for
    end for
    $base = base + n_{\text{right}} \times N^{(d)}$
  end for
  $n_{\text{left}} = n_{\text{left}} \times N^{(d)}$
end for

Algorithm 3.3 FORWARD_SUBSTITUTION and BACKWARD_SUBSTITUTION procedures referenced in Algorithm 3.2.

```
1: function FORWARD_SUBSTITUTION(L, y, b)
2:     y^(1) = b^(1)/L_11
3:     for i = 2...n do                    ▷ L is a n×n matrix and y and b are n-vectors
4:         s = 0
5:         for k = 1...i−1 do
6:             s = s + L_ik y^(k)
7:         end for
8:         y^(i) = (b^(i) − s)/L_ii
9:     end for
10: end function
11:
12: function BACKWARD_SUBSTITUTION(U, x, y)
13:     x^(n) = y^(n)/U_nn                  ▷ U is a n×n matrix and x and y are n-vectors
14:     for i = n−1...1 do
15:         s = 0
16:         for k = i+1...n do
17:             s = s + U_ik x^(k)
18:         end for
19:         x^(i) = (y^(i) − s)/U_ii
20:     end for
21: end function
```

classes of one-dimensional kernel functions however, it is possible to decompose the function $k(x,x')$ into different contributions of x and x', i.e.

$$k(x,x') = \sum_{k=1}^{\widetilde{M}_1} p_{1k} f_{1k}(x) f_{1k}(x') + \sum_{k=1}^{\widetilde{M}_2} p_{2k} f_{2k}(x_<) f_{3k}(x_>) \tag{3.16}$$

where $x_>$ and $x_<$ are the larger and smaller of x and x' respectively. It should be noted that such decompositions are not necessarily unique. Further, $\widetilde{M}_1$ and $\widetilde{M}_2$ are independent of N and only depend on the chosen kernel function. For simplicity, Eq. 3.16 can equivalently be written as

$$k(x,x') = \sum_{k=1}^{M_2} p_{2k} f_{2k}(x_<) f_{3k}(x_>) \tag{3.17}$$

if all $p_{1k} f_{1k}(x) f_{1k}(x')$ terms are rewritten as $p_{2k} f_{2k}(x_<) f_{3k}(x_>)$ with $f_{1k} = f_{2k} = f_{3k}$ and $p_{1k} = p_{2k}$. Note that M_2 in Eq. 3.17 is equal to $\widetilde{M}_1 + \widetilde{M}_2$ and lies typically

between 2 and 5 (see appendix B). For one dimension, inserting Eq. 3.17 into Eq. 3.10 and reversing the order of summation leads to:

$$\begin{aligned} \widetilde{f}(x) &= \sum_{k=1}^{M_2} \left[f_{3k}(x)\sigma_{2k}(z) + f_{2k}(x)\sigma_{3k}(z) \right] \\ \sigma_{2k}(z) &= p_{2k} \sum_{i \le z} c_i f_{2k}(x_i) \\ \sigma_{3k}(z) &= p_{2k} \sum_{i > z} c_i f_{3k}(x_i) \end{aligned} \tag{3.18}$$

where z is an index selecting the appropriate values from the training set such that $x_z \le x < x_{z+1}$. In this form, it is only necessary to sum over the M_2 terms of the kernel decomposition (Eq. 3.17) instead of all N training samples for an evaluation of $\widetilde{f}(x)$. Since usually, $M_2 \ll N$, this can result in a substantial speedup. The values of σ_{mk} are independent of x and can be precomputed and stored in a lookup table for all possible indices z. The correct index z corresponding to $\sigma_{mk}(z)$ for an arbitrary value of x can be efficiently found by binary search[106] or other search algorithms.

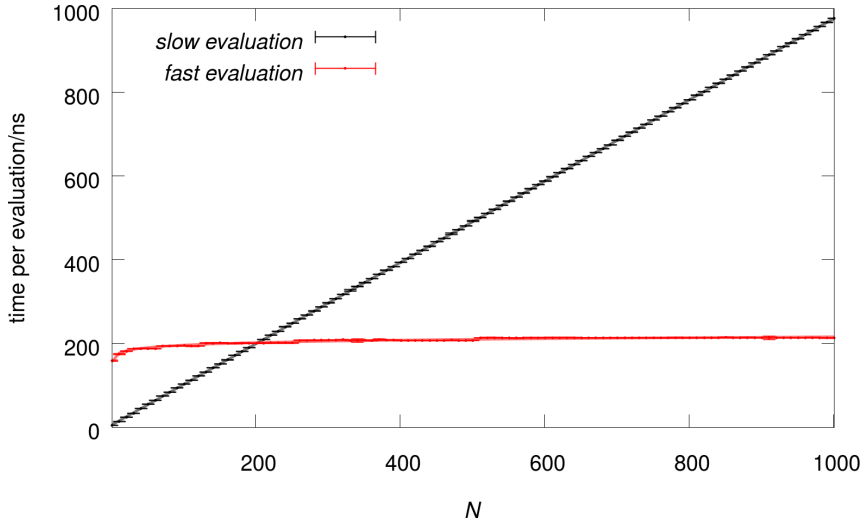


FIGURE 3.4: Average time per evaluation of the model function $\widetilde{f}(\mathbf{x})$ (black: ordinary method, red: fast method) in nanoseconds versus dataset size N. Timings were averaged over 100 sets of 100 000 evaluations per data point (error bars show one standard deviation).

Figure 3.4 compares the speed of ordinary and fast evaluation methods. As expected, the fast evaluation method scales with $O(\log N)$ (complexity of binary search), whereas the ordinary evaluation method scales with $O(N)$. Note that due to the overhead of the binary search algorithm and the cost of memory lookup, the ordinary evaluation method is actually faster up until $N \approx 200$. It is therefore recommended to benchmark the two methods in time-critical applications. The benchmark here was performed on a Desktop computer equipped with an Intel® Xeon® Processor

E3-1275 at 3.40 GHz. In principle, it is possible to further reduce the complexity of the fast evaluation method to $O(1)$ if the values of σ_{mk} are stored in a hash table.[107]

For multiple dimensions, equations similar to Eq. 3.18 can be found by applying the same reasoning recursively to each dimension, which leads to Eqs. 3.19–3.23 for the general, multi-dimensional case.

$$\widetilde{f}(\mathbf{x}) = \sum_{k=1}^{M_2^{(1)}} \left[f_{3k}^{(1)}(x^{(1)})\gamma_{2k} + f_{2k}^{(1)}(x^{(1)})\gamma_{3k} \right] \tag{3.19}$$

$$\begin{aligned}\gamma_{m^{(1)}k^{(1)}\ldots m^{(d)}k^{(d)}} = \sum_{k=1}^{M_2^{(d+1)}} \Big[& f_{3k}^{(d+1)}(x^{(d+1)})\gamma_{m^{(1)}k^{(1)}\ldots m^{(d)}k^{(d)}2k^{(d+1)}} \\ & + f_{2k}^{(d+1)}(x^{(d+1)})\gamma_{m^{(1)}k^{(1)}\ldots m^{(d)}k^{(d)}3k^{(d+1)}} \Big]\end{aligned} \tag{3.20}$$

$$\gamma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}} = \sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z}) \tag{3.21}$$

$$\begin{aligned}&\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z}) = \\ &\sum_{i^{(1)}\in s^{(1)}(m^{(1)})} \cdots \sum_{i^{(D)}\in s^{(D)}(m^{(D)})} c_{i^{(1)}i^{(2)}\ldots i^{(D)}} \times \prod_{d=1}^{D} p_{2k^{(d)}}^{(d)} f_{m^{(d)}k^{(d)}}^{(d)}(x_{i^{(d)}}^{(d)})\end{aligned} \tag{3.22}$$

$$s^{(d)}(m^{(d)}) = \begin{cases} \left\{1, 2, \ldots, z^{(d)}\right\} & m^{(d)} = 2 \\ \left\{z^{(d)}+1, z^{(d)}+2, \ldots, N^{(d)}\right\} & m^{(d)} = 3 \end{cases} \tag{3.23}$$

Here the indices $\mathbf{z} = [z^{(1)} \ \cdots \ z^{(D)}]^\mathsf{T}$ are chosen such that $x_{z^{(d)}}^{(d)} \leq x^{(d)} < x_{z^{(d)}+1}^{(d)}$, and $f_{mk}^{(d)}$ and $p_{2k}^{(d)}$ refer to the functions and constants in the decomposition of the one-dimensional kernel functions $k^{(d)}(x^{(d)}, x'^{(d)})$ (Eq. 3.17). The multi-index $i^{(1)}i^{(2)}\cdots i^{(D)}$ selects the coefficient c_i that corresponds to the point $\mathbf{x}_i = [x_{i^{(1)}}^{(1)} \ x_{i^{(2)}}^{(2)} \ \cdots \ x_{i^{(D)}}^{(D)}]^\mathsf{T}$ in the training set. The model function $\widetilde{f}(\mathbf{x})$ can then be evaluated by iteratively updating the values for γ, starting from $\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z})$, which is taken from a precomputed lookup table (see one-dimensional case, Eq. 3.18). Note that Eq. 3.19 is only applicable if the multi-dimensional kernel $K(\mathbf{x}, \mathbf{x}')$ can be written in the form given by Eq. 3.9. A complete algorithm for the fast evaluation of a multi-dimensional model function $\widetilde{f}(\mathbf{x})$ according to Eqs. 3.19–3.23 is given in pseudocode in Algorithm 3.4.

In Algorithm 3.4, whenever a subscript i is given in mentions of γ_i or σ_i, they refer to a single value that corresponds to the multi-index i, whereas mentions of γ or σ without subscript refer to the complete lookup tables containing the entries for all possible multi-indices. To evaluate an arbitrary partial derivative (of any order) of $\widetilde{f}(\mathbf{x})$, it is sufficient to replace the respective functions $f_{mk}^{(d)}$ (see Eq. 3.16) with the

Algorithm 3.4 Fast evaluation of the model function $\widetilde{f}(\mathbf{x})$ at an arbitrary point $\mathbf{x}$ according to Eqs. 3.19–3.23. The entries $z^{(d)}$ of index vector $\mathbf{z}$ are chosen such that $x^{(d)}_{z^{(d)}} \leq x^{(d)} < x^{(d)}_{z^{(d)}+1}$ for every vector $\mathbf{x}_i$ in the training set. The appropriate $z^{(d)}$ can be easily found using a search algorithm such as binary search.

$\gamma^{\text{new}} = \sigma(\mathbf{z})$ ▷ Initialize values for γ with $\sigma(\mathbf{z})$ from lookup table
$\gamma^{\text{old}} = \gamma^{\text{new}}$

for $d = D-1 \ldots 1$ **do** ▷ This loop should not be executed if $D = 1$
  $\mathbf{k} = 1$ ▷ Initialize vector $\mathbf{k} = [k^{(1)} \ \cdots \ k^{(D)}]^{\mathsf{T}}$ to $\mathbf{k} = [1 \ \cdots \ 1]^{\mathsf{T}}$
  while $k^{(1)} \leq M_2^{(1)}$ **do** ▷ Loop over all possible $\mathbf{k}$
    $\mathbf{m} = 2$ ▷ Initialize vector $\mathbf{m} = [m^{(1)} \ \cdots \ m^{(D)}]^{\mathsf{T}}$ to $\mathbf{m} = [2 \ \cdots \ 2]^{\mathsf{T}}$
    while $m^{(1)} \leq 3$ **do** ▷ Loop over all possible $\mathbf{m}$
      $\gamma^{\text{new}}_{m^{(1)}k^{(1)}\cdots m^{(d)}k^{(d)}} = 0$ ▷ Update γ^{new}
      for $k^{d+1} = 1 \ldots M_2^{d+1}$ **do**
        $\gamma^{\text{new}}_{m^{(1)}k^{(1)}\cdots m^{(d)}k^{(d)}} = \gamma^{\text{new}}_{m^{(1)}k^{(1)}\cdots m^{(d)}k^{(d)}} + \gamma^{\text{old}}_{m^{(1)}k^{(1)}\cdots 2k^{(d+1)}} f^{(d+1)}_{3k^{(d+1)}}(x^{(d+1)})$
        $\gamma^{\text{new}}_{m^{(1)}k^{(1)}\cdots m^{(d)}k^{(d)}} = \gamma^{\text{new}}_{m^{(1)}k^{(1)}\cdots m^{(d)}k^{(d)}} + \gamma^{\text{old}}_{m^{(1)}k^{(1)}\cdots 3k^{(d+1)}} f^{(d+1)}_{2k^{(d+1)}}(x^{(d+1)})$
      end for
      $m^{(d)} = m^{(d)} + 1$ ▷ Makes sure all possible $\mathbf{m}$ are looped through
      for $i = d \ldots 2$ **do**
        if $m^{(i)} > 3$ **then**
          $m^{(i)} = 2$
          $m^{(i-1)} = m^{(i-1)} + 1$
        end if
      end for
    end while
    $k^{(d)} = k^{(d)} + 1$ ▷ Makes sure all possible $\mathbf{k}$ are looped through
    for $i = d \ldots 2$ **do**
      if $k^{(i)} > M_2^{(i)}$ **then**
        $k^{(i)} = 1$
        $k^{(i-1)} = k^{(i-1)} + 1$
      end if
    end for
  end while
  $\gamma^{\text{old}} = \gamma^{\text{new}}$ ▷ Update γ^{old} for next iteration
end for

$f = 0$ ▷ Finally, the value of $\widetilde{f}(\mathbf{x})$ is computed
for $k = 1 \ldots M_2^{(d+1)}$ **do**
  $f = f + \gamma^{\text{new}}_{2k} f^{(1)}_{3k}(x^{(1)})$
  $f = f + \gamma^{\text{new}}_{3k} f^{(1)}_{2k}(x^{(1)})$
end for

corresponding derivatives (the same lookup tables can be reused).

At first glance it appears that the pre-computation of $\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}$ for all possible index vectors $\mathbf{z}$ scales as $O(N^2)$. But in fact, it is possible to calculate all required values in just $O(N)$ time, which is the same complexity as a single evaluation of $\widetilde{f}(\mathbf{x})$ using Eq. 3.10. This is possible because the sums for different $\mathbf{z}$ correspond to sums over different quadrants of a subspace of the full D-dimensional space. Since the quadrants overlap, the value of a specific $\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z})$ can be computed from a single evaluation at $\mathbf{x} = [x_{z^{(1)}} \ \cdots \ x_{z^{(D)}}]^\mathsf{T}$ and other previously computed $\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z}')$. The recurrence relation used to calculate the different $\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z})$ is

$$\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z}) = t(\mathbf{z}) + \sum_{d=1}^{D}(-1)^{d+1}\left[\sum_{\mathbf{i}\in s_d}\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}(\mathbf{z}+\mathbf{i})\right] \tag{3.24}$$

Here, s_d is the set of unique vectors $\mathbf{i}$ containing $D-d$ zeros and d entries with either 1 or -1, depending on the $m^{(d)}$ for the $\sigma_{m^{(1)}k^{(1)}\ldots m^{(D)}k^{(D)}}$ that is currently computed. More precisely, if $m^{(d)} = 2$ then $i^{(d)} = -1$ and if $m^{(d)} = 3$ then $i^{(d)} = 1$. The term $t(\mathbf{z})$ is given by Eq. 3.25

$$t(\mathbf{z}) = c_{z^{(1)}z^{(2)}\ldots z^{(D)}}\prod_{d=1}^{D} p_{m^{(d)}k^{(d)}} f^{(d)}_{m^{(d)}k^{(d)}}(x^{(d)}_{z^{(d)}}) \tag{3.25}$$

Figure 3.5 shows a graphical representation of the recurrence relation for a two-dimensional example. A complete algorithm in pseudocode for the fast pre-calculation of the lookup table σ according to Eq. 3.24 is given in Algorithm 3.5.

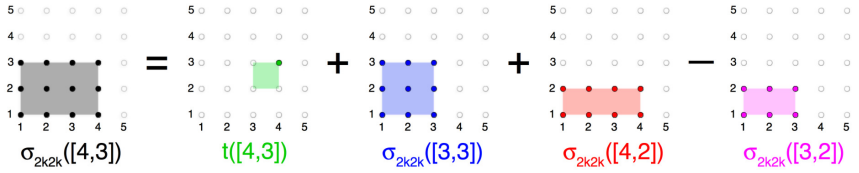


FIGURE 3.5: Graphical representation of the evaluation of Eq. 3.24 for a two-dimensional example. Here, the entry for $z^{(1)} = 4$ and $z^{(2)} = 3$ is calculated for σ_{2k2k}. The subspace shaded in magenta is counted twice and must therefore be subtracted.

In Algorithm 3.5, depending on the values of $m^{(d)}$, care must be taken in what order the different $\mathbf{z}$ are looped through. Otherwise it is possible that $\sigma(\mathbf{z}+\mathbf{i})$ in the innermost loop references an uninitialized value. The first part of the algorithm ensures that this does not happen. Note that $\sigma(\mathbf{z}+\mathbf{i})$ is assumed to be zero whenever $\mathbf{z}+\mathbf{i}$ contains indices which do not exist ($z^{(d)} + i^{(d)} < 1$ or $z^{(d)} + i^{(d)} > N^{(d)}$).

Algorithm 3.5 Fast calculation of the lookup table σ with the recurrence relation given by Eq. 3.24.

for $d = 1 \dots D$ **do** ▷ set $\mathbf{z}_{\text{start}}$, $\mathbf{z}_{\text{end}}$ and $\mathbf{z}_{\text{incr}}$ depending on $m^{(d)}$
  if $m^{(d)} = 2$ **then**
    $z^{(d)}_{\text{start}} = 0$
    $z^{(d)}_{\text{end}} = N^{(d)}$
    $z^{(d)}_{\text{incr}} = 1$
  else if $m^{(d)} = 3$ **then**
    $z^{(d)}_{\text{start}} = N^{(d)}$
    $z^{(d)}_{\text{end}} = 0$
    $z^{(d)}_{\text{incr}} = -1$
  end if
  $z^{(d)} = z^{(d)}_{\text{start}}$
end for

while $z^{(1)} \neq z^{(1)}_{\text{end}} + z^{(1)}_{\text{incr}}$ **do** ▷ loop over all possible $\mathbf{z}$ to evaluate recurrence relation
  $\sigma(\mathbf{z}) = t(\mathbf{z})$ ▷ initialize $\sigma(\mathbf{z})$ to the value at point $[x_{z^{(1)}} \ \cdots \ x_{z^{(D)}}]^{\mathsf{T}}$ (see Eq. 3.25)
  for $d = 1 \dots D$ **do** ▷ calculate outer sum in Eq. 3.24
    $sum = 0$ ▷ calculate inner sum in Eq. 3.24
    for all $\mathbf{i} \in s_d$ **do**
      $sum = sum + \sigma(\mathbf{z} + \mathbf{i})$
    end for
    $\sigma(\mathbf{z}) = \sigma(\mathbf{z}) + (-1)^{d+1} sum$ ▷ update $\sigma(\mathbf{z})$
  end for

  $z^{(D)} = z^{(D)} + z^{(D)}_{\text{incr}}$ ▷ makes sure all possible $\mathbf{z}$ are looped through
  for $d = D \dots 2$ **do**
    if $z^{(d)} = z^{(d)}_{\text{end}} + z^{(d)}_{\text{incr}}$ **then**
      $z^{(d)} = z^{(d)}_{\text{start}}$
      $z^{(d-1)} = z^{(d-1)} + z^{(d-1)}_{\text{incr}}$
    end if
  end for
end while

In short, the fast evaluation method (Eq. 3.19) trades the computational overhead of summing over all training samples (Eq. 3.10) against the storage overhead that is required to store the lookup table of $\sigma_{m^{(1)}k^{(1)}\cdots m^{(D)}k^{(D)}}$ values. The memory requirement in bytes for storing the lookup table is

$$\text{Memory} = 2^D size \prod_{d=1}^{D} M_2^{(d)}(N^{(d)} + 1) \tag{3.26}$$

where $size$ is the number of bytes required to store a single value of $\sigma_{m^{(1)}k^{(1)}\cdots m^{(D)}k^{(D)}}$. Typically, $size$ is either 4 or 8 for single or double precision floating point variables, respectively. Assuming double precision and typical kernel functions ($M_2 = 2$, see appendix B) for a three-dimensional example with an unusually large grid of $N^{(d)} = 100$ in each dimension, the storage requirement is below 500 MB of RAM. This shows that even for extremely large training sets ($N = 10^6$ in this example), the storage overhead is manageable for modern computers.

3.3 Handling incomplete data grids

Although the previous sections showed the advantage of a multi-dimensional grid structure, in practice this strict format of the training set can be either difficult to obtain or suboptimal. For example, consider the case of a reactive PES for the triatomic system ABC. The training data (*ab initio* reference energies) for such a PES must contain points that correspond to the diatom-atom configurations AB–C and AC–B as well as BC–A. Then, unless a special coordinate system is used (e.g. hyperspherical coordinates[108–110]), the grid will automatically contain points that correspond to geometries where all three atoms are either unphysically close ("fusion") or very distant (full atomization). Depending on the problem to be studied, such geometries lie in energetically inaccessible regions of the PES and the time spent performing *ab initio* calculations for such geometries should rather be spent to compute more points in relevant regions. Even when this problem is avoided, some points in the grid might be difficult to converge with the chosen *ab initio* method, or in a more general machine learning (ML) context, data might simply not be available for some points.

Fortunately, such "holes" pose no problem for the ordinary implementation of the RKHS method, i.e. when Eq. 3.10 is used to evaluate the model function and Eq. 3.12 is solved to obtain the coefficients c_i. Both methods require no underlying structure of the training set and work equally well with scattered data. From Eq. 3.10 it is evident that adding additional points to the training set will not influence the result, provided that the coefficients of such points are zero. Thus it is always possible to construct complete grids by filling all holes in the training set with additional points and setting their coefficients to zero. As such, the fast evaluation method discussed in section 3.2.2 becomes applicable even for scattered

data, provided that the data set is padded such that it forms a multi-dimensional grid.

As was pointed out in the introduction (section 3.1), solving Eq. 3.12 with Cholesky decomposition becomes time intensive for large data sets because of the $\mathcal{O}(N^3)$ scaling. In cases where this is relevant, it can be advantageous to complete the grid by assuming arbitrary values for the missing data points and use the fast method outlined in section 3.2.1 to calculate the coefficients instead. Although the coefficients $\widehat{c}_i$ obtained in this way are strictly wrong, the correct coefficients c_i can be calculated from

$$c_i = \widehat{c}_i - \sum_{j \in h} \delta_j \, [\mathbf{K}]^{-1}_{ij} \tag{3.27}$$

where h is the set of all indices that correspond to the holes in the grid, δ_j are correction coefficients and $[\mathbf{K}]^{-1}_{ij}$ is the entry at position (i, j) of the inverse of kernel matrix $\mathbf{K}$ (see Eq. 3.15). For a derivation of Eq. 3.27, refer to [39]. Note that it is not necessary to calculate the full inverse matrix $[\mathbf{K}]^{-1}$, but only the slices that contain holes in the grid. This can be efficiently accomplished using Algorithm 3.2. The correction coefficients δ_j are obtained by solving the matrix equation

$$\begin{bmatrix} Q_{11} & Q_{12} & \dots & Q_{1H} \\ Q_{21} & Q_{22} & \dots & Q_{2H} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{H1} & Q_{H2} & \dots & Q_{HH} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_H \end{bmatrix} = \begin{bmatrix} \widehat{c}_1 \\ \widehat{c}_2 \\ \vdots \\ \widehat{c}_H \end{bmatrix} \tag{3.28}$$

where H is the number of holes in the grid and the matrix $\mathbf{Q} = [Q_{ij}]$ is the submatrix of the full inverse matrix $[\mathbf{K}]^{-1}$ that corresponds to the holes in the grid. The validity of Eq. 3.28 can easily be seen from the requirement that the corrected coefficients c_i that correspond to holes must be zero (see Eq. 3.27). As such, the calculation of the coefficients for incomplete grids scales with $\mathcal{O}(H^3)$ instead of $\mathcal{O}(N^3)$, where usually $H \ll N$. If the matrix $\mathbf{Q}$ has tensor product form itself, which can be the case if a structured subset of the multi-dimensional grid is excluded from the training set, Eq. 3.28 can also be solved using Algorithm 3.2 to avoid $\mathcal{O}(H^3)$ scaling.

3.4 One-dimensional kernel functions

In this section, several one-dimensional kernel functions for different purposes are introduced. The main focus is given to kernels which are useful for constructing multi-dimensional PESs, because they allow to encode physical knowledge, i.e. a specific radial decay behaviour, into the functional form of the PES. However, some kernels useful for general ML applications are also given. All kernel functions introduced here can be decomposed into the form given by Eq. 3.17, thus the fast evaluation method described in section 3.2.2 is applicable if they are used to construct a multi-dimensional kernel $K(\mathbf{x}, \mathbf{x}')$ (see Eq. 3.9).

3.4.1 Kernels for nuclear distances

For the description of nuclear distances in a PES, Hollebeek *et al.* derived two kernel functions that are both defined in the interval $[0,\infty)$, but possess a different asymptotic decay behaviour.[39] Namely, the reciprocal power decay kernel should be chosen whenever a certain interaction in the PES follows a known r^{-n} decay law, whereas the exponential decay kernel should be chosen for short-range contributions to the intermolecular interaction that often decay exponentially to zero for large distances. General formula for both kernels are given below, a derivation can be found in [39].

The reciprocal power decay kernel is given by

$$k_{n,m}(x,x') = n^2 x_>^{-(m+1)}\ \mathrm{B}(m+1,n)\ {}_2\mathrm{F}_1\left(-n+1,m+1;n+m+1;\frac{x_<}{x_>}\right) \tag{3.29}$$

where $x_>$ and $x_<$ are the larger and smaller of x and x', $\mathrm{B}(a,b)$ is the beta function and ${}_2\mathrm{F}_1(a,b;c;d)$ is the Gauss hypergeometric function. The integers n and m determine the smoothness of the kernel function and its asymptotic behaviour. To be precise, $k_{n,m}(x,x')$ has $n-1$ smooth derivatives and decays asymptotically with $x^{-(m+1)}$. Explicit formula for the reciprocal power decay kernel and its decomposition (according to Eq. 3.17) for $n = 2,3$ and $m = 0\ldots6$ are given in appendix B.

The exponential decay kernel is given by

$$k_n(x,x') = \frac{n\cdot n!}{\beta^{2n-1}}\mathrm{e}^{-\beta x_>}\sum_{k=0}^{n-1}\frac{(2n-2-k)!}{(n-1-k)!k!}\left[\beta(x_>-x_<)\right]^k \tag{3.30}$$

where $x_>$ and $x_<$ are the larger and smaller of x and x' and the integer n determines the smoothness. Like the reciprocal power decay kernel (Eq. 3.29), the kernel function $k_n(x,x')$ has $n-1$ smooth derivatives. The positive parameter β determines the asymptotic decay behaviour $\mathrm{e}^{-\beta x}$ and can be chosen such as to be conform with physical observation. Explicit formula for the exponential decay kernel and its decomposition (Eq. 3.17) for $n = 2,3$ are given in appendix B.

Choosing a non-optimal kernel function (e.g. with a wrong asymptotic decay behaviour) is largely inconsequential, provided that the training set contains enough samples in the asymptotic regions (large r). However, a correctly chosen kernel function allows the use of much smaller training sets, which directly translates to less CPU time spent for performing the *ab initio* reference calculations needed to obtain the training data. This is demonstrated in Figure 3.6, where the RKHS method is applied to a two-dimensional model PES corresponding to a charge interacting with a point dipole and a repulsive term. When the reciprocal power decay kernel with

$m = 1$ is used for r, the true PES is well reproduced in the asymptotic regions, even though no training data is present there.

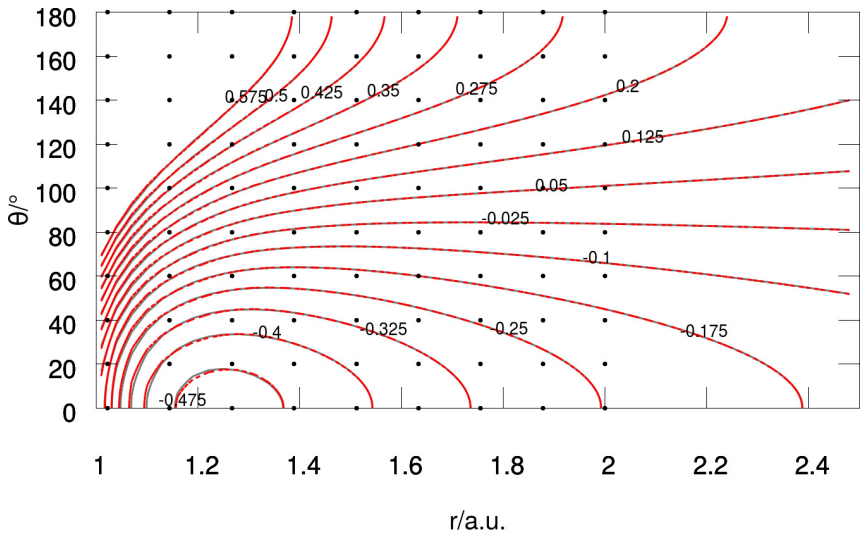


FIGURE 3.6: RKHS method applied to a two-dimensional toy potential given by $V(r,\theta) = \frac{1}{r^9} - \frac{cos(\theta)}{r^2}$. The RKHS model function (red dashed contours) is able to reproduce the true PES (solid grey contours) even in regions where no training data (black dots) is available. Contour lines are drawn in 0.075 increments starting from $V(r,\theta) = -0.475$.

3.4.2 Kernels for angular coordinates and arbitrary functions

The Taylor spline kernel is defined on the interval $[0,1]$ and can be used to model any finite interval, provided that coordinates are transformed through appropriate scaling functions first.[39] For example, for handling an angular coordinate α that is given in the interval $[0,\pi]$, the new coordinate $y \in [0,1]$

$$y(\alpha) = \frac{1 - \cos(\alpha)}{2} \tag{3.31}$$

could be introduced. Similar transformations can be found for any other finite interval, such that the Taylor spline kernel is applicable also for general ML applications. It is given by

$$k_n(x,x') = \sum_{i=0}^{n-1} x_>^i x_<^i + n x_<^n x_>^{n-1} \, {}_2\mathrm{F}_1\left(1, -n+1; n+1; \frac{x_<}{x_>}\right) \tag{3.32}$$

where $x_>$ and $x_<$ are the larger and smaller of x and x' and ${}_2\mathrm{F}_1(a,b;c;d)$ is the Gauss hypergeometric function. Similar to the previously introduced kernel functions, the Taylor spline kernel has $n-1$ smooth derivatives. Explicit formula for the Taylor

spline kernel and its decomposition (according to Eq. 3.17) for $n = 2,3$ are given in appendix B.

3.4.3 Kernels for periodic functions

The periodic spline kernel[111] is defined on the interval $[0,1)$ and can be used to model periodic functions,[112] where it is assumed that a single period of the function occurs between 0 and 1. Similar to the Taylor spline kernel described earlier, this kernel is applicable to any period length, provided that the coordinates are transformed through appropriate scaling functions first. The periodic spline kernel is given by

$$k_n(x,x') = (-1)^{n-1}\frac{B_{2n}(x_> - x_<)}{(2n)!} \tag{3.33}$$

where $x_>$ and $x_<$ are the larger and smaller of x and x' and B_{2n} is the $2n$th Bernoulli polynomial given by

$$B_n(x) = \sum_{k=0}^{n}\binom{n}{k}b_{n-k}x^k \tag{3.34}$$

where b_k are the Bernoulli numbers.[113] Similar to the previously introduced kernel functions, the periodic spline kernel has $n-1$ smooth derivatives.

3.4.4 Radial basis function kernels

Radial basis function (RBF) kernels, such as the Gaussian kernel (Eq. 3.8), are a popular choice for ML applications.[41] They directly take multi-dimensional vectors $\mathbf{x}$ and $\mathbf{x}'$ as input and need not be constructed from one-dimensional kernels according to Eq. 3.9. For the fast evaluation method (see section 3.2.2) to become applicable however, it is not only necessary to express $K(\mathbf{x},\mathbf{x}')$ as a product over one-dimensional kernels (Eq. 3.9), but these need to be decomposable into the form given by Eq. 3.17 as well. Unfortunately, this is not generally possible for RBF kernels.

However, if certain restrictions on $\mathbf{x}$ and $\mathbf{x}'$ are imposed, the Laplacian kernel is an exception to this rule. It is given by

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\gamma\|\mathbf{x}-\mathbf{x}'\|\right) \tag{3.35}$$

where γ is a hyperparameter which determines the width of the kernel. It can equivalently be written in product form (Eq. 3.9) with the one-dimensional kernel function

$$k(x,x') = \exp\left(-\gamma\|x-x'\|\right) \tag{3.36}$$

where x and x' are one of the D components of the D-dimensional input vectors $\mathbf{x}$ and $\mathbf{x}'$. Restricting x and x' to the interval $[0,\infty)$, Eq. 3.36 can be rewritten as

$$k(x,x') = \exp\left(-\gamma x_{>}\right)\exp\left(\gamma x_{<}\right) \tag{3.37}$$

where $x_{>}$ and $x_{<}$ are the larger and smaller of x and x', respectively. This is the form (Eq. 3.17) required for the fast evaluation method. In practice, the restriction of x and x' to positive values should not limit the applicability of the Laplacian kernel, as positive inputs can be obtained by a coordinate transformation, i.e. the introduction of a new coordinate $y = x + C$ where C is a sufficiently large positive constant.

3.5 Application to large datasets

To demonstrate the general applicability of the RKHS toolkit to higher-dimensional problems and large data sets, an oscillatory six-dimensional test function

$$f(\mathbf{x}) = \sum_{d=2}^{6} \exp\left(-x^{(d)}x^{(d-1)}\right)\cos\left(x^{(d)}\frac{\pi}{0.37}\right) \tag{3.38}$$

with $x^{(d)} \in [0,1]$ was considered and training sets of different size were generated. The grid size $N^{(d)}$ of each dimension was chosen to be equal and training points are equally spaced between $x^{(d)} = 0$ and $x^{(d)} = 1$ in each dimension. A Taylor spline kernel (Eq. 3.32) with $n = 2$ was chosen as the kernel function for each dimension. Note that both, equal grid size and equally spaced training points are not required by the fast RKHS methods and merely chosen for simplicity. The RKHS toolkit was applied to data sets from $N^{(d)} = 3$ to $N^{(d)} = 14$ (which corresponds to a total training set size of $N = 729$ to $N = 7\,529\,536$, see Eq. 3.14). Figure 3.7 shows the systematic improvement of the MSE of the RKHS model with increasing N and the correlation of the RKHS model with the analytical expression (Eq. 3.38).

It is worthwhile to point out that the direct solution of Eq. 3.12 for the calculation of the coefficients c_i with Cholesky decomposition is infeasible for the largest dataset ($N = 7\,529\,536$) with modern computers. Assuming that the Cholesky decomposition of a $10\,000 \times 10\,000$ matrix takes one second, the decomposition of a $7\,529\,536 \times 7\,529\,536$ matrix would take well over ten years. With the fast method (see section 3.2.1), the calculation of the coefficients takes at most a few minutes.

The interpolation capabilities of the RKHS method were also tested for high-dimensional model PESs and similar results were obtained. For example, an RKHS model for the multi-dimensional Morse potential given by

$$f(\mathbf{x}) = \sum_{d=1}^{6} D_e^{(d)}\left(1 - \exp\left(-a^{(d)}(x^{(d)} - x_e^{(d)})\right)^2 - 1\right) \tag{3.39}$$

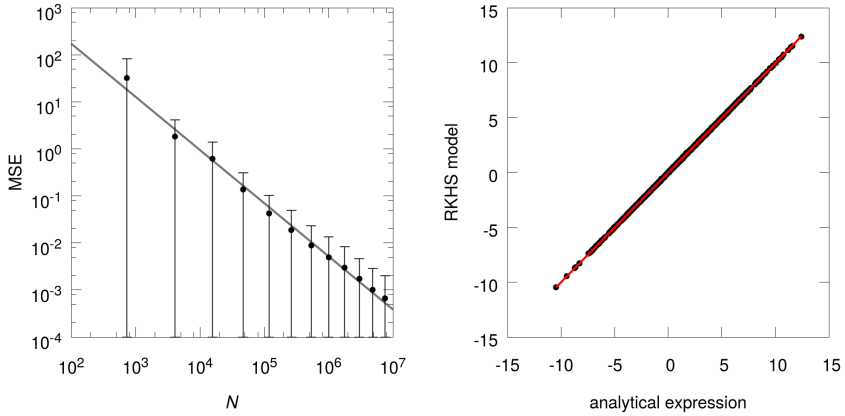


FIGURE 3.7: **Left**: MSE of the RKHS model function compared to the true function (Eq. 3.38) versus training set size N. Data points were generated by averaging the squared error of 1000 randomly chosen points (error bars show one standard deviation). **Right**: Correlation of the RKHS model ($N = 7\,529\,536$) with the analytical expression (Eq. 3.38) for 1000 randomly chosen points. The red line depicts a perfect correlation and the MSE is 6.7×10^{-4}.

with $D_e^{(d)} = a^{(d)} = x_e^{(d)} = 1$ was constructed from a regular grid of 10 points in each dimension ($N = 1\,000\,000$) between $x^{(d)} = 0.1$ and $x^{(d)} = 3$. The exponential decay kernel was chosen as kernel function for each dimension and the MSE of the RKHS model for 1000 randomly chosen points was 2.9×10^{-3}.

3.6 Conclusion

The RKHS toolkit is designed for concrete applications of the fast RKHS methods to problems arising in computational chemistry. While focus is given to the construction of PESs, the range of applications is considerably larger. The toolkit implements efficient algorithms and automatically handles incomplete grids. Further, a wide range of one-dimensional kernel functions are implemented, such that it is possible to construct a PES with a specially designed kernel with just a few function calls. The code can be easily extended if other kernel functions are required. New PESs for arbitrary systems can be readily constructed and do not require manual fitting of parameters or other human input, provided that training data is available. For example, the RKHS method has been applied to construct a PES for the N_2^+–Ar system.[114,115]

The RKHS toolkit supports any number of dimensions and can also be used for general ML applications, e.g. for the interpolation of D-dimensional functions. It allows analytical calculation of all first and second order partial derivatives (higher-order derivatives can be easily implemented in case they are needed). The RKHS

toolkit is primarily intended for constructing PESs for small systems consisting of a few atoms, for which systematically constructing reference data in grid form is feasible. For medium-sized systems, alternative approaches also based on KRR, such as gradient-domain machine learning (GDML)[116] or its symmetrized variant[117,118] have recently become available.

Acknowledgements

I would like to thank Prof. Dr. Hershel Rabitz and Dr. Tak-San Ho for interesting discussions about their previous work, which formed the basis for the RKHS toolkit.

Chapter 4

Neural network-based potential energy surfaces

This chapter starts with a brief summary of the history of artificial neural networks (NNs) and their biological motivation (section 4.1). Further, important concepts and the mathematics behind NNs are explained in more detail and possible ways to construct potential energy surfaces (PESs) with NNs are outlined. Two variants of the so-called high-dimensional neural network (HDNN) approach are described: In section 4.2, a descriptor-based HDNN is introduced, which was previously published in [119]. A message-passing HDNN (previously published in [120]) is introduced in section 4.3. Both variants are applied to various benchmark datasets to assess their performance.

4.1 Introduction

Artificial neural networks (NNs) are, as their name implies, inspired by the networks formed by neurons in the nervous system of animals. A biological neuron consists of a cell body and several filaments protruding from it, which are differentiated into dendrites and an axon. Each neuron has multiple dendrites, which receive electrical impulses from other neurons. These signals increase (excite) or decrease (inhibit) the so-called membrane potential of the neuron.[a] All received signals are integrated in the cell body and, if a certain threshold potential is exceeded, the neuron "fires", i.e. an electrical signal, the so-called action potential, is propagated to other neurons via the axon. The magnitude of the action potential does not depend on the strength of the exciting stimulus, so long as the threshold is reached. This principle is often referred to as "all-or-none law".[121] Even though each neuron has only one axon, it typically branches and connects to the dendrites of many other neurons via so-called synapses, forming intricate networks.[122] For example, the human brain contains about 100 billion neurons,[123] each forming on average 7000 synapses.[124]

[a] The membrane potential is the difference in electric potential between the interior and exterior of a neuron. In the absence of any electrical signals, it lies at a value between −40 mV to −80 mV, which is known as resting potential.[121]

In 1943, McCulloch & Pitts were the first to propose a simplified model for biological neurons: The linear threshold unit (LTU).[125] In this model, each artificial neuron or "unit" is associated with a freely chosen threshold value and receives input from an arbitrary number of excitatory neurons x_i and inhibitory neurons z_i. If the sum of excitatory signals x_i equals or exceeds the threshold value (and no inhibitory signals are received), the neuron outputs 1. Else, as long as the threshold value is not reached or the sum of inhibitory signals z_i is larger than 0, the neuron outputs 0. As such, neurons in this model are limited to binary outputs ("true", 1 or "false", 0). McCulloch & Pitts showed that their proposed artificial neurons can be combined to logic gates implementing simple Boolean functions,[125] like AND, OR and NOT (see Figure 4.1),[a] which at the time were believed to be fundamental ingredients of intelligence. While for simple functions like logic gates, it is possible to manually find an arrangement that produces the desired output, the model proposed by McCulloch & Pitts lacks a mechanism to "learn" more complicated outputs.

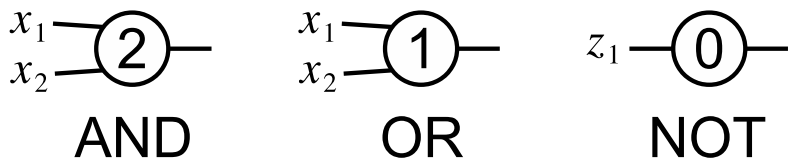


FIGURE 4.1: Implementation of AND, OR and NOT gates with LTUs.[125] The numbers indicate threshold values.

In 1949, Hebb hypothesised that the strength of synaptic connections between biological neurons increases when they are active at the same time ("what fires together, wires together"), which enables biological neural networks to learn.[126] Nearly a decade later, inspired by the Hebbian learning rule, Rosenblatt proposed the perceptron model for neurons,[127] which until today forms the basis of artificial NNs. Similar to the LTU, the perceptron has an arbitrary number of inputs x_i and outputs either 0 or 1, depending on whether its threshold value is reached or not. However, there is no distinction between excitatory and inhibitory inputs. Instead, each input is associated with a real-valued "synaptic weight" w_i and the output y of the perceptron is given by

$$y = \sigma\left(\mathbf{w} \cdot \mathbf{x} + b\right) \tag{4.1}$$

[a] The AND function takes two binary inputs (0 or 1) and is 1 when both of its inputs are 1 and 0 otherwise. The OR function is 1 when either one or both of its inputs are 1 and 0 otherwise. The NOT functions takes a single input and negates it, i.e. it is 1 when its input is 0 and vice versa.

where $\mathbf{w}$ and $\mathbf{x}$ are the vectors of weights w_i and inputs x_i, the operator '$\cdot$' denotes the dot product[a] and the real-valued bias b is used to adjust the threshold for which the Heaviside[b] step function $\sigma(x)$ given by

$$\sigma(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4.2}$$

outputs a value of 1. The function $\sigma(x)$ is also referred to as activation function and is necessary to mimic the "all-or-none" behaviour of biological neurons. To put it simply, the perceptron computes a weighted sum of its inputs and determines whether this sum is larger than its threshold value, in which case it outputs a value of 1, or not, in which case it outputs 0. A schematic representation of a perceptron with two inputs is shown in Figure 4.2.

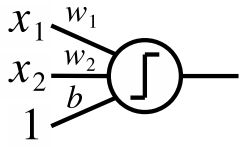


FIGURE 4.2: Schematic representation of a perceptron[127] with two inputs x_1, x_2 and synaptic weights w_1, w_2. The bias b can equivalently be thought of as the synaptic weight of an additional input that is always equal to 1.

.

Rosenblatt proposed the following learning algorithm to automatically find the synaptic weights $\mathbf{w}$ that solve a particular problem:[127] Given a training set $D = \{(o_1; \mathbf{x}_1)\}_{i=1}^{N}$ of N samples, where $\mathbf{x}_i$ is the input vector and o_i the desired output value, perform the following steps:

1. Initialize weights $\mathbf{w}$ and bias b to 0.
2. For each example i in the training set D, perform the following steps:
   (a) Calculate the output y_i of the perceptron according to Eq. 4.1
   (b) Update weights $\mathbf{w}$ and bias b according to

$$\begin{aligned} \mathbf{w}^{t+1} &= \mathbf{w}^t + \alpha\,(o_i - y_i)\,\mathbf{x}_i \\ b^{t+1} &= b^t + \alpha(o_i - y_i) \end{aligned} \tag{4.3}$$

   where α is the learning rate, a hyperparameter that determines how quickly the perceptron learns, and the superscript t denotes the values

[a] The dot product of two vectors $\mathbf{w}$ and $\mathbf{x}$ is given by the sum of their entrywise product, i.e. $\mathbf{w} \cdot \mathbf{x} = \sum_i w_i x_i$.

[b] Named after mathematician Oliver Heaviside (1850–1925).

of parameters at training step t. Note that the magnitude of parameter updates depends on the magnitude of the error, i.e. the difference $(o_i - y_i)$ between desired output o_i and prediction of the perceptron y_i.

3. Terminate the algorithm if the mean absolute error (MAE) given by

$$\mathrm{MAE} = \frac{1}{N}\sum_i |o_i - y_i| \tag{4.4}$$

is smaller than a prespecified error threshold ϵ or a set maximum number of iterations is reached, else go back to step 2.

The learning algorithm described above converges to a solution for all linearly separable[a] functions. Similar to the LTU proposed by McCulloch & Pitts, a perceptron can implement simple Boolean functions like AND, OR and NOT. However, contrary to the LTU, the appropriate parameters can be learned from examples and do not need to be provided manually (see Table 4.1).

TABLE 4.1: Learned parameters of a perceptron implementing the AND, OR and NOT functions for the learning rate $\alpha = 1$, given the indicated training sets. The specified parameters reproduce the desired outputs with zero error.

function	learned parameters	training set
AND	$\mathbf{w} = [1\ 2]$ $b = -3$	$\mathbf{x}_1 = [0\ 0]$ $o_1 = 0$ $\mathbf{x}_2 = [1\ 0]$ $o_2 = 0$ $\mathbf{x}_3 = [0\ 1]$ $o_3 = 0$ $\mathbf{x}_4 = [1\ 1]$ $o_4 = 1$
OR	$\mathbf{w} = [1\ 1]$ $b = -1$	$\mathbf{x}_1 = [0\ 0]$ $o_1 = 0$ $\mathbf{x}_2 = [1\ 0]$ $o_2 = 1$ $\mathbf{x}_3 = [0\ 1]$ $o_3 = 1$ $\mathbf{x}_4 = [1\ 1]$ $o_4 = 1$
NOT	$\mathbf{w} = [-1]$ $b = 0$	$\mathbf{x}_1 = [0]$ $o_1 = 1$ $\mathbf{x}_2 = [1]$ $o_2 = 0$

Shortly after its publication, the perceptron was thought to be the beginning of the development of an intelligent computer.[128] However, the initial enthusiasm came to an abrupt end in 1969, when Minsky & Papert showed that a single perceptron is unable to learn functions that are not linearly separable, for example the XOR function (see Figure 4.3 for an illustration).[b]

[a] A function is linearly separable when a single line can be drawn to separate positive examples, i.e. cases where the function should output 1, from negative examples, i.e. cases where the function should output 0 (see also section 2.1 in chapter 2).

[b] The XOR function takes two binary inputs x_1 and x_2 and outputs 1 when either of its inputs is 1 and the other input is 0. Contrary to the OR function, the XOR function outputs 0 when both of its inputs are 1, as well as when both inputs are 0.

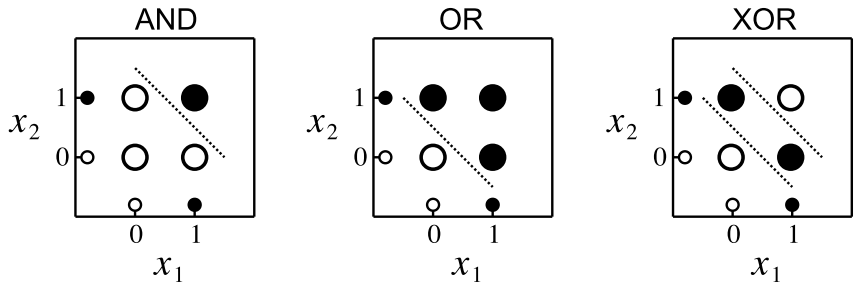


FIGURE 4.3: Schematic representation of AND, OR and XOR functions that highlights their linear (non)separability. White circles represent an output of 0 (false) and black circles an output of 1 (true). For the AND and OR functions, a single line can be drawn to separate the different outputs, whereas for the XOR function, at least two lines are necessary.

Such functions can only be represented by multiple perceptrons stacked on top of each other, a so-called multilayer perceptron (MLP).[129] An example for an MLP that implements the XOR function is shown in Figure 4.4.

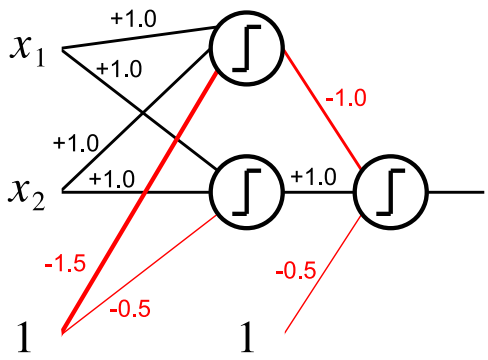


FIGURE 4.4: Schematic representation of an MLP that implements the XOR function. Positive values for weights and biases are given in black and negative values in red.

Unfortunately, the perceptron learning algorithm proposed by Rosenblatt cannot be applied to such MLPs: Since for a given input, only the desired output of the last perceptron in the stack (output layer) is known, the output of the intermediate "hidden" layer of perceptrons cannot be derived from the training data and it is unclear how to update the corresponding parameters. After it was shown that a single perceptron cannot learn simple functions like XOR, interest in artificial NNs diminished.[128]

It took until the mid 1980s until a solution for training MLPs was found with the backpropagation procedure.[130–132] Recall that the learning algorithm proposed by

Rosenblatt updates weights and biases based on the error between prediction of the perceptron and desired output (see Eq. 4.3). The backpropagation procedure is nothing more than an application of the chain rule[a] for computing derivatives: Given an objective function, for example the MAE (see Eq. 4.4), its derivative with respect to parameters in the hidden layer can be computed by working backwards from the gradient with respect to the output layer. The value of the objective function, i.e. the error, can then be minimized by standard methods such as gradient descent.[b] As it were, the error is "backpropagated" from the top to the bottom layer of perceptrons, hence the name backpropagation. All it takes for this procedure to be applicable to MLPs is to replace the binary threshold function (Eq. 4.2), which has a derivative of zero everywhere, with a smooth variant for which a non-zero gradient can be computed. For example, the sigmoid function given by

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.5}$$

is a suitable replacement (see Figure 4.5), but other functions like $\tanh(x)$ are possible as well. Additionally, the weights and bias parameters of the hidden layer should be initialized to small random numbers instead of 0 in order to facilitate the propagation of the error signal.

Note that even when the activation function is changed to Eq. 4.5, the output of an MLP as shown in Figure 4.4 is still a single value bounded between 0 and 1, i.e. it is a binary classifier, which limits possible applications. However, it is straightforward to generalize the underlying principles to allow an arbitrary number of unbounded real-valued outputs for general regression tasks. For this, the single perceptron in the output layer is replaced by as many perceptrons as there are output values and their activation function is chosen to be the identity function $\sigma(x) = x$. The output is then a linear combination of the values in the hidden layer. In a way, this is similar to the "kernel trick" in kernel ridge regression (KRR) (see chapter 3): The input is mapped to a feature space (hidden layer) in which the regression problem becomes linear.

Such a design is also known as feedforward NN. The output $\mathbf{y}$ of a feedforward NN can be written as

$$\begin{aligned} \mathbf{h} &= \sigma(\mathbf{W_1 x} + \mathbf{b_1}) \\ \mathbf{y} &= \mathbf{W_2 h} + \mathbf{b_2} \end{aligned} \tag{4.6}$$

[a] The chain rule states that the derivative $F'(x)$ of a composite function $F(x) = f(g(x))$ is given by $F'(x) = f'(g(x))g'(x)$.

[b] The gradient descent procedure minimizes the value of a function by taking steps proportional to the negative gradient, i.e. the "direction" where the function decreases most. For example, the value of the MAE function (Eq. 4.4) depends on the weights and biases of the perceptron and is thus a function of these parameters. If the gradient of the MAE with respect to weights and biases is known, the parameters can be updated to get new values that produce a lower MAE.

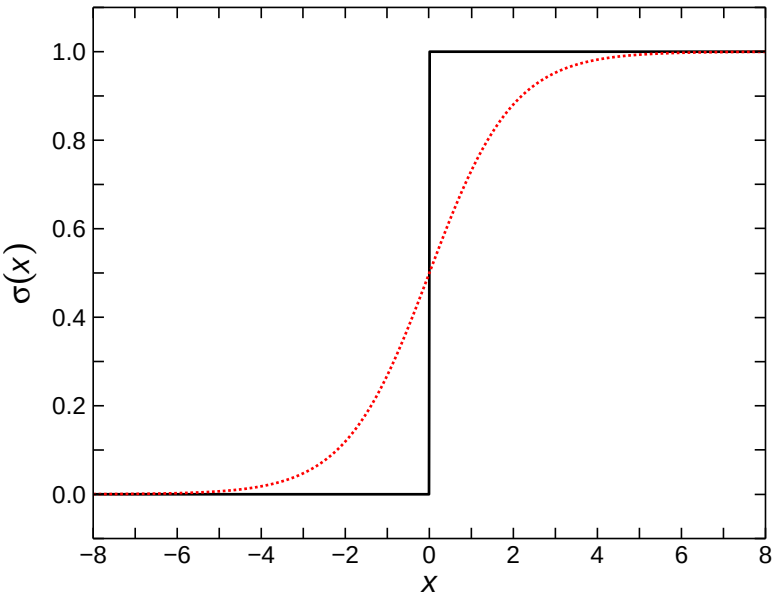


FIGURE 4.5: Comparison between Heaviside step function (solid black, Eq. 4.2) and a smooth approximation, the sigmoid function (dotted red, Eq. 4.5).

or simply

$$\mathbf{y} = \mathbf{W_2}\sigma(\mathbf{W_1}\mathbf{x} + \mathbf{b_1}) + \mathbf{b_2} \tag{4.7}$$

where $\mathbf{Wx}$ denotes matrix multiplication,[a] $\mathbf{x} \in \mathbb{R}^{n_{\text{in}}}$ are n_{in} inputs, $\mathbf{h} \in \mathbb{R}^{n_{\text{hidden}}}$ are the values of n_{hidden} perceptrons in the hidden layer and $\mathbf{y} \in \mathbb{R}^{n_{\text{out}}}$ are the n_{out} output values. The matrices $\mathbf{W_1} \in \mathbb{R}^{n_{\text{hidden}} \times n_{\text{in}}}$, $\mathbf{W_2} \in \mathbb{R}^{n_{\text{out}} \times n_{\text{hidden}}}$ and vectors $\mathbf{b_1} \in \mathbb{R}^{n_{\text{hidden}}}$, $\mathbf{b_2} \in \mathbb{R}^{n_{\text{out}}}$ contain weights and biases, respectively. Here, the notation $\sigma(\mathbf{x})$ means that the function $\sigma(x)$ is applied to the vector $\mathbf{x}$ entrywise. An illustration of a feedforward NN is given in Figure 4.6.

In fact, feedforward NNs with a single hidden layer with suitable activation function (e.g. Eq. 4.5) were proven to be general function approximators, meaning that they can approximate any mapping between inputs $\mathbf{x}$ and outputs $\mathbf{y}$, provided that the hidden layer is "wide" enough (contains sufficiently many neurons).[133–135] However, for some functions, the required number of necessary hidden neurons can grow exponentially with the number of inputs when just a single hidden layer is used.[136] Often, it is more parameter efficient[b] to construct a "deep" NN (see Figure 4.7) by stacking several

[a] The output of the matrix multiplication between a matrix $\mathbf{W}$ and a vector $\mathbf{x}$ can equivalently be thought of as a vector containing the dot products of $\mathbf{x}$ and $\mathbf{w}_i$, where $\mathbf{w}_i$ are the rows of $\mathbf{W}$. Each row $\mathbf{w}_i$ therefore corresponds to the synaptic weights of a single perceptron i (see Eq. 4.1) and the output of the matrix multiplication $\mathbf{Wx}$ is simply a vector of weighted sums of the inputs $\mathbf{x}$ with different weights $\mathbf{w}_i$.

[b] Here parameter efficiency means that in total, a smaller number of parameters is necessary to obtain an NN that approximates the target function equally well.

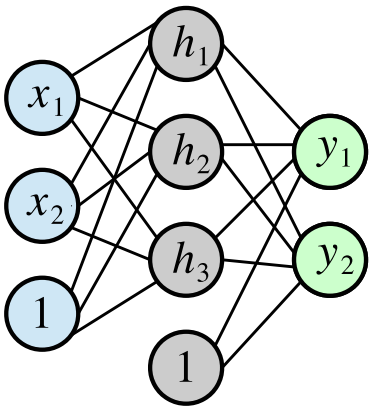


FIGURE 4.6: Schematic representation of a feedforward NN with two inputs $\mathbf{x} = [x_1\ x_2]$ (blue), a single hidden layer with three neurons $\mathbf{h} = [h_1\ h_2\ h_3]$ (grey) and two outputs $\mathbf{y} = [y_1\ y_2]$ (green), see Eqs. 4.6 and 4.7. Weights and biases are represented by black lines connecting different neurons.

"narrow" hidden layers (containing a small number of neurons) on top of each other instead of using a single "wide" hidden layer. Even though (deep) feedforward NNs can in principle represent any function, depending on the problem that needs to be solved, other NN architectures are sometimes more efficient, e.g. convolutional neural networks (CNNs) for image recognition (see [137]) or recurrent neural networks (RNNs) for the analysis of sequential data (see [138]). These alternatives usually exploit the underlying structure of the input data to reduce the number of redundant parameters.

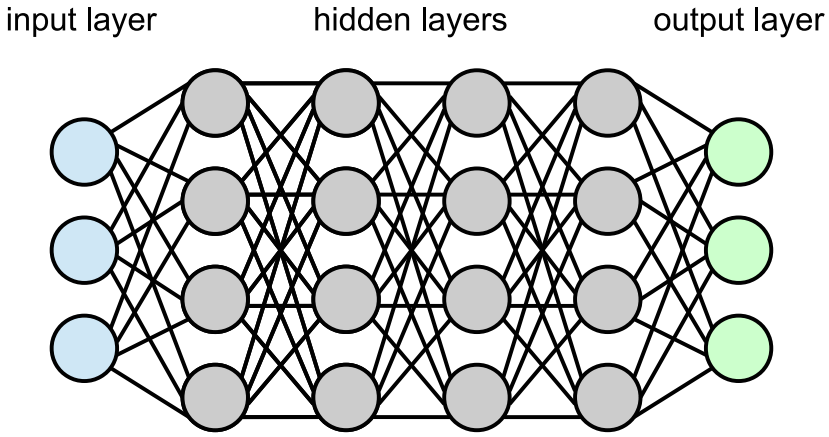


FIGURE 4.7: Schematic representation of a deep feedforward NN. Multiple hidden layers (grey) are stacked on top of the input layer (blue) before reaching the output layer (green).

For a long time, training deep NNs with backpropagation was considered to be difficult in practice because of the vanishing gradient problem:[139] Sigmoidal activation

functions (e.g. Eq. 4.5) saturate for inputs that are large in magnitude and have gradients close to zero in those "plateau regions" (see Figure 4.5). When the chain rule is applied to propagate the error signal during the backpropagation procedure, repeated multiplication of these small numbers causes the error signal to vanish in deep layers, making them difficult to train. This problem can be reduced however by carefully initializing the weight and bias parameters of the NN to prevent saturation of the activation function during the initial training steps.[139] Additionally, sigmoidal activation functions can be replaced with alternatives like the ReLU function[140] (Eq. 4.8) or its smooth approximation, the softplus function[141] (Eq. 4.9), see Figure 4.8.

$$\sigma(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4.8}$$

$$\sigma(x) = \log(e^x + 1) \tag{4.9}$$

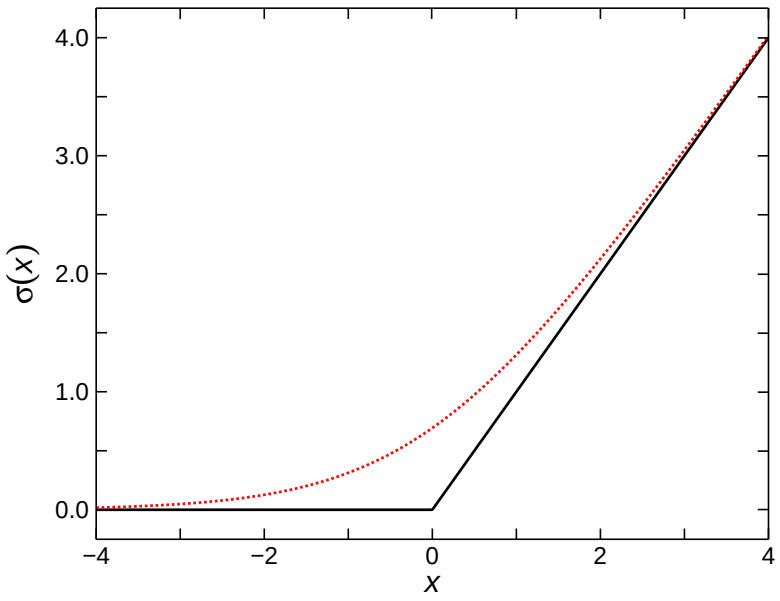


FIGURE 4.8: Comparison between ReLU function (solid black, Eq. 4.8) and its smooth approximation, the softplus function (dotted red, Eq. 4.9).

Non-sigmoidal activation functions like ReLU are nowadays among the most common choices for artificial NNs and often outperform sigmoidal activation functions,[142] even though they violate the "all-or-none" law present in biological neurons. While modern deep NNs may be far removed from their original biological inspiration, they have become powerful tools to tackle many challenging problems, including speech,[143] image[144] and facial[145] recognition, which are often too difficult to be solved with traditional methods.

Artificial NNs can also be applied in computational chemistry: Because they are general function approximators,[133–135] NNs are ideally suited to represent potential energy surfaces (PESs), which are nothing else than functions which return the energy of a chemical system given the position of its constituting nuclei. Using an appropriate dataset of reference energies, e.g. obtained by solving the Schrödinger equation (SE), an NN can be trained via the backpropagation procedure to represent the PES of any system of interest.

Perhaps the most straightforward way to construct a PES with an NN is to use a set of internal coordinates as input to a feedforward NN. As an example, consider a water molecule (H_2O), for which the relative position of its three nuclei can be fully specified by three internuclear distances r_1, r_2 and r_3 (see Figure 4.9).[a]

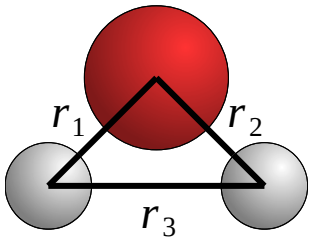


FIGURE 4.9: Sketch of a H_2O molecule (O: red, H: white), which is fully described by three internuclear distances r_1, r_2 and r_3.

While PESs have been constructed successfully in this fashion for many different chemical systems,[146–150] this approach has some disadvantages. For example, in the case of a water molecule, exchanging the two hydrogen atoms effectively swaps the numeric values of r_1 and r_2 (see Figure 4.9). However, there is no guarantee that the output of the NN stays the same after swapping both hydrogen atoms, even though it is known that the energy of a chemical system should be invariant with respect to permutation of equivalent atoms. Another disadvantage is that an NN trained for e.g. a single water molecule cannot necessarily be used to calculate the energy of a water dimer, because they require a different number of inputs.[b] As such, NN-PESs constructed by providing internal coordinates as inputs are not transferable between different systems and a new NN needs to be trained for every new system of interest. Another possible method to construct NN-PESs is to decompose the energy of the system in the spirit of a many-body expansion,[151–154] but such approaches scale poorly to large systems, because they typically involve a large number of individual

[a] Instead of three distances, the geometry could equally be specified by three different numbers. For example, the length of the two OH bonds and the angle between them is another suitable choice.

[b] A chemical system usually has $3N - 6$ internal degrees of freedom, where N is the number of nuclei. Thus, a water dimer would require twelve different inputs, whereas a single water molecule only requires three. This alone makes it difficult to use the same NN for both tasks, at least without artificially "padding" the inputs for single water molecules.

NNs (one for each term in the many-body expansion).

An alternative way to construct NN-based PESs, which circumvents the disadvantages mentioned above, was first proposed in 2007 by Behler & Parrinello for bulk silicon.[155] In these so-called high-dimensional neural networks (HDNNs), the total energy of a chemical system is decomposed into atomic contributions and an NN is used to predict the "atomic energies".[a] The HDNN approach relies on the chemically intuitive assumption that the contribution of an atom to the total energy depends mainly on its local chemical environment. Two designs of HDNN can be distinguished: In the first "descriptor-based" variant,[b] the local environment of an atom is encoded in a handcrafted descriptor vector,[156–158] which is used as input of a standard feedforward NN.[159,160] In the second "message-passing"[161] variant, nuclear charges and Cartesian coordinates are used directly as input and a specially designed deep NN is used to exchange information ("messages") between individual atoms, such that the NN can learn a meaningful representation of their chemical environments from data. This approach was first introduced by Schütt *et al.* in 2017 and has since been used in several other works.[161,163–165]

Since the vector that describes the environment of an atom has a fixed size in both descriptor-based and message-passing HDNNs and is thus independent of system size, the same NN can be applied to chemical systems with any number of atoms by simply evaluating it once for every constituting atom. This allows HDNNs to be transferable between related tasks, e.g. a single HDNN can be trained to describe water molecules, dimers or even bulk water. Further, as long as the atomic environment descriptor is designed to be translationally, rotationally and permutationally invariant, the predictions of the HDNN will automatically share the same invariances, which is desirable for energy predictions of chemical systems: The energy of a chemical system only depends on the relative position of the constituting nuclei. When the whole system is moved or rotated without changing any relative positions, the energy does not change (translational and rotational invariance). Similarly, when two equivalent atoms, say two atoms of the same element, swap their positions, the energy also does not change (permutational invariance).

In the following, both variants of HDNNs are described in more detail. Section 4.2 introduces a descriptor-based HDNN, whereas in section 4.3, a message-passing HDNN is described. Both are applied to various quantum chemical benchmark datasets and their performance is assessed.

[a] Some applications use a separate NN for each element to predict atomic energies.

[b] These types of HDNN are also referred to as Behler-Parrinello networks after their inventors.[155]

4.2 Descriptor-based HDNN

In this segment, a descriptor-based HDNN is presented. Section 4.2.1 describes the prediction process, the descriptor, the NN architecture and the NN training procedure. In section 4.2.2, trends in the prediction of atomic energies are discussed and results on the QM9 dataset[166] and several molecular dynamics (MD) datasets[116] are presented. The results are summarized and potential shortcomings of the descriptor-based HDNN are discussed in section 4.2.3.

4.2.1 Methods

To predict the energy of a system of interest, such as a molecule, a descriptor for each atom is supplied to a feedforward NN, which predicts an atomic energy contribution E_i. The individual contributions are added to obtain the total energy E_{tot}. The computational protocol is summarized below and schematically represented in Figure 4.10:

(a) The local atomic environment of every atom i, consists of its element (e.g. C, H, O, ...) and information about the relative positions $\mathbf{r}_j$ and nuclear charges Z_j of all neighbouring atoms j inside the cut-off sphere (indicated by a red circle in Figure 4.10).

(b) This information is encoded in a fixed size numeric descriptor vector. Since the descriptor is constructed to be rotationally, translationally and permutationally invariant, all symmetry equivalent atoms are encoded in the same way.

(c) The descriptor vector is supplied to a feedforward NN.

(d) The NN outputs an atomic energy contribution E_i.

(e) Finally, the individual contributions are accumulated to give $E_{\text{tot}} = \sum_i E_i$. Since addition is commutative, E_{tot} is automatically invariant with respect to atom permutations.

In the following, the atomic descriptor (section 4.2.1.1), the NN architecture (section 4.2.1.2) and the process for training the NN (section 4.2.1.3) are described in more detail. It is important to note that only total energies are required as reference data during training, as the NN automatically learns to perform the energy decomposition into atomic contributions. This way, only true quantum mechanical observables are used as reference data and no, ultimately arbitrary, energy decomposition scheme[167–169] needs to be imposed.

4.2.1.1 Atomic descriptor

Individual atoms and their local environment are represented by a descriptor, which needs to encode all information relevant to predicting its atomic energy contribution

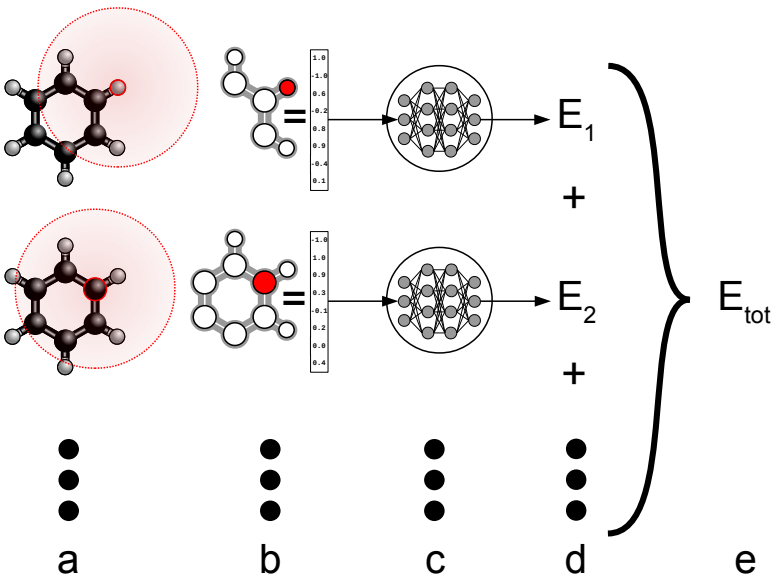


FIGURE 4.10: Schematic representation of predicting the energy E_{tot} of a chemical system (here benzene C_6H_6) with a descriptor-based HDNN.

(relative positions and species of neighbouring atoms). Further, due to the way feedforward NNs are designed (see section 4.1), the descriptor must be of fixed size, no matter how many atoms are present. Finally, it is advantageous if the descriptor is invariant with respect to transformations which do not alter the energy of the system. This way, translational invariance, rotational invariance, and invariance with respect to permutation of equivalent atoms when predicting the energy are automatically ensured.

Here, the atomic descriptor consists of two parts: One part encodes the atomic species (C, H, O, ...) and a second part encodes the local environment up to a cut-off radius R.[a] There are several reasons for introducing a cut-off. First, the energy prediction scales linearly with respect to the number of atoms present in the system of interest. Second, while the NN can be trained on rather small systems, it can then be applied to much larger systems, because locally, atomic environments of small and large systems are equivalent. Finally, it is a valid assumption that most (but not all) chemical interactions which are relevant to the energy of the system, such as bonding, are inherently short-ranged. Methods to correct for long-range interactions are well-known in the literature[40,155,170,171] and are discussed in section 4.2.2.

[a] Atomic descriptors that encode species and environment separately have been proposed previously.[158]

Species descriptor In principle, the atomic species could be encoded by a single number, either by an integer identifier (e.g. H= 1, C= 2, N= 3, ...) or by the nuclear charge Z (e.g. H= 1, C= 6, N= 7, ...). However, this introduces an ordinal relationship (e.g. H < C < N) between different atomic species, which can be detrimental to the performance of the NN. Since NNs are purely numerical algorithms, ordinal relations in inputs can directly correlate with the output response, which is not necessarily meaningful for atomic species. Alternatively, a one-hot[172] encoding (e.g. H = $[1\,0\,0\cdots]$, C = $[0\,1\,0\cdots]$, N = $[0\,0\,1\cdots]$) would be possible. However, two potential disadvantages of a one-hot encoding are that the dimensionality of the encoding vector must necessarily be equal to the cardinality of the set of atomic species present in the data and all encodings are equidistant by construction. Since, it is intuitive to expect e.g. elements from the same group in the periodic table to behave similar to one another, an optimal encoding should be able to directly represent these similarities.

For these reasons, the atomic species are rather encoded by embeddings. An embedding is a mapping from a discrete object i to a vector of real numbers $\mathbf{v}_i \in \mathbb{R}^D$, where D is the dimensionality of the embeddings. For example, word embeddings[173] find wide spread use in the field of natural language processing. Here, words are mapped to a comparatively low-dimensional vector space, such that semantically similar words (e.g. "red", "green", and "blue" or "king", "monarch", and "emperor") appear close to each other ($\|\mathbf{v}_{\text{red}} - \mathbf{v}_{\text{blue}}\| < \|\mathbf{v}_{\text{red}} - \mathbf{v}_{\text{king}}\|$). During the training process of the NN, the entries of the embedding vectors $\mathbf{v}_i$ are free parameters, such that meaningful embeddings are directly learned from data. Here, the dimensionality D of the embeddings is set equal to the number N_g of distinct groups (columns) in the periodic table which are present in the reference data. Note that a lower dimensionality would still allow a unique encoding of each element. However, elements from the same group in the periodic table are expected to have similar properties and choosing $D = N_g$ principally allows to encode every distinct group in orthogonal directions, thus avoiding ordinal relations between species. For more details on the concept of embeddings, refer to [174].

Environment descriptor All information about the local environment of a given atom i up to a cut-off radius R is contained in the neighbourhood density function ρ_i given by

$$\rho_i(\mathbf{r}) = \sum_{j,\|\mathbf{r}_j\|\leq R} Z_j \delta(\|\mathbf{r} - \mathbf{r}_j\|) \tag{4.10}$$

where the position $\mathbf{r} = [x\ y\ z]^\mathsf{T}$ is relative to atom i, Z_j and $\mathbf{r}_j$ are nuclear charge and relative position of neighbouring atom j, δ is the Dirac delta function, and the sum runs over all atoms j closer than R.[a] Note that the use of relative positions

[a] The concept of a neighbourhood density function has been used previously in the derivation of the SOAP similarity kernel.[175]

$\|\mathbf{r} - \mathbf{r}_j\|$ makes ρ_i translationally invariant and the commutativity of addition ensures permutational invariance. By construction, ρ_i is zero everywhere except for positions $\mathbf{r}_j$ of neighbouring atoms j, where the function value encodes the atomic species of j by its nuclear charge. Thus, ρ_i completely describes the local atomic environment of atom i up to a distance R.

To obtain a fixed length input for use in a feedforward layer, ρ_i is expanded into a basis set of fixed dimension

$$\rho_i(\mathbf{r}) \approx \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} \sum_{m=-l}^{l} c_{klm} \psi_{klm}(\mathbf{r}) \tag{4.11}$$

with expansion coefficients c_{klm} and basis functions $\psi_{klm}(\mathbf{r}) = g_k(r; R) Y_{lm}(\theta, \phi)$, where $g_k(r; R)$ (with $k \in [0, K-1]$) are radial basis functions and $Y_{lm}(\theta, \phi)$ are spherical harmonics (with $l \in [0, L-1]$).[a] K and L define the maximum degree of the radial and angular parts of the expansion and R the cut-off radius, respectively. To be consistent with the commonly used notation of spherical harmonics, the Cartesian coordinate vector $\mathbf{r}$ is transformed to spherical coordinates (Eq. 4.12).

$$\begin{aligned} r &= \|\mathbf{r}\| = \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arctan2(y, x) \\ \phi &= \arccos\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}} \right) \end{aligned} \tag{4.12}$$

The arctan2 function is defined in Eq. 4.13.

$$\arctan2(y, x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \arctan(\frac{y}{x}) + \pi & x < 0, y > 0 \\ \pm\pi & x < 0, y = 0 \\ \arctan(\frac{y}{x}) - \pi & x < 0, y < 0 \\ +\frac{\pi}{2} & x = 0, y > 0 \\ -\frac{\pi}{2} & x = 0, y < 0 \end{cases} \tag{4.13}$$

For the radial basis functions (RBFs) $g_k(r; R)$, many different choices are possible. Here

$$g_k(r; R) = s(r; R) \cdot \exp\left(-\frac{K^2}{R^2} \left(r - (k-1) \frac{R}{K} \right)^2 \right) \tag{4.14}$$

[a] For more details on the spherical harmonics, refer to Eq. 2.15 and Figure 2.2 in chapter 2.

is chosen, which ensures that basis functions are evenly spaced inside the cut-off sphere. Due to the cut-off function $s(r;R)$, $g_k(r;R)$ is zero whenever $r > R$. The cut-off function

$$s(r;R) = \begin{cases} 1 & \text{if } r \leq r_s \\ 1 - 6\left(\frac{r-r_s}{R-r_s}\right)^5 + 15\left(\frac{r-r_s}{R-r_s}\right)^4 - 10\left(\frac{r-r_s}{R-r_s}\right)^3 & \text{if } r_s < r < R \\ 0 & \text{if } r \geq R \end{cases} \tag{4.15}$$

with $r_s = R - \frac{R}{K}$ ensures that $g_k(r;R)$ has smooth first and second derivatives, such that no numerical artefacts are introduced when an atom enters or leaves the cut-off sphere, while leaving the Gaussian part of $g_k(r;R)$ largely unaffected (see Figure 4.11). The cut-off function $s(r;R)$ is a smooth approximation to the Heaviside step function (Eq. 4.2) and influences the value of $g_k(r;R)$ only when $r > r_s$.

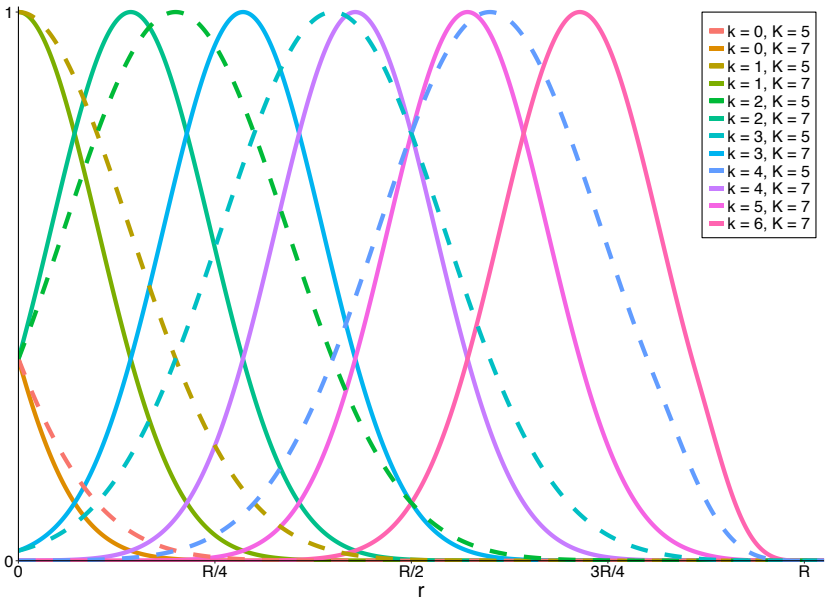


FIGURE 4.11: RBFs $g_k(r;R)$ (Eq. 4.14) for degree $K = 7$ (solid) and $K = 5$ (dashed). The different functions span equally distributed radial shells and smoothly go to zero for $r > R$ due to the cut-off function $s(r;R)$ (Eq. 4.15).

As long as K and L are sufficiently large, the information stored in the coefficients c_{klm} is comparable to that encoded in ρ_i (Eq. 4.10). Note that for predicting energies, some loss of information is not problematic as long as the resulting descriptor can distinguish different environments sufficiently well. The expansion coefficients c_{klm} for a general function $f(\mathbf{r})$ can be obtained from projecting $c_{klm} = \int f(\mathbf{r})\psi_{klm}(\mathbf{r})\mathrm{d}\mathbf{r}$. Fortunately, it is not necessary to calculate an integral to obtain the expansion coefficients for the neighbourhood density function (Eq. 4.10). Since $\rho_i(\mathbf{r})$ is the sum

of δ functions, the coefficients are efficiently obtained by summation (Eq. 4.16).

$$c_{klm} = \int \rho_i(\mathbf{r})\psi_{klm}(\mathbf{r})\mathrm{d}\mathbf{r} = \sum_{\|\mathbf{r}_j\|\leq R} Z_j\psi_{klm}(\mathbf{r}_j) \tag{4.16}$$

The values of the coefficients c_{klm} depend on the orientation of the chosen reference coordinate system, because the values of the spherical harmonics are orientation dependent. However, the coefficients for a given combination of k and l can be combined to a rotationally invariant quantity a_{kl} according to (Eq. 4.17).

$$a_{kl} = \left(\frac{4\pi}{2l+1}\sum_{m=-l}^{m=l}(-1)^m c_{klm}c_{kl-m}\right)^{\frac{1}{2}} \tag{4.17}$$

In total, there are $K \cdot L$ different a_{kl} values, which are concatenated to the atom embedding vector $\mathbf{v}$ of dimensionality N_g to form the descriptor vector $\mathbf{x}$. Because a_{kl} has continuous first derivatives with respect to the atom coordinates, derivatives necessary for e.g. force calculations are easily obtained by the chain rule. For all applications presented in section 4.2.2, $K = 7$, $L = 7$ and $R = 3$ Å were chosen as hyperparameters.

4.2.1.2 Neural network architecture

Instead of standard feedforward layers (see Eq. 4.6), square unit augmented layers[176] given by

$$\mathbf{h} = \sigma(\mathbf{x}\mathbf{W_1} + \mathbf{x}^2\mathbf{W_2} + \mathbf{b}) \tag{4.18}$$

are used to construct the NN. Here, $\mathbf{h} \in \mathbb{R}^{n_{\text{hidden}}}$ are the values of neurons in the hidden layer, $\mathbf{x}^2$ is shorthand notation for the entrywise square of the input $\mathbf{x} \in \mathbb{R}^{n_{\text{in}}}$, $\mathbf{W_1}, \mathbf{W_2} \in \mathbb{R}^{n_{\text{in}} \times n_{\text{hidden}}}$ are independent weight matrices, $\mathbf{b} \in \mathbb{R}^{n_{\text{hidden}}}$ is the bias vector, and $\sigma(x)$ the activation function (the shorthand notation $\sigma(\mathbf{x})$ means that $\sigma(x)$ is applied to the vector $\mathbf{x}$ entrywise). The reason for using square unit augmented layers is that properties reminiscent of RBF networks[177–179] can be included at little additional computational expense,[176] provided that a sigmoidal activation function is used (see Figure 4.12 for an illustration).

The activation function of the hidden layers is chosen to be $\sigma(x) = s \cdot \operatorname{arcsinh}(x)$ where $s = 1.25673480$ ensures that $\sigma(x)$ has self-normalizing properties (activations converge automatically to zero mean and unit variance), similar to the recently proposed SELU function.[180] For the output layer, the identity function $\sigma(x) = x$ is used. The function $\operatorname{arcsinh}(x)$ was found to give superior results compared to more commonly used activation functions such as $\tanh(x)$. One possible reason for the improved performance is that the function does not saturate for large or small values of x (see Figure 4.12), which alleviates the vanishing gradient problem.[139]

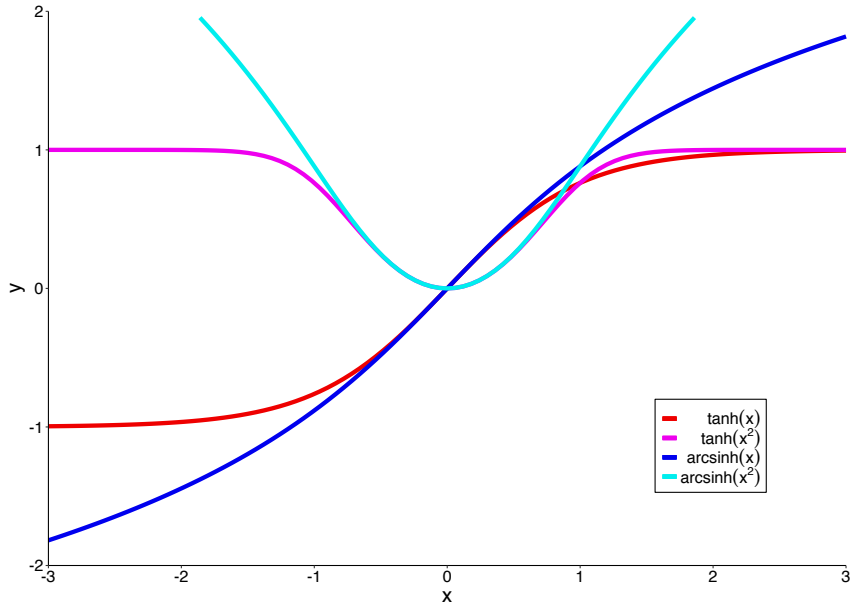


FIGURE 4.12: Graphs for two different sigmoidal activation functions $\tanh(x)$ (red) and $\operatorname{arcsinh}(x)$ (blue). When x^2 instead of x is used as input, i.e. $\tanh(x^2)$ (magenta) and $\operatorname{arcsinh}(x^2)$ (cyan), the graphs become bell-shaped and are reminiscent of RBFs.

In summary, the energy prediction consists of the following steps (see also Figure 4.10):

1. The descriptor $\mathbf{x}_i$ for atom i with nuclear charge Z_i is generated by concatenating the embedding vector $\mathbf{v}_{Z_i}$ with the environment descriptor generated from the neighbourhood density (Eq. 4.10) of atom i (see section 4.2.1.1).

2. The descriptor $\mathbf{x}_i$ is used as input for a feedforward NN, which outputs atomic energy contributions E_i. For all applications presented in section 4.2.2, the NNs have two hidden square unit augmented layers (Eq. 4.18) with 100 and 50 neurons each.

3. Steps 1 and 2 are repeated for every atom i and the contributions E_i are summed to give the total energy E_{tot}. The same NN is used for all atoms.

4.2.1.3 Training

NNs are trained to predict energies on the QM9 dataset,[166] several MD datasets,[116] and a dataset for H-transfer in malonaldehyde. The QM9 dataset forms a subset of the GDB-17 database[181] and contains 133 885 molecules consisting of H, C, N, O and F with up to 29 atoms, including up to 9 heavy atoms. The range of energies spans several thousand kcal mol^{-1}. All properties in the QM9 dataset were calculated at the B3LYP/6-31G(2df,p) level of theory.[166] The MD datasets consist of *ab initio* MD trajectories for benzene, uracil, naphthalene, aspirin, salicylic

acid, malonaldehyde, ethanol and toluene calculated at the PBE+vdW-TS[182,183] level of theory. They range in size from 150 000 to nearly 1 000 000 conformational geometries.[116] The H-transfer dataset for malonaldehyde was generated by sampling 250 000 geometries from a 5 ns MD trajectory run at 750 K using CHARMM[80] and a molecular mechanics with proton transfer (MMPT) based reactive force field (FF).[43,184]. These simulation conditions lead to ready proton transfer and constitute a set of reactive geometries. The reference energy for each geometry was calculated at the MP2/6-311++G(d,p) level of theory using Gaussian09.[70]

Prior to training, each dataset is split into three parts: the training set, the validation set and the test set. During training, the squared error per atom (SEpA) (Eq. 4.19)

$$\mathrm{SEpA} = \frac{1}{N}\left(E_{\mathrm{ref}} - \sum_{i=1}^{N} E_i\right)^2 \tag{4.19}$$

is minimized via Adam optimization in batches[185] of ten reference structures, using a learning rate of 10^{-4}. E_{ref} is the reference energy of a structure from the training set, and E_i are the predicted atomic contributions of the N atoms of the reference structure. During one so-called epoch of training, the network trains once on each datum in the training set. After each training epoch, the mean SEpA is also calculated for the structures in the validation set. Every network is trained between 5500 to 10 000 epochs and the model which performs best on the validation set is selected to predict the test set. As such, although the validation set is not directly used in training, it indirectly influences which model is selected. This method is also known as early stopping and is frequently used to prevent overfitting.[176] Since the test set is not used at all during the training process, the MAE and root mean squared error (RMSE) of predictions on the test set indicate how well the model generalizes to unknown data.

To speed up the training process and to improve convergence, all inputs (apart from embeddings) to the network are transformed to their z-score[186] according to mean and standard deviation of the respective inputs in the training set. This ensures that the numerical range of input values is close to the regions where the activation function is most responsive. Note that all numbers needed for calculating the z-scores are constants that only depend on the chosen training set and can be considered to be part of the descriptor. The transformation to z-scores or similar normalization methods have only numerical reasons and are standard practice when working with NNs.[176]

Similarly, instead of directly interpreting the output y of the NN as atomic contribution to the energy, $E_i = s_1 \cdot y + s_2$ is used instead, where s_1 and s_2 are additional scale and shift parameters that are optimized during training. However, instead of initializing them randomly like the other trainable parameters, they are initialized according to the standard deviation (s_1) and mean (s_2) of the per atom average of the reference energies in the training set. NNs are found to converge faster when the scale

and shift operations are introduced, because a larger learning rate can be used due to the predictions starting with the correct range of values. After training is finished, it is possible to incorporate s_1 and s_2 directly into the weights and biases of the output layer to save the additional computational step required by introducing $E_i = s_1 \cdot y + s_2$.

NNs are trained with Tensorflow[187] using training set sizes of 1k, 2.5k, 5k, 10k, 25k, 35k, 50k, 75k, and 100k structures for the QM9 dataset and training set sizes of 25k, 50k, and 100k for the MD and H-transfer datasets. In all cases, 2k additional structures are used as validation set, whereas the remaining structures constitute the test set. For every training set size in the QM9 dataset, five different NNs are trained based on a different, randomly chosen training, validation and test set. This provides a means to obtain statistics on their performance.

Further, to investigate whether the predictions of an NN also scale to larger systems, a single NN is trained on the QM9 dataset only on reference structures that contain 15 atoms or less (26 328 structures). Out of the remaining structures, 2k are reserved as validation set during training and the generalization error is estimated by predicting the energies of all other structures in the QM9 dataset with more than 15 atoms.

4.2.2 Results

4.2.2.1 Atomic energies

Since the NNs are trained to decompose the energy of a chemical system into atomic contributions, it is instructive to visualize the "energy spectrum" for different elements. Figure 4.13 shows atomic energy spectra for the QM9 dataset. The spectra are non-uniform and contain multiple peaks at well defined energies. The atomic energies of C atoms span the widest range (> 100 kcal mol^{-1}), followed by N (> 60 kcal mol^{-1}), O (> 40 kcal mol^{-1}), H (> 20 kcal mol^{-1}) and F (> 15 kcal mol^{-1}). It is intuitive to associate different peaks with different clusters ("types") of atoms, where atoms in the same cluster are similar in energy E_i due to similar atomic environments.

To verify this hypothesis, atoms with similar environments are clustered based on chemical graphs,[189] where nodes correspond to atoms and edges represent bonds. Two atoms are considered as bonded if their distance does not exceed 1.25 times the sum of their covalent radii. For this purpose, the following covalent radii for the different elements are assumed: H: 0.31 Å, C: 0.76 Å, N: 0.71 Å, O: 0.66 Å, and F: 0.57 Å. Starting from the atom of interest as root node, the molecular graph is traversed in a depth-first[190] tree traversal up to depth two. For every node encountered in the traversal, a string consisting of its element and the number of bonds it forms (e.g. "C4" or "H1") is appended to an initially empty string. Each newly appended substring is enclosed in parentheses, such that the bonding pattern can be reconstructed. After this process, the strings are converted to a canonical form by ordering all

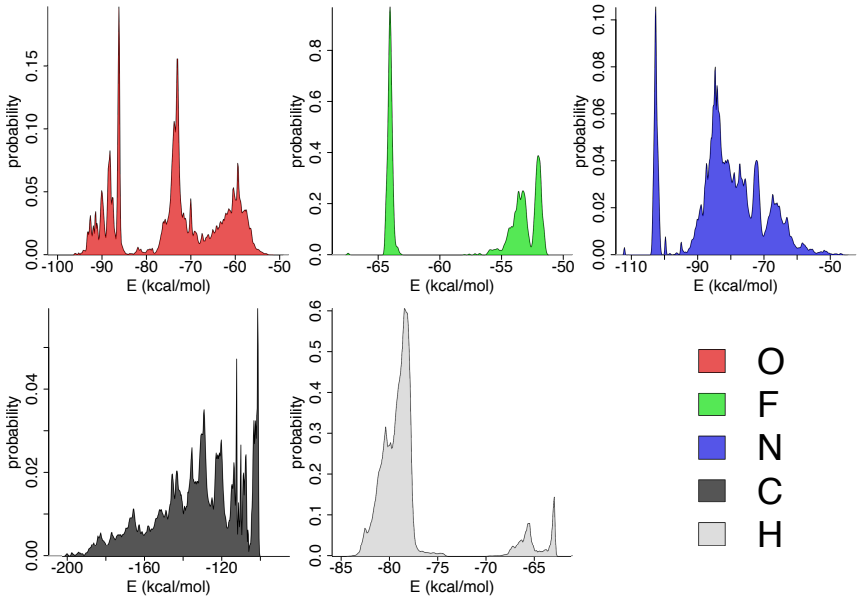


FIGURE 4.13: "Spectra" of atomic energies in the QM9 dataset for different elements (relative to the energy of a free atom). Results were obtained by averaging the predictions of all five NNs trained on 100k structures. The curves are obtained by kernel density estimation with the Sheather-Jones bandwidth selection method.[188]

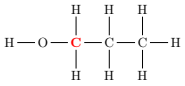
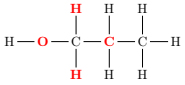
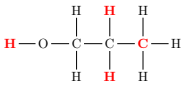
substrings enclosed in parentheses by an arbitrary priority definition.[a] All atoms that share identical final strings are assigned to the same cluster. The process is exemplified in Table 4.2 for the carbon atom in position one in 1-propanol (C_3H_7OH).

In total, the QM9 dataset[166] contains 1 230 122 H atoms, 846 557 C atoms, 139 764 N atoms, 187 996 O atoms, and 3314 F atoms. After clustering, these numbers reduce to 168 (H), 34 647 (C), 4271 (N), 1130 (O), and 22 (F) different types. The large number of different clusters is not surprising, considering the vast number of theoretically possible combinations for constructing bonding graphs of depth two, given five different atomic species and diverse possible bonding patterns for each of them. Figure 4.14 shows how many atoms in the dataset belong to each cluster. Interestingly, the counts closely follow a Pareto distribution,[191] indicating that the vast majority of atoms is described by just a few atomic types. For example, more than half of all C atoms belong to the 331 most common C-atom clusters.

Since only graph-based information (but no geometric information such as distances and angles) is considered in the clustering approach, it is not evident that atoms belonging to the same cluster are energetically similar. As a qualitative test for how meaningful the clustering is, the cluster statistics (mean and variance of atomic

[a]Without canonicalization, the order in which the graph is traversed affects the final string, which can result in two different strings for atoms with identical chemical environment.

TABLE 4.2: Step-by-step overview of the graph-based clustering method for carbon at position one in 1-propanol.

depth	string	chemical graph
0	C4	
1	C4(O2)(C4)(H1)(H1)	
2	C4(O2(H1))(C4(C4)(H1)(H1))(H1)(H1)	

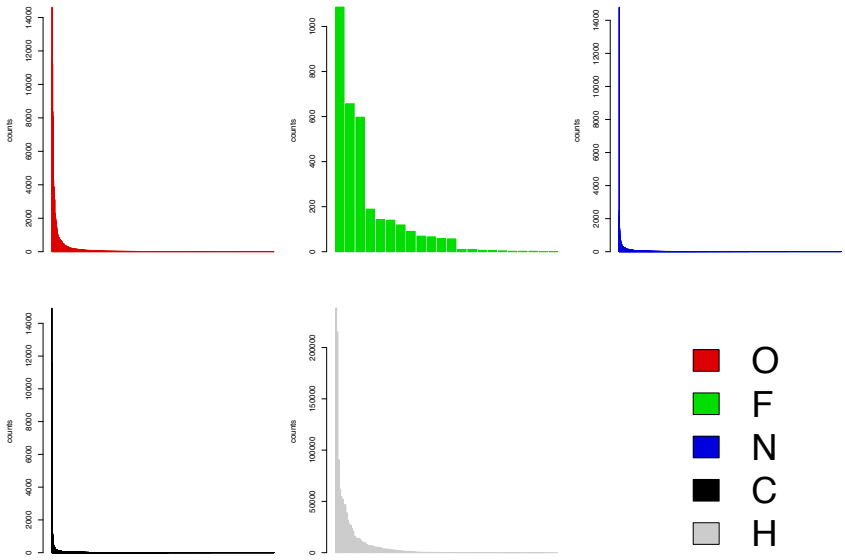


FIGURE 4.14: Total counts for atom types obtained by clustering based on chemical graphs. Each bar of a histogram corresponds to a single cluster, ordered from highest to lowest count.

energies for each cluster) from the raw data is considered (see Figure 4.13). For this, every cluster is represented by a Gaussian distribution with mean and variance equal to the corresponding cluster statistics, and normalized according to the atom count. Even though assuming a Gaussian distribution is a crude approximation, the sum of all Gaussians (see Figure 4.15) closely resembles Figure 4.13, so the graph-based clustering approach is considered to be meaningful.

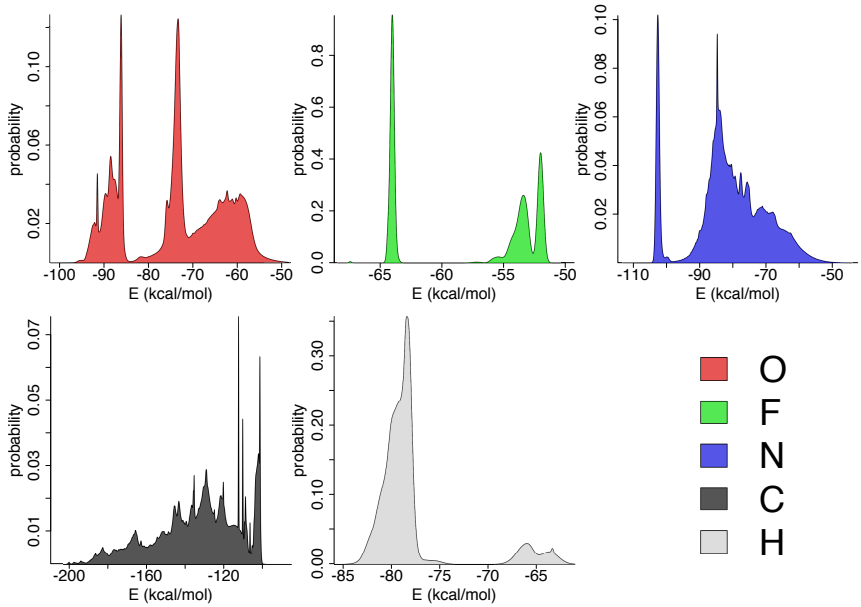


FIGURE 4.15: "Simulated spectra" of atomic energies obtained by the summation of Gaussian distributions with mean and variance equal to the corresponding cluster statistics, normalized according to their atom count.

To interpret the data, chemical similarities between different clusters are analysed and they are summarized based on functional groups into different atom types. Apart from allowing interpretation of the network predictions, the energies of different atom types can be tabulated and used for a rapid estimate of the energy of a molecule given only its chemical structure, similar to how NMR chemical shifts can be estimated.[192] Table 4.3 lists atomic energies (relative to a free atom) of functionally different C atom types.

TABLE 4.3: Atomic energies of selected **C** atom types (mean plus or minus one standard deviation).

type	diagram	$\overline{E}$ (kcal mol^{-1})
hydrocarbyls		
primary alkyl		-101.8 ± 0.7

TABLE 4.3: Atomic energies of selected **C** atom types. (*continued*)

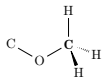
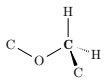
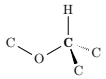
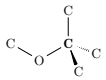
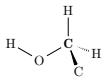
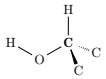
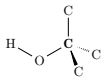
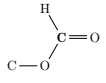
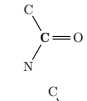
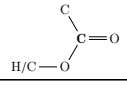
type	diagram	$\overline{E}$ (kcal mol^{-1})
secondary alkyl		−114.3 ± 4.8
tertiary alkyl		−129.5 ± 6.3
quaternary alkyl		−151.3 ± 5.9
primary alkenyl		−110.0 ± 6.2
secondary alkenyl		−128.6 ± 2.1
tertiary alkenyl		−150.2 ± 8.3
primary alkynyl		−112.5 ± 1.0
secondary alkynyl		−137.1 ± 3.0
secondary conjugated alkenyl		−134.0 ± 3.0
tertiary conjugated alkenyl		−156.9 ± 5.6
bound to nitrogen		
methyl amine		−108.6 ± 0.5
primary-C amine		−120.2 ± 7.4
secondary-C amine		−134.0 ± 8.1
tertiary-C amine		−156.7 ± 7.1
nitrile		−144.0 ± 1.9
primary-C imine		−142.0 ± 2.2
secondary-C imine		−165.6 ± 3.7
bound to oxygen (may also be bound to nitrogen)		
methoxy		−110.3 ± 0.1

TABLE 4.3: Atomic energies of selected **C** atom types. (*continued*)

type	diagram	$\overline{E}$ (kcal mol^{-1})
primary ether		−126.8 ± 6.0
secondary ether		−141.2 ± 8.4
tertiary ether		−161.4 ± 7.5
primary hydroxyl		−130.3 ± 1.3
secondary hydroxyl		−149.3 ± 4.0
tertiary hydroxyl		−168.0 ± 4.7
aldehyde		−146.0 ± 1.3
formyl amide		−161.6 ± 1.4
formyl ester		−164.5 ± 1.1
ketone		−167.2 ± 2.3
amide		−184.1 ± 4.0
carboxyl ester/acid		−188.9 ± 4.7
bound to fluorine		
“aza-conjugated" fluoro		−181.1 ± 1.3
“oxy-conjugated" fluoro		−182.9 ± 1.5
“aza-aza-conjugated" fluoro		−188.0 ± 1.3
“aza-oxy-conjugated" fluoro		−192.9 ± 1.0
fluoro methyl		−200.2 ± 0.5

Several trends can be observed: For pure hydrocarbyls, C atoms with a triple bond are more stable than C atoms with double or single bonds, in accordance with the increased bond strengths. An exception are conjugated sp^2-hybridized C atoms, which are even more stable due to their "aromatic" nature. When bound to electronegative atoms, such as N, O and F, the stabilization energy of carbon atoms appears to be correlated with the electronegativity of the bonding partner. A physically appealing interpretation is that a large difference in electronegativity increases the ionic character of the bond and therefore increases the stabilization energy.

While such trends may be obvious to trained chemists, a somewhat more subtle effect can be seen in the increasing stability from primary to quaternary C-atoms. This can be explained by hyperconjugation.[a] Such a resonance stabilization is well known for carbocations and carbon radicals, which become more stable with increasing number of neighbouring alkyl groups. A related trend is found from chemical shift measurements in ^{13}C-NMR experiments, where typical shifts increase from 15–30 ppm, to 22–45 ppm and further to 30–58 ppm when going from primary to tertiary C-atoms.[194] This is usually attributed to the increased nuclear shielding due to the additional electron density around the nucleus.

Similar observations can be made for H, N, O and F atoms.[b] Note that some of the previously discussed trends can be reversed for the other elements. For example, instead of being stabilized by neighbouring alkyl groups, O atoms typically are destabilized by the +I effect. However, this is to be expected since O atoms are already partially negatively charged due to their high electronegativity. The +I effect then leads to an amplification of this charge and therefore destabilization.

4.2.2.2 Prediction errors

QM9 dataset MAEs and RMSEs for the NNs trained with different training set sizes are summarized in Table 4.4 and compared with the performance of the DTNN[162] and SchNet.[163] The NN trained on 100k reference structures predicts structures in the QM9 dataset accurately with an MAE of 0.41 kcal mol^{-1} and an RMSE of 0.86 kcal mol^{-1}. Figure 4.16 shows the convergence of MAE and RMSE with increasing training set size.

While MAE and RMSE are useful measures for the overall performance of a method, it can also be instructive to consider how errors are distributed. Figure 4.17 reveals that for all training set sizes starting from 10k, more than half of all errors are below 0.5 kcal mol^{-1}, with most errors being < 0.1 kcal mol^{-1}. However, all distributions exhibit long tails, which implies that there are rare but extreme outliers.

[a] Electron density from occupied σ-bonds is donated to unoccupied orbitals,[189] also known as the positive inductive or +I effect.[193]

[b] Data is omitted here for conciseness, but can be found in the supporting information of [119].

TABLE 4.4: Prediction errors for the QM9 dataset. MAE and RMSE (given in kcal mol^{-1}) on the test set for different training set sizes. Results are compared to values reported in for the DTNN[162] and SchNet.[163]

	training set size	MAE	RMSE
this work	1000	1.85 ± 0.09	3.53 ± 0.57
	2500	1.23 ± 0.03	2.45 ± 0.14
	5000	0.95 ± 0.01	1.94 ± 0.10
	10 000	0.73 ± 0.01	1.59 ± 0.08
	15 000	0.63 ± 0.01	1.40 ± 0.08
	25 000	0.55 ± 0.01	1.22 ± 0.07
	35 000	0.50 ± 0.00	1.06 ± 0.02
	50 000	0.46 ± 0.01	0.98 ± 0.04
	75 000	0.43 ± 0.01	0.89 ± 0.06
	100 000	0.41 ± 0.00	0.86 ± 0.14
DTNN	25 000	1.04 ± 0.02	1.53 ± 0.02
	50 000	0.94 ± 0.01	1.37 ± 0.01
	100 000	0.84 ± 0.02	1.21 ± 0.02
SchNet	50 000	0.59	—
	100 000	0.34	—
	110 462	0.31	—

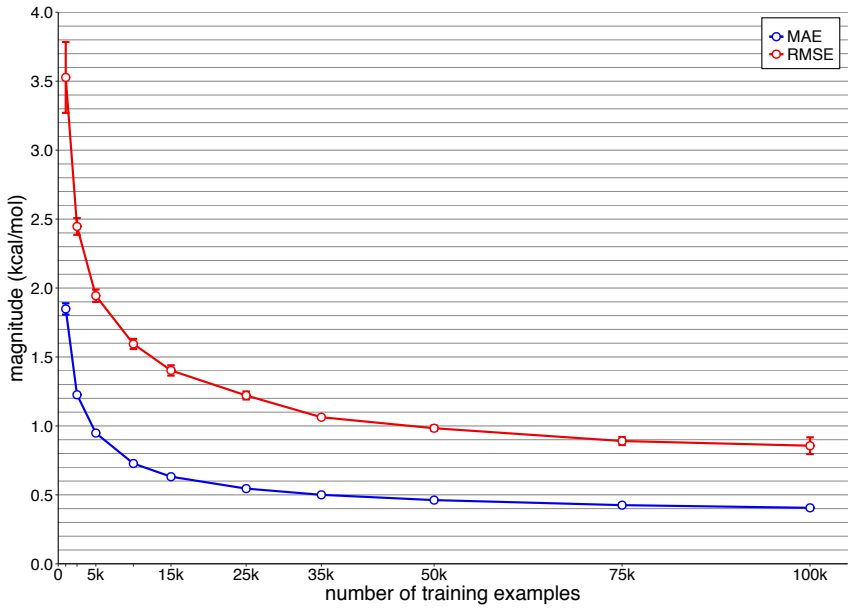


FIGURE 4.16: MAE (blue) and RMSE (red) depending on the size of the training set, averaged over five independent runs per training set size. Error bars indicate one standard deviation.

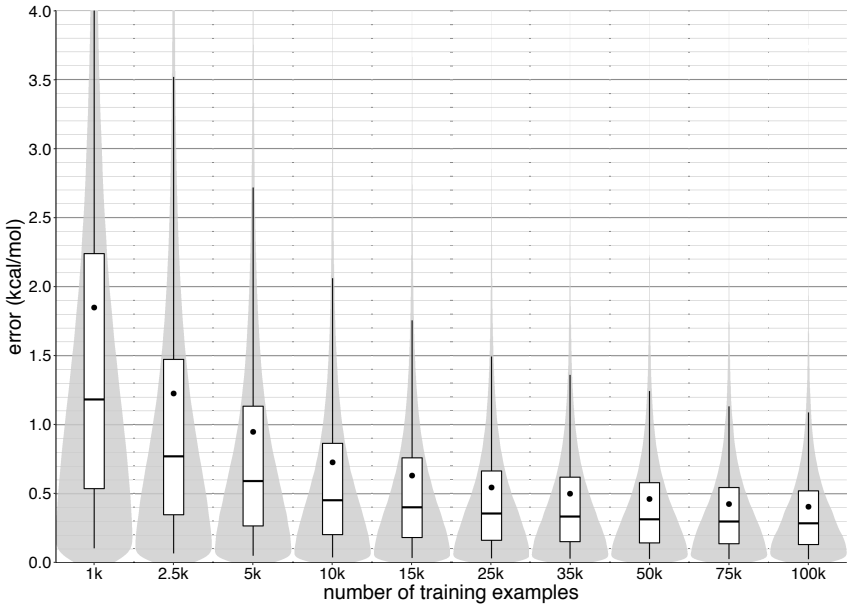


FIGURE 4.17: Normalized error distribution (grey) depending on the size of the training set (test errors from five independent runs per training set size are combined). A white box spans between the 25% and 75% quantiles, with a black horizontal line indicating the median and a black dot indicating the mean of the distribution. The whiskers mark the 5% and 95% quantiles.

The question remains whether reasons for the outliers can be identified. For example, particular structures could be difficult to predict simply because they contain rare atomic environments which are under-represented in the training set. To quantify how well a structure in the test set is represented by structures in the training set, the concept of a representation number is introduced. For every structure, the relative frequency of the atom clusters (see section 4.2.2.1) in the training set are combined via a harmonic average to form the structure's representation number. Structures with a small representation number therefore contain one or several uncommon atomic environments, which the NN could not necessarily learn to predict accurately from the data it was presented during training.

Notable examples for such structures are very small molecules, including water, methane and fluoromethane (all part of the QM9 dataset), which contain unique atomic environments not found in any other structure. For example, oxygen and hydrogen atoms in a water molecule are chemically very different to oxygen and hydrogen atoms found in other hydroxyl groups. This is highlighted by noting that the bond dissociation energy of an O–H bond in water is 119.2 kcal mol^{-1}, whereas for a typical O–H bond in hydroxyl groups, it is only 102.3 kcal mol^{-1}.[195] Similarly, the dissociation energy of C–H bond in methane is around 103.0 kcal mol^{-1}, compared to 113.0 kcal mol^{-1} for a typical C–H bond of a primary carbon.[195] Figure 4.18 reveals that large prediction errors occur almost exclusively for structures with low representation number, which are under-represented in the training set. However, the inverse is not always true and low representation numbers do not necessarily lead to large prediction errors. Since outliers follow the same patterns, it is possible to systematically improve the prediction capabilities of an NN for structures with a low representation number by simply including appropriate reference structures in the training set.

While most outliers can be explained by under-represented chemical environments in the training data, some of the largest prediction errors are probably due to a different reason. They belong to a group of eleven molecules in the QM9 dataset for which the electronic structure calculation did not converge at all (three molecules) or only using loose convergence criteria (eight molecules).[166] Most of these structures feature unconventional chemical bonding and their electronic structure potentially has multi-reference character. Therefore, it is possible that the quantum mechanical reference energies themselves are erroneous for these structures, explaining the large prediction errors. At the very least, they seem to be particularly difficult to predict for *ab initio* methods as well. Some of the difficult-to-converge structures are shown in Figure 4.19 along with their average prediction errors. Note that, even though many of these structures contain a motif reminiscent of 1,2,3-oxadiazole, the presence of this motif alone is not the cause for the large prediction errors: The QM9 dataset contains close to 1k similar structures, for which accurate predictions are possible.

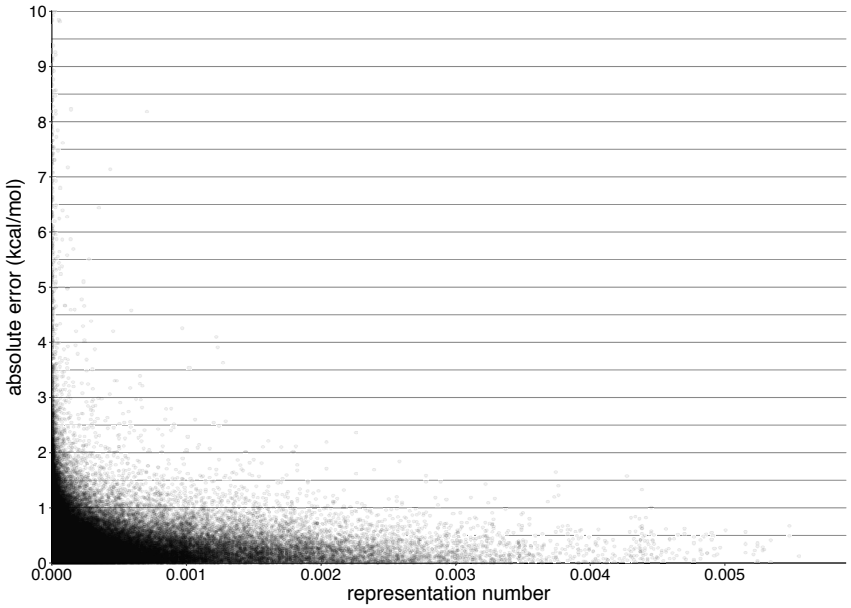


FIGURE 4.18: Absolute error versus representation number. Data for five different NNs using 100k training examples are combined. Individual data points are plotted semi-transparent, such that high density regions appear darker.

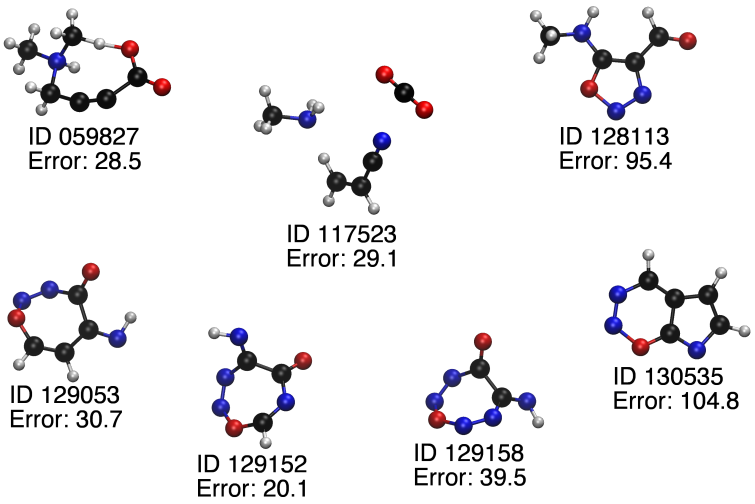


FIGURE 4.19: Difficult-to-converge structures (C: black, N: blue, O: red, H: white) with particularly large prediction errors (in kcal mol^{-1}) are shown along with their corresponding ID in the QM9 dataset. Errors are averaged across NNs trained on 100k reference structures (only NNs that were trained without the given structure in the training or validation set were considered).

While the results with randomly chosen training sets[196] are promising, it is interesting to see whether representations learned from small structures can be used to predict energies for larger structures. An NN trained on all structures in the QM9 dataset containing up to 15 atoms (26 328 structures) is able to predict structures with more than 15 atoms (107 557 structures) with an MAE of 1.01 kcal mol^{-1} and an RMSE of 1.69 kcal mol^{-1}. This demonstrates that the learned representations are transferable and can be used to accurately predict larger structures. Nonetheless, the performance is inferior compared to a randomly chosen training sets drawn from the full data set. One possible physical explanation is that this is due to the lack of an adequate description of long-range interactions, which are more important for extended structures containing many atoms. These deficiencies could be addressed by explicitly including long-range contributions into the prediction.

MD datasets MAEs and RMSEs for the NNs trained with different training set sizes are summarized in Table 4.5 and compared with results for gradient-domain machine learning (GDML).[116] Predictions are accurate for all molecules and can be systematically improved by increasing the training set size. GDML models are trained on only 1000 structures, but use the atomic forces instead of total energies as reference data, which enhances their predictive power.[116] It has been shown previously that NNs benefit as well from including forces in their loss function (see Eq. 4.19).[163] Hence, it is likely that predictions from the NN could be further improved by including force information during training.

H-transfer dataset MAEs and RMSEs for the NNs trained with different training set sizes for malonaldehyde are summarized in Table 4.6. The results show that accurate predictions are possible with rather small training set sizes and can be systematically improved by increasing the number of reference structures. A direct comparison of the NN predictions and MP2 reference energies yields a correlation coefficient of 0.997. Figure 4.20 shows a 10 ps MD trajectory of malonaldehyde, where *ab initio* reference values and NN predictions are overlayed.

4.2.3 Discussion and conclusion

Although the results show that accurate predictions can be obtained from training an NN with a descriptor based on encoding the chemical environment of an atom, it is useful to discuss potential problems and possible improvements to the prediction method. For example, even though introducing a cut-off radius R is necessary for computational efficiency, it can limit the accuracy of the NN. Since all atoms beyond the cut-off radius of $R = 3$ Å are ignored in the descriptor by construction, interactions extending over larger distances can not be captured by the present approach. Most covalent interactions relevant in chemistry are sufficiently short-ranged that this is not an issue, but there are also important long-ranged nonbonded interactions contributing to the energy, namely Coulomb and dispersion interactions. They are

TABLE 4.5: Prediction errors for the MD datasets. MAE and RMSE (given in kcal mol^{-1}) on the test sets are given for different training set sizes. Values in brackets are results for GDML.[116] Note that the GDML approach uses different reference data and training set sizes (see text).

molecule	training set size	MAE		RMSE	
aspirin	25 000	0.45		0.61	
	50 000	0.34	(0.27)	0.44	(0.36)
	100 000	0.27		0.35	
benzene	25 000	0.11		0.14	
	50 000	0.10	(0.07)	0.13	(0.09)
	100 000	0.09		0.12	
ethanol	25 000	0.21		0.30	
	50 000	0.18	(0.15)	0.24	(0.20)
	100 000	0.15		0.20	
malonaldehyde	25 000	0.44		0.60	
	50 000	0.38	(0.16)	0.51	(0.25)
	100 000	0.32		0.43	
naphthalene	25 000	0.41		0.54	
	50 000	0.37	(0.12)	0.47	(0.15)
	100 000	0.32		0.42	
salicylic acid	25 000	0.44		0.59	
	50 000	0.37	(0.12)	0.48	(0.15)
	100 000	0.32		0.42	
toluene	25 000	0.45		0.60	
	50 000	0.40	(0.12)	0.52	(0.16)
	100 000	0.35		0.46	
uracil	25 000	0.30		0.40	
	50 000	0.24	(0.11)	0.31	(0.14)
	100 000	0.20		0.26	

TABLE 4.6: Prediction errors for the H-transfer dataset for different training set sizes. MAE and RMSE are given in kcal mol^{-1}.

training set size	MAE	RMSE
25 000	0.36	0.49
50 000	0.30	0.40
100 000	0.25	0.34

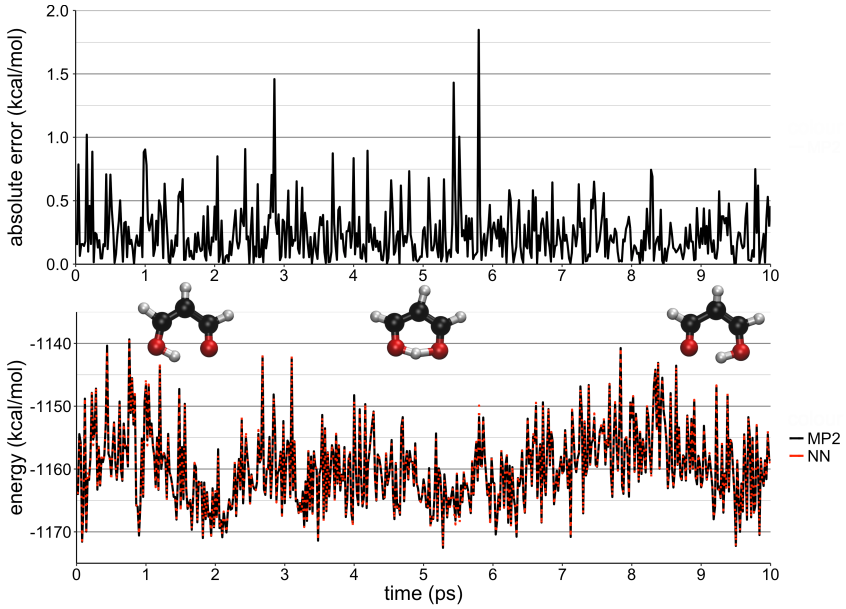


FIGURE 4.20: First 10 ps of a MD trajectory of malonaldehyde with intramolecular H-transfer. **Top**: Energy difference (absolute error) between MP2/6-311++G(d,p) reference energies and energies predicted by the NN trained on 100k reference structures. **Bottom**: Reference energies (solid black) and NN predictions (dotted red) along the trajectory.

especially important for the correct description of intermolecular interactions and are therefore crucial for condensed phase systems. While it is always possible to increase R until the error introduced by the cut-off is negligible, this is not very efficient, as a larger number of atoms would need to be considered for the calculation of the expansion coefficients c_{klm} (see Eq. 4.10 and Eq. 4.16). Further, it is likely that higher-order expansion terms (see Eq. 4.11) are necessary to resolve differences between atomic environments for larger R, such that the calculation of the descriptor becomes more expensive. Fortunately, the physical laws governing Coulomb and dispersion interaction are well understood, such that it is possible to include both contributions explicitly without increasing the cut-off R.

For better describing Coulomb interactions, separately trained NNs have previously been used[197] to predict environment-dependent Hirshfeld charges.[198] The electrostatic contribution E_{ele} is then simply subtracted from the total energy E_{tot} prior to training NNs for predicting the short-range contributions. The total energy can be recovered by combining electrostatic energies calculated from the predicted charges and the short-range contributions. Note that only charge-charge interactions are necessary for the calculation of the electrostatic energy, as interactions between higher-order multipoles[54,199] decay faster and can therefore be implicitly described in the short-range contributions.[40] Such approaches could also be incorporated into the prediction without introducing a second NN: Instead, the existing NN could simply be trained to predict an atomic energy contribution E_i and an environment dependent charge q_i simultaneously.[a] Also, it is not necessary to rely on a charge decomposition scheme such as the Hirshfeld method[198] to obtain a reference value for q_i. Recently, it was shown that an NN can be trained to predict environment dependent charges such that the electrostatic moments, a true quantum mechanical observable, are reproduced.[200] This way, no arbitrary decomposition scheme needs to be imposed.

To account for long-range dispersion interactions, it was shown[201] that the D3 scheme in density functional theory (DFT) calculations proposed by Grimme *et al.*[202] can be used for NNs without modification. Since the NN is trained on DFT reference energies, the standard C_6 coefficients[202] for calculating the dispersion interaction can be reused.

The atomic energy contributions predicted by the NN are chemically intuitive and may offer new insights. For example, they can be used as a guideline for designing novel types of empirical FFs through atom typing based on quantitative information instead of chemical intuition. Finally, it is possible to systematically improve the predictions of the NN by adding new reference data to the training set.

[a] For this, introducing a second output neuron and an appropriate modification of the objective function (Eq. 4.19) are necessary.

Acknowledgements

I would like to thank Thomas Brandl, Lorenzo D. Bizzini, Rajesh Mannancherry and Patrick Zwick for fruitful discussions concerning the interpretation of trends among atomic energy contributions. Further, I would like to express my gratitude towards Dr. Zhen-Hao Xu for generating structures for malonaldehyde.

4.3 Message-passing HDNN

In this segment, an HDNN of the message-passing type is presented. Further, two new datasets for benchmarking and comparing different quantum chemical models are introduced. Section 4.3.1 gives details on the prediction process and NN architecture, the training procedure and how datasets were generated. In section 4.3.2, results on both established benchmarks and the newly introduced datasets are presented. The results are summarized and discussed in section 4.3.3. The datasets introduced in this segment are available from `www.quantum-machine.org` or directly from [203, 204]. The code for training the neural network is freely available from `https://github.com/MeuwlyGroup/PhysNet`.

4.3.1 Methods

4.3.1.1 Neural network architecture

In the following, a special-purpose deep NN architecture designed for applications in computational chemistry is described. It predicts atomic contributions to properties (such as energy) of a chemical system composed of N atoms based on atomic features $\mathbf{x}_i \in \mathbb{R}^F$, where F is the dimensionality of the feature space.[a] The features simultaneously encode information about nuclear charge Z and local atomic environment of each atom i and are constructed by iteratively refining an initial representation depending solely on Z_i through coupling with the feature vectors $\mathbf{x}_j$ of all neighbouring atoms $j \neq i$ within a cut-off radius r_{cut}. In the following, the individual building blocks of the NN and important underlying concepts are described in more detail. Note that for the remainder of this section, superscripts l are used to denote features or parameters of layer l and the activation function $\sigma(x)$ refers to the shifted softplus function[163] given by $\sigma(x) = \log(e^x + 1) - \log(2)$.[b] All building blocks were implemented in the TensorFlow[187] framework.

[a] The atomic features take over the role of the descriptor vector in descriptor-based HDNNs, see section 4.2.1.1.

[b] The shifted softplus function is identical to the softplus function (Eq. 4.9, see Figure 4.8), but shifted downwards by the constant $\log(2)$, such that $\sigma(0) = 0$. This was found to improve the performance significantly, presumably because it is easier for the NN to learn to ignore features with small magnitude.

Embedding layer An embedding is a mapping from a discrete object to a vector of real numbers (see also the species descriptor in section 4.2.1.1). For example, word embeddings[173] find wide spread use in the field of natural language processing, where semantically similar words are mapped such that they appear close to each other in the embedding space. Here, atomic numbers $Z \in \mathbb{N}$ are mapped to embeddings $\mathbf{e}_Z \in \mathbb{R}^F$, where the entries of the embedding vectors $\mathbf{e}_Z$ are parameters. The embedding layer initializes the atomic features of an atom with nuclear charge Z to the corresponding embedding vector $\mathbf{e}_Z$ (Eq. 4.20).

$$\mathbf{x}_i^0 = \mathbf{e}_{Z_i} \tag{4.20}$$

The output of the embedding layer is passed to a stack of N_{module} modules sharing the same composition (but not parameters), which are described in the following.

Module architecture Each module in the stack (apart from the first one, which receives its input from the embedding layer) takes the output of the previous module and couples the features $\mathbf{x}_i$ of each atom i with the features $\mathbf{x}_j$ of neighbouring atoms j through an interaction block. The features are then further refined atom-wise through $N_{\text{residual}}^{\text{atomic}}$ residual blocks. Subsequently, the computation splits into two branches: One branch passes the atomic features onwards to the next module in the stack (if present) without further modification, whereas the other branch passes the features to an output block, which computes the contribution of the module to the final NN prediction. The split into two branches helps to decouple the feature representations passed between modules from the prediction task at hand. The individual components of each module are described below.

Residual block The performance of an NN should always increase, or at least stay the same, when its depth (i.e. the number of layers stacked on top of each other) is increased, as additional layers could in principle always reduce to the identity mapping and should therefore never decrease the performance. However, this is not observed in practice: As NNs get deeper, they become increasingly harder to train because of the vanishing gradients problem,[139] which leads to a degradation of their performance. To alleviate this, it was proposed to add shortcut connections to the architecture, which skip one or several layers and create a so-called residual block.[205] Since their first introduction, the design of residual blocks was further refined to allow completely unhindered gradient flow through all layers of an NN.[206] It was shown that stacking these so-called pre-activation residual blocks allows successfully training NNs more than 1000 layers deep.[206] Here, pre-activation residual blocks are used extensively to refine the atomic features according to

$$\mathbf{x}_i^{l+2} = \mathbf{x}_i^l + \mathbf{W}^{l+1}\sigma\left(\mathbf{W}^l\sigma(\mathbf{x}_i^l) + \mathbf{b}^l\right) + \mathbf{b}^{l+1} \tag{4.21}$$

where $\mathbf{x}_i^l$ and $\mathbf{x}_i^{l+2}$ are input and output features, respectively, and $\mathbf{W}^l, \mathbf{W}^{l+1} \in \mathbb{R}^{F\times F}$ and $\mathbf{b}^l, \mathbf{b}^{l+1} \in \mathbb{R}^F$ are weight and bias parameters. While both variants of residual

blocks are composed of the same number and types of operations, the pre-activation residual block allows a direct connection from input to output without passing through the activation function $\sigma(x)$. For a direct comparison between the original residual block and its pre-activation variant, see Figure 4.21.

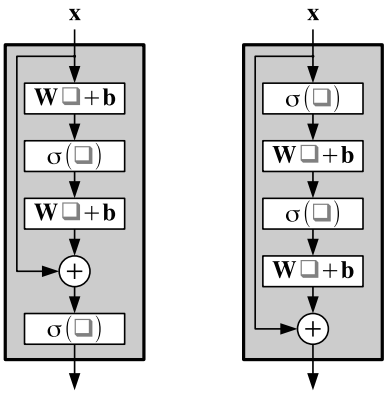


FIGURE 4.21: **Left**: Original, post-activation residual block proposed in [205]. **Right**: Pre-activation residual block proposed in [206].

Interaction block To predict properties that depend on the environment of an atom i, it is important to model interactions with its surrounding atoms j in a chemically meaningful manner. In doing so, it is crucial that all known invariances of the property of interest are respected: For example, the energy of a chemical system is known to be invariant with respect to translation, rotation and permutation of equivalent atoms.[156] Further, since it is a valid assumption that most (but not all) chemical interactions are inherently short-ranged, it is meaningful to introduce a cut-off radius r_{cut}, such that only interactions within the local environment of an atom are considered. Apart from encoding chemical knowledge directly in the modelling of interactions, this approach has the important computational advantage of making NN predictions scale linearly with system size. Based on these design principles, the feature vector $\mathbf{x}$ of an atom is refined by interacting with its local environment through a "message"[161] $\mathbf{v} \in \mathbb{R}^F$ according to

$$\mathbf{x}_i^{l+1} = \mathbf{u}^l \circ \mathbf{x}_i^l + \mathbf{W}^l \sigma(\mathbf{v}_i^l) + \mathbf{b}^l \tag{4.22}$$

where $\mathbf{u}^l, \mathbf{b}^l \in \mathbb{R}^F$ and $\mathbf{W}^l \in \mathbb{R}^{F \times F}$ are parameters and '$\circ$' denotes the Hadamard (entrywise) product. The gating vector $\mathbf{u}$, inspired by the gated recurrent unit,[207] allows individual entries of the feature vector to be damped or reinforced during the update. The final message $\mathbf{v}$ used in Eq. 4.22 is obtained by passing a "proto-message" $\tilde{\mathbf{v}}$ through $N_{\text{residual}}^{\text{interaction}}$ residual blocks (Eq. 4.21), where $\tilde{\mathbf{v}}$ is given by

$$\tilde{\mathbf{v}}_i^l = \sigma\left(\mathbf{W_I}^l \sigma(\mathbf{x}_i^l) + \mathbf{b_I}^l\right) + \sum_{j \neq i} \mathbf{G}^l \mathbf{g}(r_{ij}) \circ \sigma\left(\mathbf{W_J}^l \sigma(\mathbf{x}_j^l) + \mathbf{b_J}^l\right) \tag{4.23}$$

and $\mathbf{W_I}^l, \mathbf{W_J}^l \in \mathbb{R}^{F\times F}$, $\mathbf{b_I}^l, \mathbf{b_J}^l \in \mathbb{R}^F$ and $\mathbf{G}^l \in \mathbb{R}^{F\times K}$ are parameters and r_{ij} denotes the Euclidean distance between atoms i and j. The vector $\mathbf{g}(r_{ij}) = [g_1(r_{ij}) \ \cdots \ g_K(r_{ij})]^\mathsf{T}$ is composed of the values of K RBFs of the form

$$g_k(r_{ij}) = \phi(r_{ij}) \cdot \exp\left(-\beta_k \left(\exp(-r_{ij}) - \mu_k\right)^2\right) \tag{4.24}$$

where $\mu_k, \beta_k \in \mathbb{R}_{>0}$ are parameters that specify centre and width of $g_k(r_{ij})$, respectively, and $\phi(r_{ij})$ is a smooth cut-off function given by[208]

$$\phi(r_{ij}) = \begin{cases} 1 - 6\left(\frac{r_{ij}}{r_\text{cut}}\right)^5 + 15\left(\frac{r_{ij}}{r_\text{cut}}\right)^4 - 10\left(\frac{r_{ij}}{r_\text{cut}}\right)^3 & r_{ij} < r_\text{cut} \\ 0 & r_{ij} \geq r_\text{cut} \end{cases} \tag{4.25}$$

that ensures continuous behaviour when an atom enters or leaves the cut-off sphere. The vector $\mathbf{G}^l\mathbf{g}(r_{ij}) \in \mathbb{R}^F$ takes the role of a learnable attention mask[209] that selects different features based on the pairwise distance r_{ij} between atoms. Note that the Gaussian in Eq. 4.24 takes $\exp(-r_{ij})$ instead of r_{ij} as its argument, which biases attention masks towards a functional form that decays exponentially with r_{ij} (see Figure 4.22).

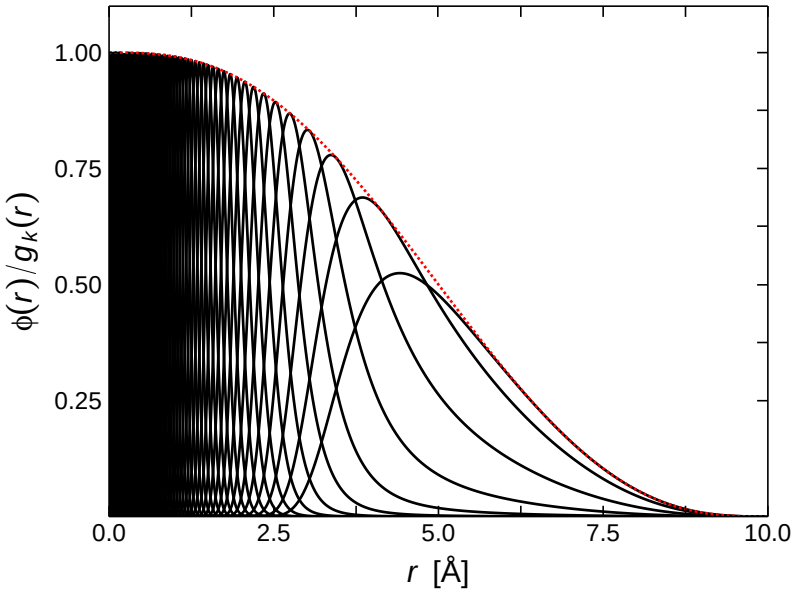


FIGURE 4.22: Cutoff-function $\phi(r)$ (dotted red curve, see Eq. 4.25) and $K = 64$ RBFs $g_k(r)$ (solid black curves, see Eq. 4.24) with β_k and μ_k equal to the initial values given in section 4.3.1.2 and $r_\text{cut} = 10$ Å. For larger r, the basis functions automatically become wider even though all $g_k(r)$ share the same β_k.

Such a bias is meaningful for a chemical system, as it entails the physical knowledge that bound state wave functions in two-body systems decay exponentially.[210] Since

only pairwise distances are used in Eq. 4.23, the output of an interaction block is automatically translationally and rotationally invariant, while the commutative property of summation ensures permutational invariance. The restriction to pairwise distances does not limit the expressiveness of the NN, as higher-order many-body interactions (such as angular dependencies) can be represented by multiple interaction blocks stacked on top of each other.

Output block Each output block passes the atomic features through $N_{\text{residual}}^{\text{output}}$ additional residual blocks and computes the output $\mathbf{y}_i^m \in \mathbb{R}^{n_{\text{out}}}$ of module m for atom i by a linear transformation of the activated features according to

$$\mathbf{y}_i^m = \mathbf{W}_{\text{out}}^m \sigma(\mathbf{x}_i^l) + \mathbf{b}_{\text{out}}^m \tag{4.26}$$

where $\mathbf{W}_{\text{out}}^m \in \mathbb{R}^{F \times n_{\text{out}}}$ and $\mathbf{b}_{\text{out}}^m \in \mathbb{R}^{n_{\text{out}}}$ are weight and bias parameters. How many entries the output vector $\mathbf{y}_i^m$ has depends on how many atomic properties are predicted at once. Two variants of NNs are considered: The first variant predicts atomic energy contributions along with atomic partial charges (i.e. $\mathbf{y}_i^m = [E_i^m \; q_i^m]^\mathsf{T}$ and $n_{\text{out}} = 2$), whereas the second variant predicts just atomic energy contributions (i.e. $\mathbf{y}_i^m = [E_i^m]$ and $n_{\text{out}} = 1$). In principle, other properties could be predicted as well.

Final prediction The final atomic properties are obtained by

$$\mathbf{y}_i = \mathbf{s}_{Z_i} \circ \left(\sum_{m=1}^{N_{\text{module}}} \mathbf{y}_i^m \right) + \mathbf{c}_{Z_i} \tag{4.27}$$

where $\mathbf{s}_Z, \mathbf{c}_Z \in \mathbb{R}^{n_{\text{out}}}$ are element-specific scale and shift parameters chosen according to the nuclear charge Z_i of atom i. The scaling and shifting of the output decouples the values of other parameters of the NN from the numeric range of target properties, which depends mainly on the chosen system of units. Element-specific (instead of global) parameters are motivated by a previous observation indicating that atomic properties of distinct elements can span vastly different ranges (see Figure 4.13 in section 4.2.2.1). The final prediction for the total energy of a system of interest composed of N atoms is obtained by summation of the atomic energy contributions E_i (Eq. 4.28).

$$E = \sum_{i=1}^{N} E_i \tag{4.28}$$

Analytical derivatives of E with respect to the Cartesian coordinates $\{\mathbf{r}_1, \ldots, \mathbf{r}_N\}$ of the atoms, for example to derive the forces $\mathbf{F}_i$ acting on each atom i for MD simulations, are readily obtained by reverse mode automatic differentiation.[211]

A potential shortcoming of Eq. 4.28 is the fact that all long-range interactions contributing to E beyond the cut-off radius r_{cut}, such as electrostatic interactions,

cannot be properly modelled by the NN. Fortunately, their functional form is known and they can be explicitly included when computing E. Other types of long-range interactions for which the functional form is also known analytically, for example dispersion corrections like DFT-D3,[202] can be included as well. Because of the shortcomings of Eq. 4.28, for NNs that also predicts atomic partial charges, E is calculated by

$$E = \sum_{i=1}^{N} E_i + k_e \sum_{i=1}^{N} \sum_{j>i}^{N} \tilde{q}_i \tilde{q}_j \chi(r_{ij}) + E_{\mathrm{D3}} \tag{4.29}$$

instead, where E_{D3} is the DFT-D3 dispersion correction,[202] k_e is the Coulomb constant, and $\tilde{q}_i$ and $\tilde{q}_j$ are the corrected partial charges of atoms i and j. They are obtained from the partial charges q_i predicted by the NN according to

$$\tilde{q}_i = q_i - \frac{1}{N}\left(\sum_{j=1}^{N} q_j - Q\right) \tag{4.30}$$

where Q is the total charge of the system. As it is not guaranteed *a priori* that the sum of all predicted atomic partial charges q_i is equal to the total charge Q of the system (although the result is usually very close when the NN is properly trained), a correction scheme like Eq. 4.30 is necessary to guarantee charge conservation. The function $\chi(r_{ij})$

$$\chi(r_{ij}) = \phi(2r_{ij}) \frac{1}{\sqrt{r_{ij}^2 + 1}} + \left(1 - \phi(2r_{ij})\right) \frac{1}{r_{ij}} \tag{4.31}$$

smoothly interpolates between the correct r_{ij}^{-1} dependence of the Coulomb law at long-range and a damped term at small distances to avoid the singularity at $r_{ij} = 0$ (see Figure 4.23). The summation over all atom pairs when evaluating the long-range interactions in Eq. 4.29 makes the evaluation of E scale quadratically with system size. Fortunately, various schemes to recover linear scaling are described in the literature, e.g. Ewald summation[212] or cut-off methods[213], and can be applied without modification.

The concept of augmenting an NN-PES with long-range interactions was first proposed in [197]. However, most previous works use a separately trained NN to predict atomic partial charges.[160,197] Here the same NN is used to predict both, atomic energy contributions and partial charges (see Eqs. 4.26 and 4.27). Aside from computational advantages (only a single NN needs to be trained and evaluated), shared feature representations in such "multi-task learning" are believed to increase the generalization capabilities of NNs.[214–216]

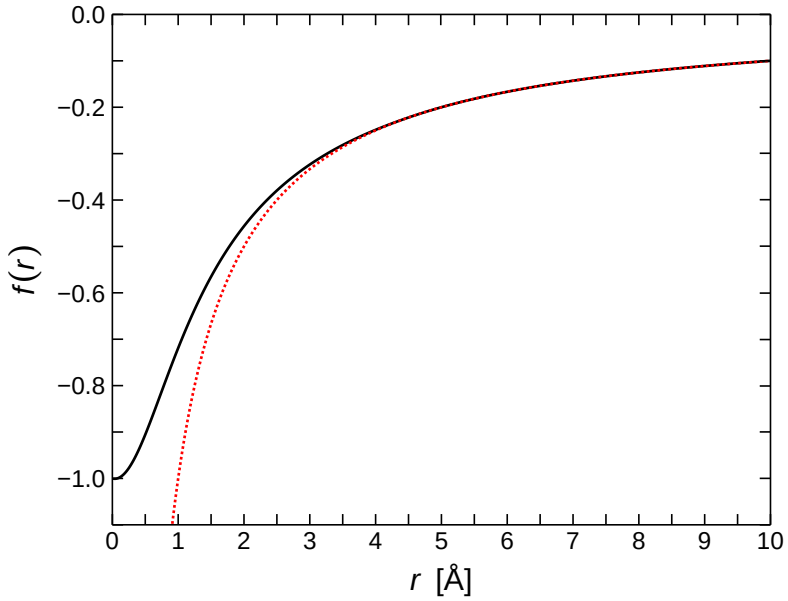


FIGURE 4.23: The dotted red curve shows the correct $\frac{1}{r}$ dependence of the Coulomb law, whereas the solid black curve shows $\chi(r)$ (Eq. 4.31) for $r_{\text{cut}} = 10$ Å. For $r > \frac{r_{\text{cut}}}{2}$, both curves are identical by construction.

Apart from allowing the computation of long-range electrostatic interactions, the partial charges $\tilde{q}_i$ can also be used to predict the electric dipole moment $\mathbf{p}$ of a structure according to

$$\mathbf{p} = \sum_{i=1}^{N} \tilde{q}_i \mathbf{r}_i \tag{4.32}$$

where $\mathbf{r}_i$ are the Cartesian coordinates of atom i. The ability to predict $\mathbf{p}$ is useful for example for the calculation of infrared spectra.[200,217]

Hyperparameters The NN can be tuned by hyperparameters that control its width and depth, i.e. how many neurons each layer contains and how many layers are stacked on top of each other. While it would be possible to optimize hyperparameters for individual learning tasks, for example via a grid search, it was found that this is not necessary for good performance. For simplicity, all NN applied in section 4.3.2 share the same architecture with the hyperparameters summarized in Table 4.7, unless specified otherwise.

Summary The NN architecture is schematically shown in Figure 4.24 and summarized below:

A: Overview. The input nuclear charges Z_i of N atoms are transformed to feature vectors $\mathbf{x}_i \in \mathbb{R}^F$ via an embedding layer (purple, Eq. 4.20) and passed iteratively through a stack of N_{module} modular building blocks (green). From the input

TABLE 4.7: Hyperparameter values controlling the NN architecture. F determines the dimensionality of feature space, K the number of RBFs, N_{module} the number of stacked modules, $N_{\text{residual}}^{\text{atomic}}$, $N_{\text{residual}}^{\text{interaction}}$ and $N_{\text{residual}}^{\text{output}}$ the number of residual blocks in different module components, and r_{cut} the cut-off radius.

hyperparameter	**value**
F	128
K	64
N_{module}	5
$N_{\text{residual}}^{\text{atomic}}$	2
$N_{\text{residual}}^{\text{interaction}}$	3
$N_{\text{residual}}^{\text{output}}$	1
r_{cut}	10 Å

Cartesian coordinates $\mathbf{r}_i$ of the atoms, all pairwise distances within a cut-off radius r_{cut} are calculated and expanded in a set of K RBFs (yellow, Eq. 4.24) forming the entries of the vectors $\mathbf{g}(r_{ij}) \in \mathbb{R}^K$, which are additional inputs to each module. The output of all modules is summed to form the final atom-wise predictions of the NN, e.g. atomic energy contributions E_i and partial charges q_i (Eq. 4.27).

B: Modular building block. Each module transforms its input through an interaction block (blue) followed by $N_{\text{residual}}^{\text{atomic}}$ residual blocks (grey). The computation then splits into two branches: One branch transforms the input further through an output block (red) to form the module output, whereas the other branch passes the transformed input directly to the next module in the hierarchy.

C: Interaction block. After passing through the activation function $\sigma(x)$, the incoming features of the central atom i and neighbouring atoms j split paths and are further refined through separate layers. The attention mask $\mathbf{Gg}(r_{ij})$ selects features of atoms j based on their distance to atom i and adds them to its features in order to compute the proto-message $\tilde{\mathbf{v}}$ (Eq. 4.23), which is refined through $N_{\text{residual}}^{\text{interaction}}$ residual blocks (grey) to the message $\mathbf{v}$. After an additional activation and linear transformation, the message, which represents the interactions between atoms, is added to the gated feature representations $\mathbf{u} \circ \mathbf{x}$ (Eq. 4.22).

D: Output block. Each output block passes its input through $N_{\text{residual}}^{\text{output}}$ residual blocks (grey) and a layer with linear activation to compute the final output of a module (Eq. 4.26).

E: Residual block. Each residual block refines its input by adding a residual computed by a shallow NN with two layers (Eq. 4.21).

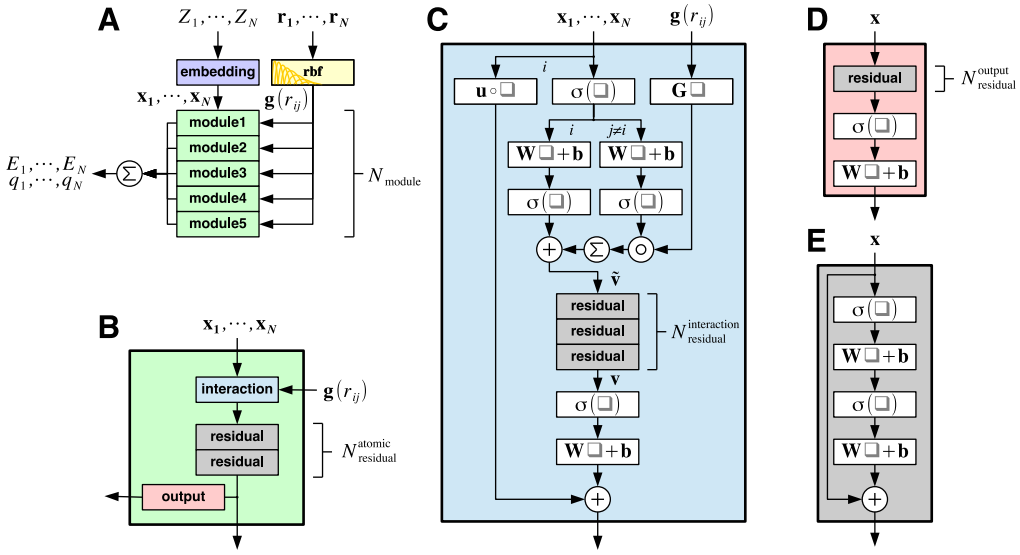


FIGURE 4.24: Schematic representation of the NN architecture. **A**: Overview. **B**: Modular building block. **C**: Interaction block. **D**: Output block. **E**: Residual block.

4.3.1.2 Training

Before the training of an NN starts, its parameters need to be initialized. All entries of the embedding vectors $\mathbf{e}_Z$ are initialized with random values uniformly distributed between $-\sqrt{3}$ and $\sqrt{3}$ (such that they have unit expected variance) and the weight matrices $\mathbf{W}$, $\mathbf{W_I}$ and $\mathbf{W_J}$ are initialized to random orthogonal matrices with entries scaled such that their variance corresponds to the value recommended in the Glorot initialization scheme.[139] The entries of all bias vectors $\mathbf{b}$, $\mathbf{b_I}$, $\mathbf{b_J}$, $\mathbf{b}_{\text{out}}$, and matrices $\mathbf{G}$ and $\mathbf{W}_{\text{out}}$ are initialized to zero, whereas the entries of the gating vectors $\mathbf{u}$ are initialized to one. The centres μ_k of the RBFs are set to K equally spaced values between $\exp(-r_{\text{cut}})$ and 1 and their widths β_k are initialized to $\left(2K^{-1}\left(1-\exp(-r_{\text{cut}})\right)\right)^{-2}$ (see Figure 4.22).

After initialization, the parameters of the NN are optimized by minimizing a loss function $\mathcal{L}$ using AMSGrad[218] with a learning rate of 10^{-3} (other hyperparameters of the optimizer are set to the default values recommended in [218]) and a batch size of 32 reference structures. Depending on which variant of NN is used, the loss term $\mathcal{L}$ for a reference structure with N atoms is defined as

$$\mathcal{L} = w_E \left|E - E^{\text{ref}}\right| + \frac{w_F}{3N} \sum_{i=1}^{N} \sum_{\alpha=1}^{3} \left| -\frac{\partial E}{\partial r_{i,\alpha}} - F_{i,\alpha}^{\text{ref}} \right| + \mathcal{L}_{\text{nh}} \tag{4.33}$$

for an NN that just predicts atomic energy contributions and as

$$\mathcal{L} = w_E \left| E - E^{\text{ref}} \right| + \frac{w_F}{3N} \sum_{i=1}^{N} \sum_{\alpha=1}^{3} \left| -\frac{\partial E}{\partial r_{i,\alpha}} - F_{i,\alpha}^{\text{ref}} \right| + w_Q \left| \sum_{i=1}^{N} q_i - Q^{\text{ref}} \right| + \frac{w_p}{3} \sum_{\alpha=1}^{3} \left| \sum_{i=1}^{N} q_i r_{i,\alpha} - p_\alpha^{\text{ref}} \right| + \mathcal{L}_{\text{nh}} \tag{4.34}$$

for an NN that also predicts atomic partial charges q_i. Here, E^{ref} and Q^{ref} are reference energy and total charge, p_α^{ref} are the Cartesian components of the reference dipole moment $\mathbf{p}^{\text{ref}}$, $F_{i,\alpha}^{\text{ref}}$ are the Cartesian components of the reference force $\mathbf{F}_i^{\text{ref}}$ acting on atom i, and $r_{i,\alpha}$ is the αth Cartesian coordinate of atom i. The energy prediction E of the NN is given either by Eq. 4.28 or Eq. 4.29, depending on which variant is used.

The weighting hyperparameters w_E, w_F, w_Q and w_p determine the relative contribution of the individual error terms to the loss term. Note that the numeric ranges of the error terms (and therefore their contributions to $\mathcal{L}$) also depend on the chosen system of units. For simplicity, weighting hyperparameters are not optimized for individual learning tasks and instead always set to $w_E = w_Q = w_p = 1$ and $w_F = 10^2$ (when all quantities are measured in atomic units). The higher relative weight of force errors is motivated by the fact that forces alone determine the dynamics of a chemical system and accurate force predictions are therefore most important for MD simulations. For datasets where any of the reference quantities used in Eqs. 4.33 and 4.34 are not available, the corresponding weight is set to zero.

The term $\mathcal{L}_{\text{nh}}$ is a "non-hierarchicality penalty", inspired by a similar regularization method introduced in [165], given either by

$$\mathcal{L}_{\text{nh}} = \frac{\lambda_{\text{nh}}}{N} \sum_{i=1}^{N} \sum_{m=2}^{N_{\text{module}}} \frac{1}{2} \left[\frac{(E_i^m)^2}{(E_i^m)^2 + (E_i^{m-1})^2} + \frac{(q_i^m)^2}{(q_i^m)^2 + (q_i^{m-1})^2} \right] \tag{4.35}$$

or

$$\mathcal{L}_{\text{nh}} = \frac{\lambda_{\text{nh}}}{N} \sum_{i=1}^{N} \sum_{m=2}^{N_{\text{module}}} \frac{(E_i^m)^2}{(E_i^m)^2 + (E_i^{m-1})^2} \tag{4.36}$$

depending on the variant of NN used, and λ_{nh} is the corresponding regularization hyperparameter. The $\mathcal{L}_{\text{nh}}$ term penalizes when the predictions of individual modules do not decay with increasing depth in the hierarchy. Since deeper feature representations of atoms capture increasingly higher-order interactions, such a regularization is motivated by the fact that higher-order terms in many-body expansions of the energy are known to decay rapidly. For simplicity, λ_{nh} is not tuned for individual learning tasks and instead always set to 10^{-2}.

During the training process, an exponential moving average of all parameter values is kept using a decay rate of 0.999. Overfitting is prevented using early stopping:[219] After every epoch (one pass over all reference structures in the training set), the loss function is evaluated on a validation set of reference structures using the exponentially averaged parameters. After training, the NN parameters that performed best on the validation set are selected. Since the validation set is used indirectly during the training procedure, the final performance (see section 4.3.2) is always measured on a separate test set.

Only true quantum mechanical observables, such as total energy, forces or dipole moments, are used as reference when training the NN (see Eqs. 4.33 and 4.34). While it would also be possible to train the NN to directly reproduce atomic energies and partial charges obtained using a decomposition method,[167–169,198] such schemes are essentially arbitrary and it is unclear whether the corresponding decompositions are meaningful. Further, it is not always guaranteed that the quantities obtained from such methods vary smoothly when the molecular geometry changes, which makes it difficult for an NN to learn them. By only relying on quantum mechanical observables, the NN automatically learns to perform a smooth decomposition in a data-driven way.

4.3.1.3 Dataset generation

In the following, the generation of two new benchmark datasets, which probe chemical reactivity, long-range electrostatics, and many-body molecular interactions, is described.

S_N2 reactions The S_N2 reactions dataset probes chemical reactions of the kind $X^- + H_3C–Y \rightarrow X–CH_3 + Y^-$ and contains structures for all possible combinations of $X, Y \in \{F, Cl, Br, I\}$. It consists of different geometries for the high-energy transition regions, ion-dipole bound state complexes and long-range (> 10 Å) interactions of CH_3X molecules with Y^- ions. The dataset also includes various structures for several smaller molecules that can be formed in fragmentation reactions, such as CH_3X, HX, CHX or CH_2X^- with $X \in \{F, Cl, Br, I\}$, as well as geometries for H_2, CH_2, CH_3^+ and XY interhalogen compounds for all possible combinations of $X, Y \in \{F, Cl, Br, I\}$. In total, the dataset provides reference energies, forces and dipole moments for 452 709 structures calculated at the DSD-BLYP-D3(BJ)/def2-TZVP level of theory.[202,220–222]

Different conformations for each species present in the S_N2 reactions dataset were sampled by running MD simulations at a temperature of 5000 K with a time step of 0.1 fs using the Atomic Simulation Environment (ASE).[223] The necessary forces were obtained with the semi-empirical PM7 method[224] implemented in MOPAC2016.[225] Structures were saved every 10 steps and for each of them, reference energies, forces and dipole moments were calculated at the DSD-BLYP-D3(BJ)/def2-TZVP[202,220–222] level of theory using the ORCA 4.0.1 code.[226,227] The DSD-BLYP functional is one

of the best performing double hybrid methods in the GMTKN55 benchmark.[228] All MD simulations were started from the PM7-optimized geometries. For the reaction complexes $[XCH_3Y]^-$, MD simulations were randomly started either from the respective van der Waals complexes, or, to better sample the transition regions, from the transition state (calculated with PM7) of the respective reaction. Further, long-range interactions were sampled by choosing a random conformation from the CH_3X MD simulations and randomly placing an ion Y^- in the vicinity of the CH_3X molecule such that its distance to any other atom is at most 16 Å. Table 4.8 lists how many structures are present for each species.

TABLE 4.8: Number of structures for each species present in the S_N2 reactions dataset.

species	count	species	count	species	count
$[FCH_3Cl]^-$	44 501	CH_3I	3500	HCl	3500
$[FCH_3Br]^-$	44 501	CH_3^+	3500	HBr	3500
$[FCH_3I]^-$	44 501	CH_2F^-	3500	HI	3500
$[ClCH_3Br]^-$	44 501	CH_2Cl^-	3500	F_2	2000
$[ClCH_3I]^-$	44 501	CH_2Br^-	3500	FCl	2000
$[BrCH_3I]^-$	44 501	CH_2I^-	3500	FBr	2000
$[FCH_3F]^-$	24 801	CH_2	3500	FI	2000
$[ClCH_3Cl]^-$	24 801	CHF	3500	Cl_2	1999
$[BrCH_3Br]^-$	24 801	CHCl	3500	ClBr	2000
$[ICH_3I]^-$	24 801	CHBr	3500	ClI	2000
CH_3F	3500	CHI	3500	Br_2	2000
CH_3Cl	3500	H_2	3500	BrI	2000
CH_3Br	3500	HF	3500	I_2	2000

Solvated protein fragments The solvated protein fragments dataset probes many-body intermolecular interactions between "protein fragments" and water molecules, which are important for the description of many biologically relevant condensed phase systems. It contains structures for all possible "amons"[229] (hydrogen-saturated covalently bonded fragments) of up to eight heavy atoms (C, N, O, S) that can be derived from possible chemical graphs of proteins containing the 20 natural amino acids[a] connected via peptide bonds or disulfide bridges. For amino acids that can occur in different charge states due to (de-)protonation, all possible structures with up to a total charge of $\pm 2e$ are included. These structures are augmented with solvated variants containing a varying number of water molecules such that the total number of heavy atoms does not exceed 21. The dataset also contains randomly sampled dimer interactions of protein fragments, as well as structures of pure water

[a]The natural amino acids are alanine, arginine, asparagine, aspartic acid, cysteine, glutamine, glutamic acid, glycine, histidine, isoleucine, leucine, lysine, methionine, phenylalanine, proline, serine, threonine, tryptophan, tyrosine, and valine.

with up to 40 molecules. In total, it contains reference energies, forces and dipole moments for 2 731 180 structures calculated at the revPBE-D3(BJ)/def2-TZVP level of theory.[202,221,222,230] On average, the structures contain 21 atoms (with a maximum of 120 atoms) and consist of 63% hydrogen, 19% carbon, 12% oxygen, 5% nitrogen, and 1% sulphur atoms.

Available benchmark datasets cover conformational and chemical degrees of freedom, but they usually do not probe many-body intermolecular interactions, which are important in the description of condensed phase systems. Due to their biological importance, proteins in aqueous solution are a particularly relevant system of this kind. However, even small proteins contain hundreds of atoms, which makes *ab initio* reference calculations for them prohibitively expensive. Fortunately, it is possible to construct a predictive machine learning (ML) model for large molecules by training it only on smaller molecules that are structurally similar.[229] These so-called "amons" can be readily constructed by considering a large molecule as chemical graph, generating all possible connected subgraphs with a fixed number of heavy atoms and saturating the resulting structures with hydrogen atoms. For a more detailed description, refer to [229]. Due to the fact that most proteins are comprised of just 20 different amino acids, many bonding patterns are shared between proteins and a relatively small number of amons is sufficient to cover all possibilities.

The different structures in the dataset were constructed as follows: All amons with up to eight heavy atoms (C, N, O, S) were constructed according to the method described in [229] for all possible chemical graphs of proteins containing the 20 natural amino acids connected via peptide bonds or disulfide bridges. For amons derived from amino acids that can occur in different charge states due to (de-)protonation, all variants with up to a total charge of $\pm 2e$ are included. This results in 2307 different molecules. In order to sample interactions with solvent molecules, the amon structures were augmented by randomly placing up to 20 water molecules in their vicinity, such that the total number of heavy atoms does not exceed 21. This results in 29 991 additional structures. Further, interactions between different amons were sampled by generating all possible dimers from amons with up to 3 heavy atoms resulting in 867 possible combinations. Important interactions between different amino acids were included by adding sidechain-sidechain and backbone-backbone complexes from the BioFragment Database[231] (3480 structures). Further, interactions between water molecules were sampled by constructing water clusters with up to 21 molecules. Each water cluster is complemented by a variant with an additional proton, as well as with a variant lacking one proton in order to sample the different possible charge states of water. This results in 24 additional structures.

All structures were optimized using the semi-empirical PM7 method[224] implemented in MOPAC2016.[225] Starting from the optimized geometry, 100 different conformations for each structure were sampled by running MD simulations (at the same level of theory) at a temperature of 1000 K with a time step of 0.1 fs using the ASE.[223] Structures were saved every 10 steps and for each of them, reference energies, forces and dipole moments were calculated at the revPBE-D3(BJ)/def2-TZVP[202,221,222,230] level of theory using the ORCA 4.0.1 code.[226,227] The revPBE functional is one of the best performing GGA functionals in the GMTKN55 benchmark.[228]

While this initial dataset already covers many different chemical situations, it is not guaranteed that the contained structures cover chemical and configurational space sufficiently well to account for all situations that might be relevant in MD simulations. For this reason, the initial dataset was iteratively augmented using an adaptive sampling method:[170,232] An ensemble of three NNs trained (see section 4.3.1.2) on the initial dataset is used to run MD simulations and all structures for which the NN predictions have a standard deviation larger than a threshold value (here 1 kcal mol^{-1}) are saved. For each structure saved in this process, energies, forces and dipole moments were calculated with the reference *ab initio* method and added to the dataset. Afterwards, the NNs were retrained and the sampling process was repeated. In total, the dataset was adaptively augmented in this way for four times, after which significant deviations between NN predictions were found to be rare. Finally, energies, forces and dipole moments were calculated with the reference method for 10 000 structures of 40 water molecules in a spherical arrangement (obtained by running MD simulations with PM7, see above) to include training examples similar to bulk phase water. The final dataset contains data for 2 731 180 structures.

4.3.2 Results

In this section, the NN described in section 4.3.1.1 is applied to various quantum-chemical datasets that all probe different aspects of chemical space (i.e. chemical and/or conformational degrees of freedom). Apart from the well-established benchmarks QM9,[166] MD17[116] and ISO17,[163] the NN is applied to the two new datasets introduced in section 4.3.1.3 that probe chemical reactivity and many-body intermolecular interactions.

QM9 The QM9 dataset[166] is a widely used benchmark for the prediction of several properties of molecules in equilibrium. It consists of geometric, energetic, electronic, and thermodynamic properties for ≈134 000 small organic molecules made up of H, C, O, N, and F atoms. These molecules correspond to the subset of all species with up to nine heavy atoms (C, O, N, and F) out of the GDB-17 chemical universe database.[181] All properties were calculated at the B3LYP/6-31G(2df,p) level of theory. About 3000 molecules within QM9 fail a geometric consistency check or were difficult to converge[166] and are commonly removed from the dataset.[161,163,165,233]

Since the QM9 dataset contains only equilibrium geometries, the benchmark probes just chemical degrees of freedom.

Table 4.9 lists the performance of various ML models published in the literature for predicting the total energy (property U_0) on the pruned QM9 dataset (≈131 000 structures) for different training set sizes. Results for the NN are averaged over five independent runs using the same training set. The performance can be further improved by *bagging*:[234] An ensemble of five trained NNs significantly improves upon the performance of a single NN.

TABLE 4.9: MAEs in kcal mol^{-1} for energy predictions in the QM9 dataset for the DTNN,[162] SchNet[163,235] (best reported values are given), HIP-NN,[165] and the NN introduced in section 4.3.1.1 (*) for different training set sizes $N_{\text{train}} + N_{\text{valid}}$. The performance of an ensemble[234] of five NNs is also reported (**). The best results are shown in bold.

$N_{\text{train}} + N_{\text{valid}}$	**DTNN**	**SchNet**	**HIP-NN**	*	**
110 426	—	0.26	0.26	0.19	**0.14**
100 000	0.84	0.34	0.26	0.19	**0.14**
50 000	0.94	0.49	0.35	0.30	**0.24**

MD17 The MD17 dataset[116] is a collection of structures, energies and forces from *ab initio* MD simulations of eight small organic molecules. All trajectories were calculated at a temperature of 500 K and a resolution of 0.5 fs using the PBE+vdW-TS electronic structure method.[182,183] The datasets range in size from 150 000 to almost 1 000 000 conformations and cover energy ranges between 20 to 48 kcal mol^{-1} and force components between 266 to 570 kcal mol^{-1} Å^{-1}. The task is to predict energies (and forces) using a separate model for each molecule. Since each task is limited to a single molecule, the MD17 benchmark probes only conformational degrees of freedom.

Table 4.10 lists the performance of the DTNN,[162] HIP-NN,[165] GDML[116] and SchNet[163] on the MD17 benchmark and compares them to the performance of the NN introduced in section 4.3.1.1 (averaged over five independent runs). For a fair comparison between different models, it should be noted that while all of them are trained on a total of 50 000 structures (combined training and validation sets), they use different subsets of the available data for training: DTNN and HIP-NN are trained on energies only, GDML is trained on forces only, and the remaining models are trained on both energies and forces.

ISO17 The ISO17 dataset[163] consists of short MD trajectories of 127 isomeric molecules with the composition $C_7O_2H_{10}$ drawn randomly from the largest set of isomers in QM9. Each trajectory samples 5000 conformations at a resolution of

TABLE 4.10: MAEs for predictions of energy (in kcal mol^{-1}) and forces (in kcal mol^{-1} $Å^{-1}$) for molecules in the MD17 dataset for the DTNN,[162] HIP-NN,[165] GDML,[116] SchNet[163,235] (best reported values are given) and the NN introduced in section 4.3.1.1 (*). The performance of an ensemble of five NNs is also reported (**). The best results in each category are shown in bold.

		DTNN	**HIP-NN**	**GDML**	**SchNet**	*	**
Aspirin	*energy*	—	—	0.13	**0.12**	**0.12**	**0.12**
	forces	—	—	**0.02**	0.33	0.06	0.04
Benzene	*energy*	**0.04**	0.06	0.07	0.07	0.07	0.07
	forces	—	—	0.24	0.17	0.15	**0.14**
Ethanol	*energy*	—	—	**0.05**	**0.05**	**0.05**	**0.05**
	forces	—	—	0.09	0.05	0.03	**0.02**
Malonaldehyde	*energy*	0.19	0.09	0.08	**0.07**	**0.07**	**0.07**
	forces	—	—	0.09	0.08	0.04	**0.03**
Naphthalene	*energy*	—	—	0.12	**0.11**	0.12	0.12
	forces	—	—	**0.03**	0.11	0.04	**0.03**
Salicylic acid	*energy*	0.41	0.20	0.11	**0.10**	0.11	0.11
	forces	—	—	**0.03**	0.19	0.04	**0.03**
Toluene	*energy*	0.18	0.14	**0.09**	**0.09**	0.10	0.10
	forces	—	—	0.05	0.09	**0.03**	**0.03**
Uracil	*energy*	—	—	0.11	**0.10**	**0.10**	**0.10**
	forces	—	—	**0.03**	0.11	**0.03**	**0.03**

1 fs. In total, the dataset contains 635 000 structures for which energies and forces, calculated at the PBE+vdW-TS level of theory,[182,183] are reported. The task is to predict energies (and forces) for two different scenarios: In the first variant (known molecules/unknown conformations), the training set contains ≈80% of all molecules and conformations (400 000 structures for training and 4000 structures for validation) and the task is to predict the remaining ≈20% of conformations for the same subset of molecules present in the training set (101 000 structures). Thus, this variant of the benchmark tests the generalization capabilities of the model for unknown conformations of previously seen molecules. In the second, more challenging variant (unknown molecules/unknown conformations), the training set remains the same, but the task is to predict all 5000 conformations of the ≈20% of molecules not present in the training set (130 000 structures). Here, generalization capabilities of the model are tested for unknown conformations of unknown molecules. Both variants of the ISO17 benchmark probe chemical and conformational degrees of freedom.

Table 4.11 compares the performance of SchNet[163] with the NN introduced in section 4.3.1.1 (averaged over five independent runs) for the two variants of the ISO17 benchmark. The performance of an ensemble of five NNs is also reported. All models were trained on both energy and force information.

TABLE 4.11: MAEs for predictions of energy (in kcal mol^{-1}) and forces (in kcal mol^{-1} $Å^{-1}$) for the two variants of the ISO17 benchmark. The NN introduced in section 4.3.1.1 (*) is compared with SchNet.[163] The performance of an ensemble[234] of five NNs is also reported (**). The best results are shown in bold.

		SchNet	*	**
known molecules /	*energy*	0.36	**0.10**	**0.10**
unknown conformations	*forces*	1.00	0.12	**0.08**
unknown molecules /	*energy*	**2.40**	2.94	2.86
unknown conformations	*forces*	2.18	1.38	**1.13**

S_N2 reactions For a detailed description of the dataset, see section 4.3.1.3. The task is to predict energies, forces and dipole moments using a single model for all structures contained in the dataset, testing the generalization capabilities of the model across chemical and conformational degrees of freedom, chemical reactions, and challenging long-range intermolecular interactions.

Table 4.12 lists the performance of the NN described in section 4.3.1.1 with and without explicit long-range electrostatic interactions (see Eqs. 4.29 and 4.28). The results in each case are averaged over five independent runs and the performance of ensembles of five NNs is also reported. All models were trained on 400 000 structures with 5000 structures used for validation. Because of the partial charge correction

scheme (see Eq. 4.30), total charge is always predicted exactly. However, for completeness, the error for the prediction of total charge (in e) using the uncorrected partial charges is also given.

TABLE 4.12: MAEs for predictions of energy (in kcal mol^{-1}), forces (in kcal mol^{-1} Å^{-1}), dipole moments (in D), and total charge (in e) for the S_N2 reactions dataset for NNs with ("with l.r.", Eq. 4.29) and without ("no l.r.", Eq. 4.28) electrostatic long-range augmentation (*). The performance of an ensemble[234] of five NNs is also reported (**). The best results in each category are shown in bold.

	* **no l.r.**	** **no l.r.**	* **with l.r.**	** **with l.r.**
energy	0.071	0.070	**0.009**	**0.009**
forces	0.035	0.032	0.012	**0.009**
dipole	—	—	0.0044	**0.0042**
charge	—	—	0.000 23	**0.000 19**

The NN without the explicit inclusion of long-range interactions (Eq. 4.28) performs significantly worse. This is to be expected, as for this dataset, ion-dipole interactions, which decay with the square of the distance, play an important role for determining the overall energy. As their influence extends well beyond the cut-off distance (here 10 Å), an NN without long-range augmentation cannot properly account for them. This effect is also seen in Figure 4.25, which shows minimum energy paths (MEPs) for all S_N2 reactions of the kind $X^- + H_3C–Y \rightarrow X–CH_3 + Y^-$ covered in the dataset[a] along the reaction coordinate defined by the distance difference $r_{CY} - r_{CX}$.

While the NN including explicit long-range interactions (Eq. 4.29) is able to reproduce the reference energies accurately across the whole range of values of the reaction coordinate (apart from minor deviations in the asymptotics), the NN without long-range interactions (Eq. 4.28) shows qualitatively wrong asymptotic behaviour (see Figure 4.26). A correct description of the asymptotics is crucial for quantitative predictions of reaction rates with MD simulations, as errors can strongly influence the maximum impact parameter for collisions at which a reaction is still possible. Note that the MEPs calculated with the reference *ab initio* method were not included in the training data.

Solvated protein fragments For a detailed description of the dataset, see section 4.3.1.3. The task is to predict all properties using a single model, which tests the generalization capabilities across chemical and conformational degrees of freedom in gas and solution phase, proton transfer and challenging many-body intermolecular

[a] All possible combinations X–Y with $X, Y \in \{F, Cl, Br, I\}$.

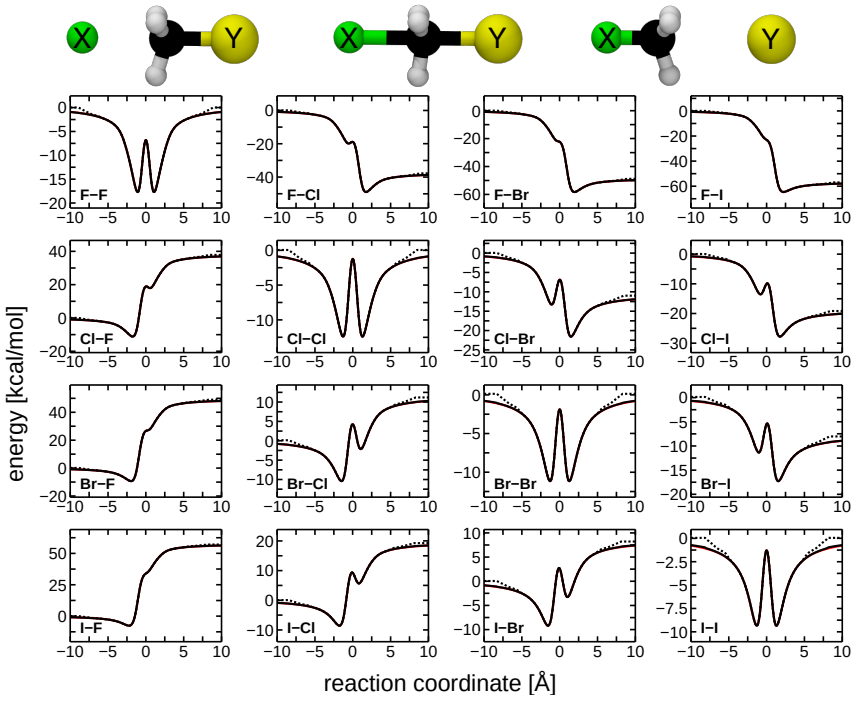


FIGURE 4.25: MEPs for S_N2 reactions (different combinations of X–Y), calculated using an ensemble of five NNs with (solid black) and without (dotted black) explicit long-range interactions. The solid red line (mostly occluded by the solid black line) depicts the MEP calculated using the reference *ab initio* method.

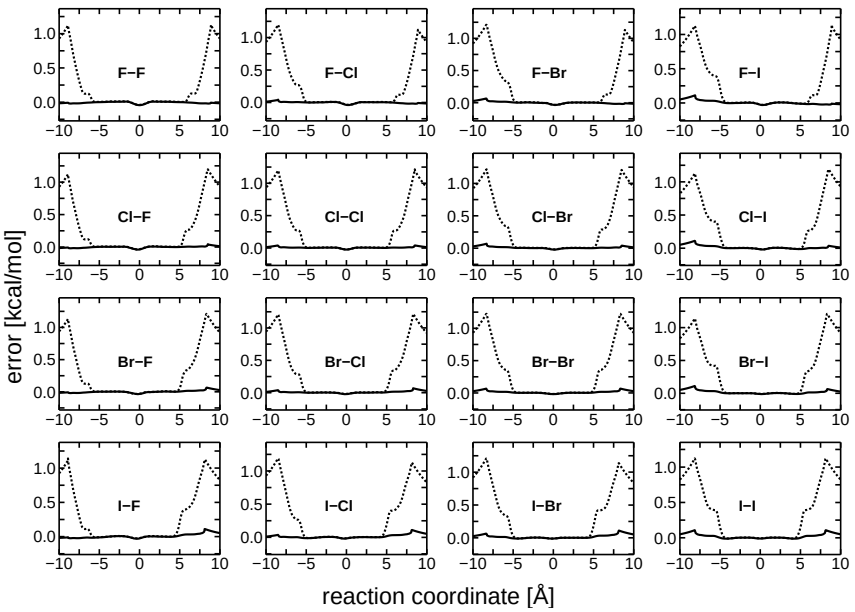


FIGURE 4.26: Energy prediction errors for S_N2 reactions (different combinations of X–Y), calculated using an ensemble of five NNs with (solid black) and without (dotted black) explicit long-range interactions. The model without long-range interactions displays significant errors ($\approx$ 1kcal mol^{-1}) in the asymptotic regions of the PES.

interactions.

Table 4.13 lists the performance of the NN described in section 4.3.1.1 (averaged over five runs) and an ensemble consisting of five NNs. All models were trained on 2 560 000 structures with 100 000 structures used for validation. For completeness, the error for the prediction of total charge using the uncorrected partial charges is also given. Because the solvated protein fragments dataset contains structures with widely different numbers of atoms (up to 120), Table 4.13 also reports the MAE for energy predictions per atom.

Since non-covalent interactions play a crucial rule for the structure of large systems like proteins, the ensemble of NNs was also used to predict interaction energies for sidechain-sidechain interactions (SSIs) (3380 structures) and backbone-backbone interactions (BBIs) (100 structures) in the BioFragment Database[231] and compared to reference *ab initio* values. For each case, interaction energies were determined by subtracting monomer energies from the energy of the complex. The NN ensemble predictions correlate well with the reference values (see Figure 4.27) and have MAEs of 0.28 kcal mol^{-1} and 0.21 kcal mol^{-1} for the SSI and BBI complexes, respectively. Note that although structures in the BioFragment Database are included in the data used for training the NNs (see section 4.3.1.3), the dataset contains only total energies and NNs were therefore never directly trained to reproduce interaction

TABLE 4.13: MAEs for predictions of energy and energy per atom (in kcal mol^{-1}), forces (in kcal mol^{-1} Å^{-1}), dipole moments (in D), and total charge (in e) for the solvated protein fragments dataset for the NN described in section 4.3.1.1 (*) augmented with long-range interactions (Eq. 4.29). The performance of an ensemble[234] of five NNs is also reported (**). The best results in each category are shown in bold.

	*	**
energy	1.03	**0.95**
energy/atom	0.054	**0.050**
forces	0.88	**0.72**
dipole	0.060	**0.054**
charge	0.004	**0.003**

energies. Despite this fact, NNs are able to learn a meaningful decomposition of the total energy into intramolecular and intermolecular contributions and predict interaction energies accurately.

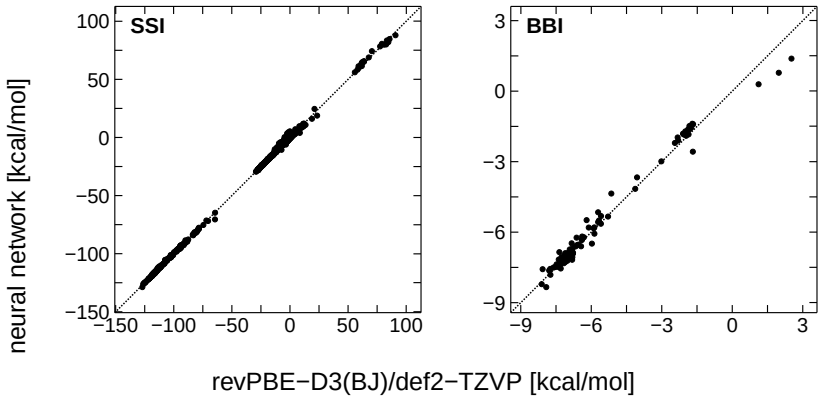


FIGURE 4.27: Correlation of interaction energies for structures in the BioFragment Database[231] (left: SSI, right: BBI) predicted with the ensemble of NNs with values obtained from *ab initio* calculations at the revPBE-D3(BJ)/def2-TZVP level of theory. For both cases, the NN predictions correlate well with the reference data (SSI: $R^2 = 0.9997$, BBI: $R^2 = 0.9922$).

In order to test whether predictions also generalize to larger molecules, deca-alanine (Ala_{10}), which is a widely used model system to study protein folding dynamics,[236] is considered as a test case. Starting from a previously published helical structure of Ala_{10} (capped with an acetylated N-terminus and amidated C-terminus),[237] its geometry was optimized with the BFGS algorithm[238] using the ensemble of NNs, as well as revPBE-D3(BJ)/def2-TZVP to determine the necessary energy gradients. The energies (relative to free atoms) of the optimized structures are −11339.49 kcal mol^{-1}

and −11317.05 kcal mol^{-1} for the *ab initio* method and the NN, respectively, which corresponds to a relative error of about 0.20%. Although the ensemble of NNs predicts the optimized structure to be about 0.207 kcal mol^{-1} atom^{-1} less stable than the *ab initio* method, both optimized geometries are structurally almost indistinguishable (RMSD = 0.21 Å, see Figure 4.28). This result is remarkable, considering that the "protein fragments" used for training the NNs contain at most eight heavy atoms (see section 4.3.1.3), whereas Ala_{10} consists of 54 heavy atoms (109 atoms in total).

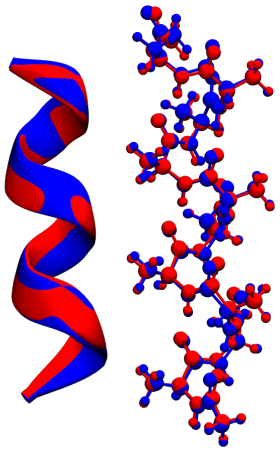


FIGURE 4.28: Optimized structures of helical Ala_{10} in Cartoon representation (left) and as ball-and-stick model (right). The structures obtained using an ensemble of NNs (red) and the reference *ab initio* method (blue) are superimposed to highlight differences.

As a final test, the folding process of Ala_{10} was investigating by running unbiased Langevin dynamics[239] with the ASE[223] at a temperature of 300 K and using a time step of 0.1 fs. The necessary forces were obtained from the predictions of the ensemble of NNs. Starting from the optimized structure of stretched Ala_{10}, the simulation was run for a total of 400 000 time steps (40 ps). After about 30 ps of simulation, Ala_{10} folds into a wreath-shaped structure (see Figure 4.29), in which it remains for the remainder of the simulation.

To determine whether the PES explored during the dynamics is representative of the PES computed using the reference method, the energy of 20 structures sampled at 2 ps intervals along the trajectory was evaluated with the ensemble of NNs and revPBE-D3(BJ)/def2-TZVP. On average, the prediction error for these structures is 0.233 kcal mol^{-1} atom^{-1} (0.23% relative error), with minimum and maximum errors of 0.072 kcal mol^{-1} atom^{-1} (0.07% relative error) and 0.405 kcal mol^{-1} atom^{-1} (0.39% relative error), respectively. Finally, to determine whether the wreath-shaped conformation obtained at the end of the trajectory is a local minimum on the Ala_{10} PES, its geometry was optimized with BFGS using the ensemble of NNs, as well as the reference *ab initio* method to determine the necessary energy gradients. The energies (relative to free atoms) of the optimized structures are

−11339.95 kcal mol^{-1} and −11337.07 kcal mol^{-1} for the *ab initio* method and the NNs, respectively (which corresponds to a relative error of about 0.03%). Both optimized geometries are very similar (RMSD = 0.52 Å, see Figure 4.29), however, the NNs predict the wreath-shaped geometry to be more stable than the helical form by about 0.184 kcal mol^{-1} atom^{-1}, whereas according to the *ab initio* method, both structures have almost the same energy (the wreath-shaped geometry is still more stable, but only by about 0.004 kcal mol^{-1} atom^{-1}). The RMSD of Ala_{10} with respect to the optimized wreath-shaped structure along the trajectory is shown in Figure 4.30.

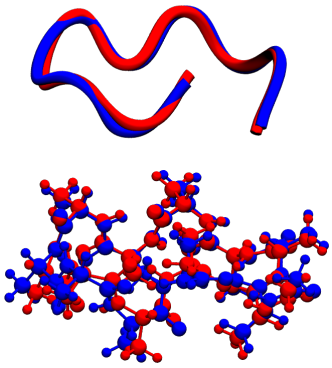


FIGURE 4.29: Optimized structures of wreath-shaped Ala_{10} in Cartoon representation (top) and as ball-and-stick model (bottom). The structures obtained using an ensemble of NNs (red) and the reference *ab initio* method (blue) are superimposed to highlight differences.

4.3.3 Discussion and conclusion

While optimized helical structures of Ala_{10} using the NN-PES (trained on the solvated protein fragments dataset) and the *ab initio* method are almost identical (see Figure 4.28), the relative error in the energy prediction of the ensemble of NNs is about an order of magnitude larger than for the wreath-shaped structure of Ala_{10} (see Figure 4.29). A possible explanation for this discrepancy could be the large dipole moment of helical protein structures due to the cumulative effect of the individual dipole moments of carbonyl groups aligned along the helix axis.[240] The electric field associated with a large dipole moment likely leads to strong polarization effects, which potentially influence the total energy substantially. While polarization effects can be captured implicitly by the NN model due to its ability to assign environment-dependent partial charges to atoms, it is likely that the structures included in the training data do not contain sufficient information to describe the cumulative polarization effects of multiple aligned dipole moments. A bigger dataset of reference structures including helical motifs would likely be needed for a proper description of such phenomena.

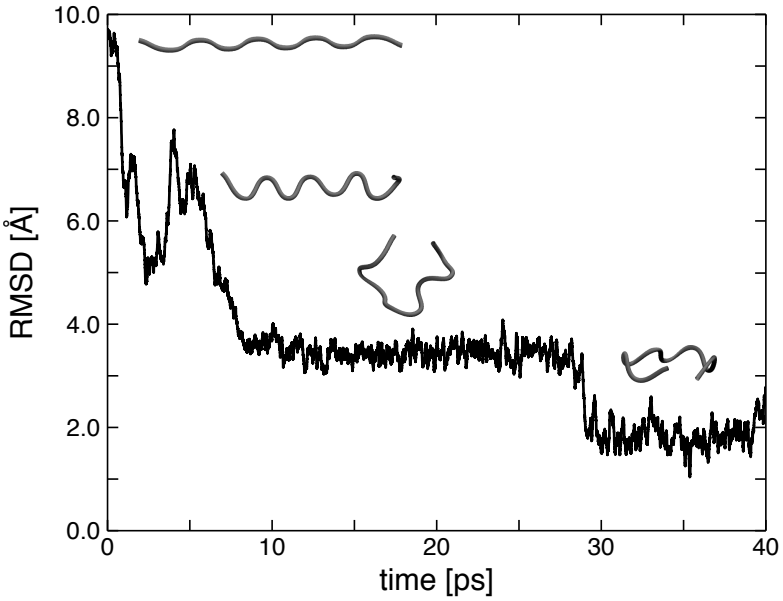


FIGURE 4.30: RMSD of Ala_{10} with respect to the optimized wreath-shaped geometry (see Figure 4.29) over the course of a 40 ps MD trajectory. Cartoon representations of the structure for representative snapshots are shown as well.

In summary, the message-passing HDNN is able to accurately predict energies and forces for a wide range of structures across chemical and conformational degrees of freedom and different datasets. For S_N2 reactions of methyl halides with halide anions, it was shown that including long-range electrostatic interactions explicitly in the model significantly improves the qualitative shape of the predicted PES close to and beyond the cut-off radius. Further, it was shown that the NN can learn to distinguish between intra- and intermolecular contributions in SSIs and BBIs of proteins in a meaningful manner. When the proposed NN is trained on a large set of small reference structures, it was shown that the resulting model is able to generalize to larger structures like Ala_{10} with similar structural motifs. This result suggests that with a systematically constructed set of small reference structures, it is possible to build a transferable NN-PES applicable to a wide range of chemical systems. However, some large-scale effects, for example strong electric fields due to multiple aligned microscopic dipole moments, might not be properly accounted for when training only on small molecules.

Chapter 5

Conclusion and future directions

In this thesis, three different approaches to construct potential energy surfaces (PESs) are explored and ways to improve these methods are discussed. The minimal distributed charge model (MDCM) is introduced as a way to improve the description of electrostatic interactions in empirical force fields (FFs) without reducing their computational efficiency, which is one of their greatest advantages. Kernel ridge regression (KRR) is described as a method to directly interpolate pointwise solutions to the Schrödinger equation (SE) in order to construct PESs for small systems. Algorithms to speed up this approach are implemented in the reproducing kernel Hilbert space (RKHS) toolkit, which largely automates the construction of PESs for small systems rivalling the computational efficiency of FFs. Finally, PESs based on artificial neural networks (NNs) are shown to be an accurate alternative to FFs for larger systems and are able to describe chemical reactions. Provided they are trained on appropriate reference data, NN-based PESs are transferable between similar classes of compounds, for example proteins.

For small and medium-sized systems, the construction of efficient and accurate PESs with KRR is a well-established and matured method. For larger systems however, it is worthwhile to explore further possibilities for improvement. Future research could for example focus on constructing an NN-based PESs applicable to all chemical systems without retraining, which would largely replace the need for performing *ab initio* calculations in many applications. For this purpose, an appropriate set of reference data, which systematically covers general rules of chemistry, would be required for training the NN. However, while NNs are arguably one of the most promising approaches to construct an "ideal" PES, they are unlikely to challenge the computational efficiency of empirical FFs. For this reason, possible ways to improve the accuracy of FFs without sacrificing their efficiency will remain important. It might even be possible to combine the advantages of FFs and NNs: Similar to the well-established quantum mechanics/molecular mechanics (QM/MM) approach[241] the strengths of NNs (accuracy and reactivity) and FFs (speed) could be combined to study chemical processes in condensed phase systems.

Bibliography

1. Dirac, P. A. M. Quantum mechanics of many-electron systems. *Proc. R. Soc. Lond. A* **123,** 714–733 (1929).
2. Schrödinger, E. An undulatory theory of the mechanics of atoms and molecules. *Phys. Rev.* **28,** 1049–1070 (1926).
3. Born, M. & Oppenheimer, R. Zur Quantentheorie der Molekeln. *Ann. Phys.* **389,** 457–484 (1927).
4. Bunker, P. & Moss, R. The breakdown of the Born-Oppenheimer approximation: The effective vibration-rotation Hamiltonian for a diatomic molecule. *Mol. Phys.* **33,** 417–424 (1977).
5. Coxon, J. A. & Hajigeorgiou, P. G. Born-Oppenheimer breakdown in the ground state of carbon monoxide: A direct reduction of spectroscopic line positions to analytical radial Hamiltonian operators. *Can. J. Phys.* **70,** 40–54 (1992).
6. Pisana, S. *et al.* Breakdown of the adiabatic Born-Oppenheimer approximation in graphene. *Nat. Mater.* **6,** 198 (2007).
7. Ziman, J. M. *Electrons and Phonons: The Theory of Transport Phenomena in Solids* (Oxford University Press, 1960).
8. Hospital, A., Goñi, J. R., Orozco, M. & Gelpí, J. L. Molecular dynamics simulations: Advances and applications. *Adv. Appl. Bioinform. Chem.* **8,** 37 (2015).
9. Newton, I. *Philosophiae Naturalis Principia Mathematica* (G. Brookman, 1687).
10. Karplus, M. & McCammon, J. A. Molecular dynamics simulations of biomolecules. *Nat. Struct. Mol. Biol.* **9,** 646–652 (2002).
11. Boltzmann, L. *Vorlesungen über Gastheorie* (Johann Ambrosius Barth Verlag, 1896).
12. González, M. *Force fields and molecular dynamics simulations* in *École thématique de la Société Française de la Neutronique* **12** (EDP Sciences, 2011), 169–200.
13. Brooks, B. R. *et al.* CHARMM: A Program for macromolecular energy, minimization, and dynamics calculations. *J. Chem. Comp.* **4,** 187–217 (1983).

14. Rappé, A. K., Casewit, C. J., Colwell, K., Goddard III, W. & Skiff, W. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *J. Am. Chem. Soc.* **114,** 10024–10035 (1992).

15. Pearlman, D. A. *et al.* AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Comput. Phys. Commun.* **91,** 1–41 (1995).

16. Cornell, W. D. *et al.* A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.* **117,** 5179–5197 (1995).

17. Halgren, T. A. Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *J. Comput. Chem.* **17,** 490–519 (1996).

18. Jorgensen, W. L., Maxwell, D. S. & Tirado-Rives, J. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.* **118,** 11225–11236 (1996).

19. Lennard-Jones, J. E. On the determination of molecular fields. – II. From the equation of state of a gas. *Proc. R. Soc. Lond. A* **106,** 463–477 (1924).

20. Patel, S. & Brooks, C. L. CHARMM fluctuating charge force field for proteins: I. Parameterization and application to bulk organic liquid simulations. *J. Comput. Chem.* **25,** 1–16 (2004).

21. Patel, S., Mackerell, A. D. & Brooks, C. L. CHARMM fluctuating charge force field for proteins: II. Protein/solvent properties from molecular dynamics simulations using a nonadditive electrostatic model. *J. Comput. Chem.* **25,** 1504–1514 (2004).

22. Xie, W., Pu, J., MacKerell, A. D. & Gao, J. Development of a polarizable intermolecular potential function (PIPF) for liquid amides and alkanes. *J. Chem. Theory Comput.* **3,** 1878–1889 (2007).

23. Ponder, J. W. *et al.* Current status of the AMOEBA polarizable force field. *J. Phys. Chem. B* **114,** 2549–2564 (2010).

24. Tersoff, J. New empirical model for the structural properties of silicon. *Phys. Rev. Lett.* **56,** 632 (1986).

25. Brenner, D. W. & Garrison, B. J. Dissociative valence force field potential for silicon. *Phys. Rev. B* **34,** 1304 (1986).

26. Tersoff, J. New empirical approach for the structure and energy of covalent systems. *Phys. Rev. B* **37,** 6991 (1988).

27. Van Duin, A. C., Dasgupta, S., Lorant, F. & Goddard, W. A. ReaxFF: A reactive force field for hydrocarbons. *J. Phys. Chem. A* **105,** 9396–9409 (2001).

28. Franke, R. & Nielson, G. Smooth interpolation of large sets of scattered data. *Int. J. Numer. Meth. Eng.* **15,** 1691–1704 (1980).

29. Nguyen, K. A., Rossi, I. & Truhlar, D. G. A dual-level shepard interpolation method for generating potential energy surfaces for dynamics calculations. *J. Chem. Phys.* **103,** 5522–5530 (1995).

30. Bettens, R. P. & Collins, M. A. Learning to interpolate molecular potential energy surfaces with confidence: A Bayesian approach. *J. Chem. Phys.* **111,** 816–826 (1999).

31. Lancaster, P. & Salkauskas, K. Surfaces generated by moving least squares methods. *Math. Comput.* **37,** 141–158 (1981).

32. Ischtwan, J. & Collins, M. A. Molecular potential energy surfaces by interpolation. *J. Chem. Phys.* **100,** 8080–8088 (1994).

33. Dawes, R., Thompson, D. L., Wagner, A. F. & Minkoff, M. Interpolating moving least-squares methods for fitting potential energy surfaces: A strategy for efficient automatic data point placement in high dimensions. *J. Chem. Phys.* **128,** 084107 (2008).

34. Cassam-Chenaï, P. & Patras, F. Symmetry-adapted polynomial basis for global potential energy surfaces – Applications to XY_4 molecules. *J. Math. Chem* **44,** 938–966 (2008).

35. Braams, B. J. & Bowman, J. M. Permutationally invariant potential energy surfaces in high dimensionality. *Int. Rev. Phys. Chem.* **28,** 577–606 (2009).

36. Paukku, Y., Yang, K. R., Varga, Z. & Truhlar, D. G. Global *ab initio* ground-state potential energy surface of N_4. *J. Chem. Phys.* **139,** 044309 (2013).

37. Ho, T.-S. & Rabitz, H. A general method for constructing multidimensional molecular potential energy surfaces from *ab initio* calculations. *J. Chem. Phys.* **104,** 2584–2597 (1996).

38. Hollebeek, T., Ho, T.-S. & Rabitz, H. A fast algorithm for evaluating multidimensional potential energy surfaces. *J. Chem. Phys.* **106,** 7223–7227 (1997).

39. Hollebeek, T., Ho, T.-S. & Rabitz, H. Constructing multidimensional molecular potential energy surfaces from *ab initio* data. *Annu. Rev. Phys. Chem.* **50,** 537–570 (1999).

40. Behler, J. Neural network potential-energy surfaces in chemistry: A tool for large-scale simulations. *Phys. Chem. Chem. Phys.* **13,** 17930–17955 (2011).

41. Rupp, M. Machine learning for quantum mechanics in a nutshell. *Int. J. Quantum Chem.* **115,** 1058–1073 (2015).

42. Nagy, T., Yosa Reyes, J. & Meuwly, M. Multisurface adiabatic reactive molecular dynamics. *J. Chem. Theory Comput.* **10,** 1366–1375 (2014).

43. Lammers, S., Lutz, S. & Meuwly, M. Reactive force fields for proton transfer dynamics. *J. Comput. Chem.* **29,** 1048–1063 (2008).

44. Unke, O. T., Devereux, M. & Meuwly, M. Minimal distributed charges: Multipolar quality at the cost of point charge electrostatics. *J. Chem. Phys.* **147,** 161712 (2017).

45. Morse, P. M. Diatomic molecules according to the wave mechanics. II. Vibrational levels. *Phys. Rev.* **34,** 57 (1929).

46. Urey, H. C. & Bradley Jr, C. A. The vibrations of pentatonic tetrahedral molecules. *Phys. Rev.* **38,** 1969 (1931).

47. Pauli, W. Über den Zusammenhang des Abschlusses der Elektronengruppen im Atom mit der Komplexstruktur der Spektren. *Z. Phys.* **31,** 765–783 (1925).

48. Halgren, T. A. The representation of van der Waals (vdW) interactions in molecular mechanics force fields: Potential form, combination rules, and vdW parameters. *J. Am. Chem. Soc.* **114,** 7827–7843 (1992).

49. Lorentz, H. A. Über die Anwendung des Satzes vom Virial in der kinetischen Theorie der Gase. *Ann. Phys.* **248,** 127–136 (1881).

50. Berthelot, D. Sur le mélange des gaz. *Compt. Rendus* **126,** 1703–1706 (1898).

51. Buckingham, R. A. The classical equation of state of gaseous helium, neon and argon. *Proc. R. Soc. Lond. A* **168,** 264–283 (1938).

52. Stone, A. Distributed multipole analysis, or how to describe a molecular charge distribution. *Chem. Phys. Lett.* **83,** 233–239 (1981).

53. Stone, A. *The Theory of Intermolecular Forces* (Oxford University Press, 2013).

54. Kramer, C., Gedeck, P. & Meuwly, M. Atomic multipoles: Electrostatic potential fit, local reference axis systems, and conformational dependence. *J. Chem. Comp.* **33,** 1673–1688 (2012).

55. Liem, S. & Popelier, P. High-rank quantum topological electrostatic potential: Molecular dynamics simulation of liquid hydrogen fluoride. *J. Chem. Phys.* **119,** 4560–4566 (2003).

56. Devereux, M., Raghunathan, S., Fedorov, D. G. & Meuwly, M. A novel, computationally efficient multipolar model employing distributed charges for molecular dynamics simulations. *J. Chem. Theory. Comput.* **10,** 4229–4241 (2014).

57. Gao, Q. *et al.* Octahedral point-charge model and its application to fragment molecular orbital calculations of chemical shifts. *Chem. Phys. Lett.* **593,** 165–173 (2014).

58. Marquardt, D. W. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.* **11,** 431–441 (1963).

59. Rios, L. M. & Sahinidis, N. V. Derivative-free optimization: A review of algorithms and comparison of software implementations. *J. Global Optim.* **56,** 1247–1293 (2013).

60. Storn, R. & Price, K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11,** 341–359 (1997).

61. Das, S. & Suganthan, P. N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **15,** 4–31 (2011).

62. Tschumper, G. S. *et al.* Anchoring the water dimer potential energy surface with explicitly correlated computations and focal point analyses. *J. Chem. Phys.* **116,** 690–701 (2002).

63. Bereau, T., Kramer, C. & Meuwly, M. Leveraging symmetries of static atomic multipole electrostatics in molecular dynamics simulations. *J. Chem. Theory. Comput.* **9,** 5450–5459 (2013).

64. El Hage, K., Bereau, T., Jakobsen, S. & Meuwly, M. Impact of quadrupolar electrostatics on atoms adjacent to the sigma-hole in condensed-phase simulations. *J. Chem. Theory. Comput.* **12,** 3008–3019 (2016).

65. Mezura-Montes, E., Velázquez-Reyes, J. & Coello, C. A. *Modified differential evolution for constrained optimization* in *2006 IEEE International Conference on Evolutionary Computation* (2006), 25–32.

66. Bondi, A. Van der Waals volumes and radii. *J. Phys. Chem.* **68,** 441–451 (1964).

67. Popelier, P. L. A., Aicken, F. M. & O'Brien, S. E. in *Chemical modelling: Applications and theory* 143–198 (Royal Society of Chemistry Specialist Periodical Report, 2000).

68. Lillestolen, T. C. & Wheatley, R. J. Redefining the atom: Atomic charge densities produced by an iterative stockholder approach. *Chem. Commun.* 5909–5911 (2008).

69. Misquitta, A. J., Stone, A. J. & Fazeli, F. Distributed multipoles from a robust basis-space implementation of the iterated stockholder atoms procedure. *J. Chem. Theory. Comput.* **10,** 5405–5418 (2014).

70. Frisch, M. J. *et al. Gaussian-09 Revision A.02* Gaussian Inc. Wallingford CT 2009.

71. Shi, Y. *et al.* Polarizable atomic multipole-based AMOEBA force field for proteins. *J. Chem. Theory. Comput.* **9,** 4046–4063 (2013).

72. Stone, A. J. Distributed multipole analysis: Stability for large basis sets. *J. Chem. Theory Comput.* **1,** 1128–1132 (2005).

73. Lee, B. & Richards, F. M. The interpretation of protein structures: Estimation of static accessibility. *J. Mol. Biol.* **55,** 379–400 (1971).

74. Devereux, M., Gresh, N., Piquemal, J.-P. & Meuwly, M. A supervised fitting approach to force field parametrization with application to the SIBFA polarizable force field. *J. Comput. Chem.* **35,** 1577–1591 (2014).

75. Gresh, N., Cisneros, G. A., Darden, T. A. & Piquemal, J.-P. Anisotropic, polarizable molecular mechanics studies of inter- and intramolecular interactions and ligand-macromolecule complexes. A bottom-up strategy. *J. Chem. Theory. Comput.* **3,** 1960–1980 (2007).

76. Auffinger, P., Hays, F. A., Westhof, E. & Ho, P. S. Halogen bonds in biological molecules. *Proc. Natl. Acad. Sci. U.S.A.* **101,** 16789–16794 (2004).

77. Politzer, P., Murray, J. S. & Concha, M. C. σ-hole bonding between like atoms; a fallacy of atomic charges. *J. Mol. Model.* **14,** 659–665 (2008).

78. Jorgensen, W. L. & Schyman, P. Treatment of halogen bonding in the OPLS-AA force field: Application to potent anti-HIV agents. *J. Chem. Theory. Comput.* **8,** 3895–3901 (2012).

79. Hédin, F., El Hage, K. & Meuwly, M. A toolkit to fit nonbonded parameters from and for condensed phase simulations. *J. Chem. Inf. Model.* **56,** 1479–1489 (2016).

80. Brooks, B. R. *et al.* CHARMM: The biomolecular simulation program. *J. Comput. Chem.* **30,** 1545–1614 (2009).

81. Ryckaert, J.-P., Ciccotti, G. & Berendsen, H. J. Numerical integration of the cartesian equations of motion of a system with constraints: Molecular dynamics of n-alkanes. *J. Comput. Phys.* **23,** 327–341 (1977).

82. Jorgensen, W. L. Quantum and statistical mechanical studies of liquids. 10. Transferable intermolecular potential functions for water, alcohols, and ethers. Application to liquid water. *J. Am. Chem. Soc.* **103,** 335–340 (1981).

83. Benesty, J., Chen, J., Huang, Y. & Cohen, I. in *Noise Reduction in Speech Processing* 1–4 (Springer, 2009).

84. Stone, A. J. Electrostatic damping functions and the penetration energy. *J. Phy. Chem. A* **115,** 7017–7027 (2011).

85. Wang, Q. *et al.* General model for treating short-range electrostatic penetration in a molecular mechanics force field. *J. Chem. Theory. Comput.* **11,** 2609–2618 (2015).

86. Tröster, P., Lorenzen, K. & Tavan, P. Polarizable six-point water models from computational and empirical optimization. *J. Phys. Chem. B* **118,** 1589–1602 (2014).

87. Kusalik, P. G. & Svishchev, I. M. The spatial structure in liquid water. *Science* **265,** 1219–1221 (1994).

88. Mahoney, M. W. & Jorgensen, W. L. A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. *J. Chem. Phys.* **112,** 8910–8922 (2000).

89. Mobley, D. L. *Experimental and calculated small molecule hydration free energies* 2013.

90. Mobley, D. L. & Guthrie, J. P. FreeSolv: A database of experimental and calculated hydration free energies, with input files. *J. Comput. Aided Mol. Des.* **28,** 711–720 (2014).

91. Unke, O. T. & Meuwly, M. Toolkit for the construction of reproducing kernel-based representations of data: Application to multidimensional potential energy surfaces. *J. Chem. Inf. and Mod.* **57,** 1923–1931 (2017).

92. Freedman, D. A. *Statistical Models: Theory and Practice* (Cambridge University Press, 2009).

93. Schölkopf, B., Smola, A. & Müller, K.-R. *Kernel principal component analysis* in *International conference on artificial neural networks* (1997), 583–588.

94. Boser, B. E., Guyon, I. M. & Vapnik, V. N. *A training algorithm for optimal margin classifiers* in *Proceedings of the fifth annual workshop on Computational learning theory* (1992), 144–152.

95. Schölkopf, B., Smola, A. & Müller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.* **10,** 1299–1319 (1998).

96. Theodoridis, S., Koutroumbas, K., *et al. Pattern Recognition* (Elsevier, 2008).

97. Smola, A. J. & Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **14,** 199–222 (2004).

98. Weinert, H. L. *Reproducing kernel Hilbert spaces: Applications in statistical signal processing* (Hutchinson, 1982).

99. Aronszajn, N. Theory of reproducing kernels. *Trans. Amer. Math. Soc.* **68,** 337–404 (1950).

100. Schölkopf, B., Herbrich, R. & Smola, A. J. *A generalized representer theorem* in *International Conference on Computational Learning Theory* (Springer Berlin Heidelberg, 2001), 416–426.

101. Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K. & Schölkopf, B. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* **12** (2001).

102. Hofmann, T., Schölkopf, B. & Smola, A. J. Kernel methods in machine learning. *Ann. Stat.* 1171–1220 (2008).

103. Golub, G. H. & van Loan, C. F. *Matrix Computations* (JHU Press, 2012).

104. Tikhonov, A. N., Arsenin, V. I. & John, F. *Solutions of Ill-Posed Problems* (Winston Press, 1977).

105. Fernandes, P. & Plateau, B. Triangular solution of linear systems in tensor product format. *ACM SIGMETRICS Perform. Eval. Rev.* **28,** 30–32 (2001).

106. Sedgewick, R. *Algorithms* (Pearson Education, 1988).

107. Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. *Introduction to Algorithms* (MIT Press, 2001).

108. Johnson, B. On hyperspherical coordinates and mapping the internal configurations of a three body system. *J. Chem. Phys.* **73,** 5051–5058 (1980).

109. Pack, R. T. Coordinates for an optimum CS approximation in reactive scattering. *Chem. Phys. Lett.* **108,** 333–338 (1984).

110. Pack, R. T. & Parker, G. A. Quantum reactive scattering in three dimensions using hyperspherical (APH) coordinates. Theory. *J. Chem. Phys.* **87,** 3888–3921 (1987).

111. Wahba, G. *Spline models for observational data* (SIAM, 1990).

112. Smola, A. J., Schölkopf, B. & Müller, K.-R. The connection between regularization operators and support vector kernels. *Neural Netw.* **11,** 637–649 (1998).

113. Hazewinkel, M. *Encyclopaedia of Mathematics* (Reidel, 2001).

114. Unke, O. T., Castro-Palacio, J. C., Bemish, R. J. & Meuwly, M. Collision-induced rotational excitation in N_2^+ ($^2\Sigma_g^+$, $v = 0$)–Ar: Comparison of computations and experiment. *J. Chem. Phys.* **144,** 224307 (2016).

115. Denis-Alpizar, O., Unke, O. T., Bemish, R. J. & Meuwly, M. Quantum and quasiclassical trajectory studies of rotational relaxation in Ar–N_2^+ collisions. *Phys. Chem. Chem. Phys.* **19,** 27945–27951 (2017).

116. Chmiela, S. *et al.* Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.* **3,** e1603015 (2017).

117. Chmiela, S., Sauceda, H. E., Poltavsky, I., Müller, K.-R. & Tkatchenko, A. sGDML: Constructing Accurate and Data Efficient Molecular Force Fields Using Machine Learning. *arXiv preprint arXiv:1812.04986* (2018).

118. Chmiela, S., Sauceda, H. E., Müller, K.-R. & Tkatchenko, A. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* **9,** 3887 (2018).

119. Unke, O. T. & Meuwly, M. A reactive, scalable, and transferable model for molecular energies from a neural network approach based on local information. *J. Chem. Phys.* **148,** 241708 (2018).

120. Unke, O. T. & Meuwly, M. PhysNet: A neural network for predicting energies, forces, dipole moments and partial charges. *J. Chem. Theory Comput.* Article ASAP (2019).

121. Alberts, B *et al. Molecular Biology of the Cell* (Garland Science, 1994).

122. Levitan, I. B. & Kaczmarek, L. K. *The Neuron: Cell and Molecular Biology* (Oxford University Press, 2001).

123. Von Bartheld, C. S., Bahney, J. & Herculano-Houzel, S. The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting. *J. Comp. Neurol.* **524,** 3865–3895 (2016).

124. Drachman, D. A. Do we have brain to spare? *Neurology* **64,** 2004–2005 (2005).

125. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Sci.* **5,** 115–133 (1943).

126. Hebb, D. O. *The Organization of Behavior: A Neuropsychological Theory* (John Wiley & Sons, 1949).

127. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65,** 386 (1958).

128. Olazaran, M. A sociological study of the official history of the perceptrons controversy. *Soc. Stud. Sci.* **26,** 611–659 (1996).

129. Minsky, M. & Papert, S. A. *Perceptrons: An Introduction to Computational Geometry* (MIT Press, 1969).

130. Werbos, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* PhD thesis (Harvard University, 1974).

131. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. *Learning internal representations by error propagation* tech. rep. (University of San Diego, 1985).

132. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323,** 533–536 (1986).

133. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **2,** 359–366 (1989).

134. Cybenko, G. Approximation by superposition of sigmoidal functions. *Math. Control Signals Syst.* **2,** 303–314 (1989).

135. Chen, T. & Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Netw. Learn. Syst.* **6,** 911–917 (1995).

136. Eldan, R. & Shamir, O. *The power of depth for feedforward neural networks* in *Conference on Learning Theory* (2016), 907–940.

137. LeCun, Y. & Bengio, Y. in *The Handbook of Brain Theory and Neural Networks* 276–279 (MIT Press, 1995).

138. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9,** 1735–1780 (1997).

139. Glorot, X. & Bengio, Y. *Understanding the difficulty of training deep feedforward neural networks* in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics* (2010), 249–256.

140. Nair, V. & Hinton, G. E. *Rectified linear units improve restricted boltzmann machines* in *Proceedings of the 27th International Conference on Machine Learning* (2010), 807–814.

141. Glorot, X., Bordes, A. & Bengio, Y. *Deep sparse rectifier neural networks* in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* (2011), 315–323.

142. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *Imagenet classification with deep convolutional neural networks* in *Advances in Neural Information Processing Systems* (2012), 1097–1105.

143. Hinton, G. *et al.* Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **29,** 82–97 (2012).

144. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

145. Lawrence, S., Giles, C. L., Tsoi, A. C. & Back, A. D. Face recognition: A convolutional neural-network approach. *IEEE Trans. Neural Netw.* **8,** 98–113 (1997).

146. Blank, T. B., Brown, S. D., Calhoun, A. W. & Doren, D. J. Neural network models of potential energy surfaces. *J. Chem. Phys.* **103,** 4129–4137 (1995).

147. Brown, D. F., Gibbs, M. N. & Clary, D. C. Combining *ab initio* computations, neural networks, and diffusion Monte Carlo: An efficient method to treat weakly bound molecules. *J. Chem. Phys.* **105,** 7597–7604 (1996).

148. Tafeit, E. *et al.* Neural networks as a tool for compact representation of *ab initio* molecular potential energy surfaces. *J. Mol. Graph.* **14,** 12–18 (1996).

149. No, K. T., Chang, B. H., Kim, S. Y., Jhon, M. S. & Scheraga, H. A. Description of the potential energy surface of the water dimer with an artificial neural network. *Chem. Phys. Lett.* **271,** 152–156 (1997).

150. Prudente, F. V. & Neto, J. S. The fitting of potential energy surfaces using neural networks. Application to the study of the photodissociation processes. *Chem. Phys. Lett.* **287,** 585–589 (1998).

151. Gassner, H., Probst, M., Lauenstein, A. & Hermansson, K. Representation of intermolecular potential functions by neural networks. *J. Phys. Chem. A* **102,** 4596–4605 (1998).

152. Manzhos, S. & Carrington Jr, T. A random-sampling high dimensional model representation neural network for building potential energy surfaces. *J. Chem. Phys.* **125,** 084109 (2006).

153. Manzhos, S. & Carrington Jr, T. Using redundant coordinates to represent potential energy surfaces with lower-dimensional functions. *J. Chem. Phys.* **127,** 014103 (2007).

154. Malshe, M *et al.* Development of generalized potential-energy surfaces using many-body expansions, neural networks, and moiety energy approximations. *J. Chem. Phys.* **130,** 184102 (2009).

155. Behler, J. & Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **98,** 146401 (2007).

156. Behler, J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.* **134,** 074106 (2011).

157. Khorshidi, A. & Peterson, A. A. Amp: A modular approach to machine learning in atomistic simulations. *Comput. Phys. Commun.* **207,** 310–324 (2016).

158. Artrith, N., Urban, A. & Ceder, G. Efficient and accurate machine-learning interpolation of atomic energies in compositions with many species. *Phys. Rev. B* **96,** 014112 (2017).

159. Smith, J. S., Isayev, O. & Roitberg, A. E. ANI-1: An extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* **8,** 3192–3203 (2017).

160. Yao, K., Herr, J. E., Toth, D. W., Mckintyre, R. & Parkhill, J. The TensorMol-0.1 model chemistry: A neural network augmented with long-range physics. *Chem. Sci.* **9,** 2261–2269 (2018).

161. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212* (2017).

162. Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R. & Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.* **8,** 13890 (2017).

163. Schütt, K. T. *et al. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions* in *Advances in Neural Information Processing Systems* (2017).

164. Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A. & Müller, K.-R. SchNet – A deep learning architecture for molecules and materials. *J. Chem. Phys.* **148,** 241722 (2018).

165. Lubbers, N., Smith, J. S. & Barros, K. Hierarchical modeling of molecular energies using a deep neural network. *J. Chem. Phys.* **148,** 241715 (2018).

166. Ramakrishnan, R., Dral, P. O., Rupp, M. & Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Sci. Data* **1** (2014).

167. Blanco, M., Martín Pendás, A & Francisco, E. Interacting quantum atoms: A correlated energy decomposition scheme based on the quantum theory of atoms in molecules. *J. Chem. Theory. Comput.* **1,** 1096–1109 (2005).

168. Francisco, E, Martín Pendás, A & Blanco, M. A molecular energy decomposition scheme for atoms in molecules. *J. Chem. Theory. Comput.* **2,** 90–102 (2006).

169. Mitoraj, M. P., Michalak, A. & Ziegler, T. A combined charge and energy decomposition scheme for bond analysis. *J. Chem. Theory. Comput.* **5,** 962–975 (2009).

170. Behler, J. Representing potential energy surfaces by high-dimensional neural network potentials. *J. Phys. Condens. Matter* **26,** 183001 (2014).

171. Behler, J. Hochdimensionale neuronale Netze für Potentialhyperflächen großer molekularer und kondensierter Systeme. *Angew. Chem.* **129,** 13006–13020 (2017).

172. Harris, D. & Harris, S. *Digital Design and Computer Architecture* (Morgan Kaufmann, 2010).

173. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. *Distributed representations of words and phrases and their compositionality* in *Advances in Neural Information Processing Systems* (2013), 3111–3119.

174. Globerson, A., Chechik, G., Pereira, F. & Tishby, N. Euclidean embedding of co-occurrence data. *J. Mach. Learn. Res.* **8,** 2265–2295 (2007).

175. Bartók, A. P., Kondor, R. & Csányi, G. On representing chemical environments. *Phys. Rev. B* **87,** 184115 (2013).

176. LeCun, Y., Bottou, L., Orr, G. B. & Müller, K.-R. *Neural Networks: Tricks of the Trade* (Springer, 1998).

177. Broomhead, D. S. & Lowe, D. *Radial basis functions, multi-variable functional interpolation and adaptive networks* tech. rep. (Royal Signals and Radar Establishment Malvern (United Kingdom), 1988).

178. Lowe, D. Multi-variable functional interpolation and adaptive networks. *Complex Syst.* **2,** 321–355.

179. Schwenker, F., Kestler, H. A. & Palm, G. Three learning phases for radial-basis-function networks. *Neural Netw.* **14,** 439–458 (2001).

180. Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515* (2017).

181. Ruddigkeit, L., Van Deursen, R., Blum, L. C. & Reymond, J.-L. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *J. Chem. Inf. Model.* **52,** 2864–2875 (2012).

182. Perdew, J. P., Burke, K. & Ernzerhof, M. Generalized gradient approximation made simple. *Phys. Rev. Lett.* **77,** 3865 (1996).

183. Tkatchenko, A. & Scheffler, M. Accurate molecular van der Waals interactions from ground-state electron density and free-atom reference data. *Phys. Rev. Lett.* **102,** 073005 (2009).

184. Yang, Y. & Meuwly, M. A generalized reactive force field for nonlinear hydrogen bonds: Hydrogen dynamics and transfer in malonaldehyde. *J. Chem. Phys.* **133,** 064503 (2010).

185. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

186. Zill, D., Wright, W. S. & Cullen, M. R. *Advanced Engineering Mathematics* (Jones & Bartlett Learning, 2011).

187. Abadi, M. *et al. TensorFlow: Large-scale machine learning on heterogeneous systems* Software available from tensorflow.org. 2015. <`http://tensorflow.org/`>.

188. Sheather, S. J. & Jones, M. C. A reliable data-based bandwidth selection method for kernel density estimation. *J. Royal Stat. Soc.* 683–690 (1991).

189. McNaught, A. D. & McNaught, A. D. *Compendium of Chemical Terminology* (Blackwell Science, 1997).

190. Even, S. *Graph Algorithms* (Cambridge University Press, 2011).

191. Arnold, B. C. *Pareto Distribution* (John Wiley & Sons, 2015).

192. Proctor, W. & Yu, F. The dependence of a nuclear magnetic resonance frequency upon chemical compound. *Phys. Rev.* **77,** 717 (1950).

193. Stock, L. M. The origin of the inductive effect. *J. Chem. Educ* **49,** 400 (1972).

194. Wehril, F., Marchand, A. P. & Wehrli, S. *Interpretation of Carbon-13 NMR Spectra* (John Wiley & Sons, 1988).

195. Lange, N. A. *Lange's Handbook of Chemistry* (McGraw-Hill, 1967).

196. Pronobis, W., Tkatchenko, A. & Müller, K.-R. Many-body descriptors for predicting molecular properties with machine learning: Analysis of pairwise and three-body interactions in molecules. *J. Chem. Theory Comput.* **14,** 2991–3003 (2018).

197. Artrith, N., Morawietz, T. & Behler, J. High-dimensional neural-network potentials for multicomponent systems: Applications to zinc oxide. *Phys. Rev. B* **83,** 153101 (2011).

198. Hirshfeld, F. L. Bonded-atom fragments for describing molecular charge densities. *Theor. Chem. Acc.* **44,** 129–138 (1977).

199. Piquemal, J.-P., Gresh, N. & Giessner-Prettre, C. Improved formulas for the calculation of the electrostatic contribution to the intermolecular interaction energy from multipolar expansion of the electronic distribution. *J. Phys. Chem. A* **107,** 10353–10359 (2003).

200. Gastegger, M., Behler, J. & Marquetand, P. Machine learning molecular dynamics for the simulation of infrared spectra. *arXiv preprint arXiv:1705.05907* (2017).

201. Morawietz, T. & Behler, J. A density-functional theory-based neural network potential for water clusters including van der Waals corrections. *J. Phys. Chem. A* **117,** 7356–7366 (2013).

202. Grimme, S., Antony, J., Ehrlich, S. & Krieg, H. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. *J. Chem. Phys.* **132,** 154104 (2010).

203. Unke, O. T. & Meuwly, M. *S_N2 reactions dataset* Zenodo. `http://doi.org/10.5281/zenodo.2605341`. 2019.

204. Unke, O. T. & Meuwly, M. *Solvated protein fragments dataset* Zenodo. `http://doi.org/10.5281/zenodo.2605372`. 2019.

205. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 770–778.

206. He, K., Zhang, X., Ren, S. & Sun, J. *Identity mappings in deep residual networks* in *European Conference on Computer Vision* (2016), 630–645.

207. Cho, K. *et al.* Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

208. Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K. & Worley, S. *Texturing & Modeling: A Procedural Approach* (Morgan Kaufmann, 2003).

209. Mnih, V., Heess, N., Graves, A., *et al. Recurrent models of visual attention* in *Advances in Neural Information Processing Systems* (2014), 2204–2212.

210. O'Connor, A. Exponential decay of bound state wave functions. *Commun. Math. Phys.* **32,** 319–340 (1973).

211. Baydin, A. G., Pearlmutter, B. A., Radul, A. A. & Siskind, J. M. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* **18,** 1–43 (2018).

212. Darden, T., York, D. & Pedersen, L. Particle mesh Ewald: An $N \cdot \log(N)$ method for Ewald sums in large systems. *J. Chem. Phys.* **98,** 10089–10092 (1993).

213. Steinbach, P. J. & Brooks, B. R. New spherical-cutoff methods for long-range forces in macromolecular simulation. *J. Comput. Chem.* **15,** 667–683 (1994).

214. Thrun, S. *Is learning the n-th thing any easier than learning the first?* in *Advances in Neural Information Processing Systems* (1996), 640–646.

215. Caruana, R. Multitask learning. *Mach. Learn.* **28,** 41–75 (1997).

216. Baxter, J. A model of inductive bias learning. *J. Artif. Intell. Res.* **12,** 149–198 (2000).

217. Nebgen, B. *et al.* Transferable dynamic molecular charge assignment using deep neural networks. *J. Chem. Theory Comput.* **14,** 4687–4698 (2018).

218. Reddi, S. J., Kale, S. & Kumar, S. *On the convergence of ADAM and beyond* in *International Conference on Learning Representations* (2018).

219. Prechelt, L. in *Neural Networks: Tricks of the Trade* 53–67 (Springer, 2012).

220. Kozuch, S., Gruzman, D. & Martin, J. M. DSD-BLYP: A general purpose double hybrid density functional including spin component scaling and dispersion correction. *J. Phys. Chem. C* **114,** 20801–20808 (2010).

221. Grimme, S., Ehrlich, S. & Goerigk, L. Effect of the damping function in dispersion corrected density functional theory. *J. Comput. Chem.* **32,** 1456–1465 (2011).

222. Weigend, F. & Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy. *Phys. Chem. Chem. Phys.* **7,** 3297–3305 (2005).

223. Larsen, A. H. *et al.* The atomic simulation environment – A Python library for working with atoms. *J. Phys. Condens. Matter* **29,** 273002 (2017).

224. Stewart, J. J. Optimization of parameters for semiempirical methods VI: More modifications to the NDDO approximations and re-optimization of parameters. *J. Mol. Model.* **19,** 1–32 (2013).

225. Stewart, J. J. P. *MOPAC2016* Stewart Computational Chemistry, Colorado Springs, CO, USA. 2016. <`http://openmopac.net/`>.

226. Neese, F. The ORCA program system. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2,** 73–78 (2012).

227. Neese, F. Software update: The ORCA program system, version 4.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **8,** e1327 (2018).

228. Goerigk, L. *et al.* A look at the density functional theory zoo with the advanced GMTKN55 database for general main group thermochemistry, kinetics and noncovalent interactions. *Phys. Chem. Chem. Phys.* **19,** 32184–32215 (2017).

229. Huang, B. & von Lilienfeld, O. A. The "DNA" of chemistry: Scalable quantum machine learning with "amons". *arXiv preprint arXiv:1707.04146* (2017).

230. Zhang, Y. & Yang, W. Comment on "Generalized gradient approximation made simple". *Phys. Rev. Lett.* **80,** 890 (1998).

231. Burns, L. A. *et al.* The BioFragment Database (BFDb): An open-data platform for computational chemistry analysis of noncovalent interactions. *J. Chem. Phys.* **147,** 161727 (2017).

232. Behler, J. Constructing high-dimensional neural network potentials: A tutorial review. *Int. J. Quantum Chem.* **115,** 1032–1050 (2015).

233. Faber, F. A. *et al.* Prediction errors of molecular machine learning models lower than hybrid DFT error. *J. Chem. Theory Comput.* **13,** 5255–5264 (2017).

234. Breiman, L. Bagging predictors. *Mach. Learn.* **24,** 123–140 (1996).

235. Schütt, K. T. *et al.* SchNetPack: A Deep Learning Toolbox For Atomistic Systems. *J. Chem. Theory Comput.* **15,** 448–455 (2018).

236. Hazel, A., Chipot, C. & Gumbart, J. C. Thermodynamics of deca-alanine folding in water. *J. Chem. Theory Comput.* **10,** 2836–2844 (2014).

237. Park, S., Khalili-Araghi, F., Tajkhorshid, E. & Schulten, K. Free energy calculation from steered molecular dynamics simulations using Jarzynski's equality. *J. Chem. Phys.* **119,** 3559–3566 (2003).

238. Fletcher, R. *Practical Methods of Optimization* (John Wiley & Sons, 2013).

239. Langevin, P. Sur la théorie du mouvement brownien. *C. R. Acad. Sci. Paris.* **146,** 530–533 (1908).

240. Hol, W. G. J., Van Duijnen, P. T. & Berendsen, H. J. C. The α-helix dipole and the properties of proteins. *Nature* **273,** 443–446 (1978).

241. Warshel, A. & Levitt, M. Theoretical studies of enzymic reactions: dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. *J. Mol. Biol.* **103,** 227–249 (1976).

242. Mielke, S. L., Garrett, B. C. & Peterson, K. A. A hierarchical family of global analytic Born-Oppenheimer potential energy surfaces for the H + H_2 reaction ranging in quality from double-zeta to the complete basis set limit. *J. Chem. Phys.* **116,** 4142–4161 (2002).

Appendix A

Parameters for MDCMs

In the following, the exact charge positions x, y, z (in Bohrs a_0) and magnitudes q (in units of elementary charges e) for all minimal distributed charge models (MDCMs) discussed in chapter 2 are given. As a reference, the Cartesian coordinates of all atoms of the respective molecules are also given.

Water

coordinates

atom	x	y	z
O	0.000	0.000	0.222
H	0.000	1.431	−0.890
H	0.000	−1.431	−0.890

MDCM3

x	y	z	q
0.000	0.000	−0.433	−2.516
0.000	0.974	−0.744	1.258
0.000	−0.974	−0.744	1.258

MDCM4

x	y	z	q
0.619	0.000	0.019	−0.470
−0.619	0.000	0.019	−0.470
0.000	1.485	−0.811	0.470
0.000	−1.485	−0.811	0.470

MDCM6

x	y	z	q
0.000	0.000	0.858	−1.403
−0.384	0.000	0.309	−2.122
0.384	0.000	0.309	−2.123
0.000	1.673	−1.002	0.324
0.000	0.000	0.477	5.000
0.000	−1.673	−1.002	0.324

MDCM9

x	y	z	q
0.162	0.111	0.262	−3.481
−0.161	0.112	0.262	−3.526
0.000	−0.489	0.364	1.116
−0.001	0.274	0.282	5.000
0.000	−0.385	−0.654	−2.651
0.000	0.399	−0.648	−2.659
0.000	1.451	−0.947	0.596
0.000	0.002	−0.618	5.000
0.000	−1.445	−0.941	0.605

Imidazole

coordinates

atom	*x*	*y*	*z*
N	0.000	2.090	0.000
N	−1.404	−1.863	0.000
C	1.207	−1.866	0.000
C	−2.071	0.537	0.000
C	2.121	0.566	0.000
H	−3.982	1.260	0.000
H	−0.010	3.995	0.000
H	4.008	1.339	0.000
H	2.270	−3.610	0.000

MDCM6

x	*y*	*z*	*q*
2.240	−3.962	−0.012	0.114
−2.051	−2.594	0.002	−0.299
2.976	0.942	−0.025	−1.480
−4.444	0.997	0.000	0.117
0.054	4.555	−0.004	0.189
3.301	1.072	−0.024	1.359

MDCM9

x	*y*	*z*	*q*
2.956	−3.682	−0.011	0.091
−0.760	−2.597	0.005	3.978
−0.915	−2.427	0.006	−5.000
−1.357	−0.261	0.005	5.000
−1.666	0.130	0.004	−4.432
1.677	0.111	−0.012	−0.681
−3.274	0.995	0.000	0.547
−0.067	4.271	−0.003	0.241
3.781	1.075	−0.012	0.256

MDCM13

x	*y*	*z*	*q*
2.157	−3.363	0.000	0.313
0.774	−1.480	0.010	4.724
1.159	−1.516	0.010	−5.000
−1.833	−1.778	0.013	−1.106
−2.947	−0.077	0.032	4.325
0.055	3.672	0.000	−5.000
2.833	−0.233	0.000	5.000
−3.091	0.214	0.034	−5.000
2.884	−0.072	−0.002	−4.830
−3.564	0.819	0.032	1.184
0.291	3.688	0.002	2.524
−0.165	3.784	−0.001	2.593
4.089	1.277	−0.020	0.273

MDCM16

x	*y*	*z*	*q*
2.284	−3.642	−0.394	−0.309
2.370	−3.698	−0.237	0.459
0.769	−2.350	−0.009	4.778
0.912	−2.314	−0.003	−4.759
−1.632	−2.263	−0.006	−0.690
−2.884	−0.002	0.019	0.452
−0.135	1.659	0.020	−4.089
2.716	−0.324	0.001	0.666
1.828	0.667	0.001	−3.449
1.198	1.110	0.009	4.294
−1.223	1.190	0.000	5.000
−1.798	0.961	−0.366	−1.773
−1.963	0.933	0.493	−1.232
−4.427	1.584	0.073	0.096
−0.033	4.007	0.009	0.410
4.372	1.550	−0.022	0.147

Protonated imidazole

coordinates

atom	x	y	z
N	0.000	−2.132	0.000
N	1.252	1.727	0.000
C	−1.362	1.796	0.000
C	2.060	−0.668	0.000
C	−2.158	−0.656	0.000
H	−2.407	3.545	0.000
H	3.999	−1.298	0.000
H	0.036	−4.044	0.000
H	−4.031	−1.458	0.000
H	2.404	3.254	0.000

MDCM5

x	y	z	q
−2.795	4.141	0.001	0.139
4.328	−1.398	−0.003	0.195
−0.007	−4.492	0.002	0.263
−4.696	−1.697	−0.002	0.140
2.633	3.635	0.003	0.263

MDCM10

x	y	z	q
−2.608	3.758	−0.119	0.201
−1.846	2.486	−0.661	−0.078
2.422	3.331	0.750	−0.901
0.038	−4.168	−0.717	−1.864
−4.090	−0.707	0.058	0.190
−3.195	−0.814	0.216	−0.206
4.426	−1.416	0.001	0.173
0.035	−4.168	−0.615	2.166
−4.197	−2.019	0.104	0.123
2.438	3.351	0.556	1.195

MDCM13

x	y	z	q
−2.555	3.493	0.019	0.257
−0.737	0.927	−0.004	4.973
−1.082	0.932	−0.005	−4.891
2.208	0.393	0.006	1.560
1.684	1.106	0.009	−1.801
2.207	−1.719	−0.003	0.838
2.742	−1.102	−0.026	−1.056
0.594	−2.480	0.026	−0.635
−2.713	0.260	−0.015	0.421
4.317	−1.423	−0.047	0.249
0.036	−3.944	0.014	0.515
−4.568	−1.870	0.002	0.116
2.429	3.247	−0.000	0.452

MDCM18

x	y	z	q
−2.664	3.834	−0.006	0.185
−3.015	3.981	−0.061	−0.002
−1.143	1.232	−0.136	−1.778
−1.596	0.761	−0.144	1.229
0.284	1.696	−0.117	1.582
1.475	1.656	−0.086	−1.998
3.258	−1.149	−0.013	−0.281
0.483	−1.989	−0.017	−3.103
0.877	−2.188	−0.020	1.671
−2.743	0.062	−0.087	0.702
0.369	−1.233	−0.008	1.331
−1.454	−1.461	−0.055	0.746
4.181	−1.393	0.003	0.302
−0.003	−4.193	0.005	0.368
−4.487	−1.611	0.009	0.168
−2.306	−0.360	−0.079	−1.469
2.096	1.245	−0.083	0.917
2.444	3.329	−0.011	0.430

Bromobenzene

coordinates

atom	*x*	*y*	*z*
Br	3.423	0.000	0.000
C	−5.447	0.000	0.000
C	−4.122	−2.280	0.000
C	−1.485	−2.298	0.000
C	−0.198	0.000	0.000
C	−1.485	2.298	0.000
C	−4.122	2.280	0.000
H	−7.498	0.000	0.000
H	−5.135	−4.065	0.000
H	−0.447	−4.064	0.000
H	−0.447	4.064	0.000
H	−5.136	4.065	0.000

MDCM7

x	*y*	*z*	*q*
−4.784	−1.439	−0.025	−0.385
4.579	−0.152	0.023	4.846
4.502	−0.155	0.023	−5.000
−7.282	0.688	−0.010	0.136
−5.606	−3.474	−0.028	0.194
−0.275	−3.995	0.010	0.116
−0.913	4.659	0.011	0.092

MDCM10

x	*y*	*z*	*q*
−4.476	−3.291	0.017	−1.561
4.588	0.028	0.033	4.753
4.476	0.028	0.033	−5.000
0.871	0.034	0.045	0.179
−4.514	3.276	−0.040	−1.048
−8.130	0.204	0.012	0.074
−4.635	−3.500	0.015	1.470
−0.370	−4.713	0.032	0.088
−0.077	4.723	−0.040	0.078
−4.697	3.590	−0.038	0.967

MDCM12

x	*y*	*z*	*q*
−3.532	1.387	−0.017	2.523
−4.107	−2.325	0.029	−0.498
4.325	0.156	0.003	2.337
3.969	0.172	0.001	−2.879
0.168	0.708	−0.060	−2.471
−3.894	1.800	−0.007	−2.682
0.689	0.597	−0.049	2.656
−7.827	0.223	0.023	0.099
−4.935	−3.684	0.034	0.286
−0.374	−4.592	0.025	0.094
−0.232	4.577	−0.073	0.094
−4.819	3.379	0.024	0.441

MDCM19

x	*y*	*z*	*q*
−6.505	0.122	−0.032	1.509
−5.812	0.039	−0.028	−4.913
−5.311	0.364	−0.024	3.432
−4.285	−1.906	0.064	−0.533
−1.185	−1.840	0.056	2.688
−1.259	−2.481	−1.031	−0.343
−1.199	−2.239	0.721	−0.715
2.382	0.268	0.052	3.209
−1.248	0.144	0.033	0.747
−0.262	0.567	0.028	−0.993
−4.537	1.682	−0.006	−0.868
3.258	0.177	0.056	−4.878
4.072	0.129	0.059	2.005
−5.682	−1.044	0.012	0.892
−5.185	−4.470	−0.001	0.107
−1.098	−1.429	−0.008	−1.702
−0.440	−4.578	0.048	0.070
−0.243	4.545	−0.006	0.094
−5.050	3.956	−0.001	0.193

Appendix B

RKHS toolkit: Tutorial and kernel decompositions

In the following, a tutorial on how to construct a potential energy surface (PES) with the reproducing kernel Hilbert space (RKHS) toolkit is given. Further, the explicit functional form of the most common one-dimensional kernel functions discussed in section 3.4 of chapter 3 and their decompositions (see Eq. 3.17 in chapter 3) are listed along with the enum values by which the respective kernels are identified in the RKHS toolkit.

Tutorial on PES construction using the RKHS toolkit

Here, a step-by-step guide on how to construct a PES from electronic structure data using the RKHS toolkit is given. It is recommended to download the ZIP archive "PES_Tutorial.zip" from `https://github.com/MeuwlyGroup/RKHS` to follow along with the examples. A full documentation of the capabilities of the RKHS toolkit is available from `https://github.com/MeuwlyGroup/RKHS`.

In general, the generation and use of a PES with the RKHS toolkit can be divided into three distinct steps:

1. Generation of reference data
2. Construction of kernel interpolations
3. Evaluation of the PES

This tutorial is a guide through the three necessary steps to generate and evaluate a PES for the H_3 system from scratch.

Step 1: Generation of reference data

In order to be able to generate reference data, first it must be decided how the PES should be represented. For a global reactive PES for a triatomic system, a many-body expansion is a straightforward approach:

$$V(r_{12}, r_{13}, r_{23}) = V^{(2)}(r_{12}) + V^{(2)}(r_{13}) + V^{(2)}(r_{23}) + V^{(3)}(r_{12}, r_{13}, r_{23}) + V^{\infty} \quad \text{(B.1)}$$

Here, r_{12}, r_{13} and r_{23} are the internuclear distances, $V^{(2)}$ are two-body terms, $V^{(3)}$ is a three-body term and V^{∞} is the energy at infinite separation of all three particles (constant). A different choice for the representation of the PES in other coordinate systems, e.g. hyperspherical[108–110] or Jacobi coordinates, would also be also possible.

Once the representation of the PES is decided, reference data can be generated. As a next step, meaningful coordinate grids to perform *ab initio* scans for the different terms (two-body and three-body) have to be chosen.

In order to speed up the calculations necessary for the tutorial, reference data is obtained by evaluating an existing PES for the H_3-system[242] instead of performing *ab initio* calculations. This also allows users to generate their own data set and test the obtained interpolation against the true reference PES, which is time intensive with *ab initio* reference data. For the purpose of this tutorial, a grid of 21 points (0.250, 0.500, 0.750, 0.900, 1.000, 1.100, 1.200, 1.300, 1.401, 1.500, 1.600, 1.700, 1.800, 1.900, 2.000, 2.250, 3.000, 4.000, 5.000, 6.000, 15.00 all in a_0) is chosen for all internuclear distances, but other choices would be possible. Note that in principle, scans for each two-body term and for the three-body term could use different grid points (and even entirely different coordinate systems) for each coordinate. The three-body grid therefore consists of a total $21^3 = 9261$ points out of which 5271 points are physically impossible (e.g. $r_{12} + r_{13} - r_{23} < 0$). However, this does not impede the applicability of the RKHS toolkit, since all unphysical points can simply be marked as "holes" (see section 3.3) in the grid by setting their value to NaN. It would also be possible to avoid unphysical points in the grid altogether by introducing a new coordinate system for the three-body term, e.g.

$$\begin{aligned} s_{12} &= r_{13} + r_{23} - r_{12} \\ s_{13} &= r_{12} + r_{23} - r_{13} \\ s_{23} &= r_{12} + r_{13} - r_{23} \end{aligned} \tag{B.2}$$

If the grid contains only values of $s > 0$, unphysical points do not exist. However, for simplicity, a coordinate grid using unmodified internuclear distances is chosen for this tutorial.

The code "**gendata.f90**" generates three CSV-files containing reference data (one for the two-body terms, one for the three-body term, and one file containing V^{∞}) as input for the RKHS toolkit (see next step). Refer to the source code for details such as the specific format of the CSV-files.

Step 2: Construction of kernel interpolations

With the reference data generated, the raw data needs to be processed in order to have meaningful input data for the RKHS interpolation. The asymptotic energy V^{∞} is subtracted from the *ab initio* reference data and a new CSV-file "**2body.csv**" is

generated (for details refer to the source code "**constructkernels.f90**"). In order to perform the RKHS interpolation, only a few subroutine calls are needed:

```
1  program constructkernels
2  use RKHS
3  implicit none
4  type(kernel) :: H2 !stores kernel for the 2-body term
5  call H2%read_grid("2body.csv")
6  call H2%k1d(1)%init(RECIPROCAL_POWER_N2_M5_KERNEL) !choose type
7  call H2%calculate_coefficients_fast()
8  call H2%calculate_sums() !calculate lookup table
9  call H2%save_to_file("2body.kernel")  !save to binary file
10 end program constructkernels
```

Other choices for the kernel function with different decay behaviour are possible by substituting the enum value in line 6. All implemented kernel functions are listed along with their enum values at the end of this appendix. To evaluate the RKHS interpolation it is sufficient to call

```
real(kind(0d0)) :: r !internuclear distance (input)
real(kind(0d0)) :: E !stores potential energy (output)
call H2%evaluate_fast((/r/),E) !fast evaluation
!or alternatively
call H2%evaluate_slow((/r/),E) !slow evaluation
```

Similarly, the input data for the three-body RKHS interpolation is generated by subtracting the asymptotic energy V^{∞} and all two-body contributions from the *ab initio* reference data and a new CSV-file "**3body.csv**" is generated (for details refer to the source code "**constructkernels.f90**"). The two-body contributions can either be taken from *ab initio* data or by evaluating the two-body kernel that was constructed previously. The RKHS interpolation of the three-body term is again performed using only a few subroutine calls:

```
program constructkernels
use RKHS
implicit none
type(kernel) :: H3 ! stores kernel for the 3-body term
call H3%read_grid("3body.csv")
call H3%k1d(1)%init(RECIPROCAL_POWER_N2_M5_KERNEL)
call H3%k1d(2)%init(RECIPROCAL_POWER_N2_M5_KERNEL)
call H3%k1d(3)%init(RECIPROCAL_POWER_N2_M5_KERNEL)
call H3%calculate_coefficients_slow()
call H3%calculate_sums()
call H3%save_to_file("3body.kernel")
end program constructkernels
```

Step 3: Evaluation of the PES

Now that the RKHS interpolations for two- and three-body terms have been generated, in order to evaluate the PES, the different terms need to be evaluated separately and added back together (see Eq. B.1). The addition of the constant term V^{∞} is optional and does not affect derivatives (and derived forces) of the PES. It is recommended to write a small interface for the PES evaluation (see **"pes.f90"**) that wraps the evaluation of the separate kernels and adds their individual contributions back

together (**"pes.f90"** exemplifies how to evaluate derivatives, see also the documentation available from `https://github.com/MeuwlyGroup/RKHS`). Note that the kernel construction from step 2 of the tutorial does not need to be repeated and kernels can be initialized directly from binary files. The code **"evaluatepes.f90"** evaluates the RKHS interpolated PES at several points and compares them to the values obtained using the analytical reference PES.[242] The largest error is 3.11×10^{-8} E_h and the mean is 3.68×10^{-9} E_h for reference energies which spans more than 18 E_h units.

Explicit functional forms of kernels and their decompositions

Reciprocal power decay kernel with $n = 2$ and $m = 0$ ($M_2 = 2$)

`RECIPROCAL_POWER_N2_M0_KERNEL`

$$k(x, x') = \frac{2}{x_>} - \frac{2}{3}\frac{x_<}{x_>^2}$$

$$p_{21} = 2 \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x}$$
$$p_{22} = -\frac{2}{3} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^2}$$

Reciprocal power decay kernel with $n = 2$ and $m = 1$ ($M_2 = 2$)

`RECIPROCAL_POWER_N2_M1_KERNEL`

$$k(x, x') = \frac{2}{3x_>^2} - \frac{1}{3}\frac{x_<}{x_>^3}$$

$$p_{21} = \frac{2}{3} \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x^2}$$
$$p_{22} = -\frac{1}{3} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^3}$$

Reciprocal power decay kernel with $n = 2$ and $m = 2$ ($M_2 = 2$)

`RECIPROCAL_POWER_N2_M2_KERNEL`

$$k(x, x') = \frac{1}{3x_>^3} - \frac{1}{5}\frac{x_<}{x_>^4}$$

$$p_{21} = \frac{1}{3} \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x^3}$$
$$p_{22} = -\frac{1}{5} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^4}$$

Reciprocal power decay kernel with $n = 2$ and $m = 3$ ($M_2 = 2$)

`RECIPROCAL_POWER_N2_M3_KERNEL`

$$k(x, x') = \frac{1}{5x_>^4} - \frac{2}{15}\frac{x_<}{x_>^5}$$

$$p_{21} = \frac{1}{5} \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x^4}$$
$$p_{22} = -\frac{2}{15} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^5}$$

Reciprocal power decay kernel with $n = 2$ and $m = 4$ ($M_2 = 2$)

`RECIPROCAL_POWER_N2_M4_KERNEL`

$$k(x, x') = \frac{2}{15x_>^5} - \frac{2}{21}\frac{x_<}{x_>^6}$$

$$p_{21} = \frac{2}{15} \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x^5}$$
$$p_{22} = -\frac{2}{21} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^6}$$

Reciprocal power decay kernel with $n = 2$ and $m = 5$ ($M_2 = 2$)

`RECIPROCAL_POWER_N2_M5_KERNEL`

$$k(x, x') = \frac{2}{21x_>^6} - \frac{1}{14}\frac{x_<}{x_>^7}$$

$$p_{21} = \frac{2}{21} \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x^6}$$
$$p_{22} = -\frac{1}{14} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^7}$$

Reciprocal power decay kernel with $n = 2$ and $m = 6$ ($M_2 = 2$)

`RECIPROCAL_POWER_N2_M6_KERNEL`

$$k(x, x') = \frac{1}{14x_>^7} - \frac{1}{18}\frac{x_<}{x_>^8}$$

$$p_{21} = \frac{1}{14} \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x^7}$$
$$p_{22} = -\frac{1}{18} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^8}$$

Reciprocal power decay kernel with $n = 3$ and $m = 0$ $(M_2 = 3)$

`RECIPROCAL_POWER_N3_M0_KERNEL`

$$k(x, x') = \frac{3}{x_>} - \frac{3}{2}\frac{x_<}{x_>^2} + \frac{3}{10}\frac{x_<^2}{x_>^3}$$

$$p_{21} = 3 \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x}$$
$$p_{22} = -\frac{3}{2} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^2}$$
$$p_{23} = \frac{3}{10} \qquad f_{23}(x) = x^2 \qquad f_{33}(x) = \frac{1}{x^3}$$

Reciprocal power decay kernel with $n = 3$ and $m = 1$ $(M_2 = 3)$

`RECIPROCAL_POWER_N3_M1_KERNEL`

$$k(x, x') = \frac{3}{4x_>^2} - \frac{3}{5}\frac{x_<}{x_>^3} + \frac{3}{20}\frac{x_<^2}{x_>^4}$$

$$p_{21} = \frac{3}{4} \qquad f_{21}(x) = 1 \qquad f_{31}(x) = \frac{1}{x^2}$$
$$p_{22} = -\frac{3}{5} \qquad f_{22}(x) = x \qquad f_{32}(x) = \frac{1}{x^3}$$
$$p_{23} = \frac{3}{20} \qquad f_{23}(x) = x^2 \qquad f_{33}(x) = \frac{1}{x^4}$$

Reciprocal power decay kernel with $n = 3$ and $m = 2$ $(M_2 = 3)$

`RECIPROCAL_POWER_N3_M2_KERNEL`

$$k(x, x') = \frac{3}{10x_>^3} - \frac{3}{10}\frac{x_<}{x_>^4} + \frac{3}{35}\frac{x_<^2}{x_>^5}$$

$$
\begin{aligned}
p_{21} &= \frac{3}{10} & f_{21}(x) &= 1 & f_{31}(x) &= \frac{1}{x^3} \\
p_{22} &= -\frac{3}{10} & f_{22}(x) &= x & f_{32}(x) &= \frac{1}{x^4} \\
p_{23} &= \frac{3}{35} & f_{23}(x) &= x^2 & f_{33}(x) &= \frac{1}{x^5}
\end{aligned}
$$

Reciprocal power decay kernel with $n = 3$ and $m = 3$ ($M_2 = 3$)

`RECIPROCAL_POWER_N3_M3_KERNEL`

$$k(x,x') = \frac{3}{20x_>^4} - \frac{6}{35}\frac{x_<}{x_>^5} + \frac{3}{56}\frac{x_<^2}{x_>^6}$$

$$
\begin{aligned}
p_{21} &= \frac{3}{20} & f_{21}(x) &= 1 & f_{31}(x) &= \frac{1}{x^4} \\
p_{22} &= -\frac{6}{35} & f_{22}(x) &= x & f_{32}(x) &= \frac{1}{x^5} \\
p_{23} &= \frac{3}{56} & f_{23}(x) &= x^2 & f_{33}(x) &= \frac{1}{x^6}
\end{aligned}
$$

Reciprocal power decay kernel with $n = 3$ and $m = 4$ ($M_2 = 3$)

`RECIPROCAL_POWER_N3_M4_KERNEL`

$$k(x,x') = \frac{3}{35x_>^5} - \frac{3}{28}\frac{x_<}{x_>^6} + \frac{1}{28}\frac{x_<^2}{x_>^7}$$

$$
\begin{aligned}
p_{21} &= \frac{3}{35} & f_{21}(x) &= 1 & f_{31}(x) &= \frac{1}{x^5} \\
p_{22} &= -\frac{3}{28} & f_{22}(x) &= x & f_{32}(x) &= \frac{1}{x^6} \\
p_{23} &= \frac{1}{28} & f_{23}(x) &= x^2 & f_{33}(x) &= \frac{1}{x^7}
\end{aligned}
$$

Reciprocal power decay kernel with $n = 3$ and $m = 5$ ($M_2 = 3$)

`RECIPROCAL_POWER_N3_M5_KERNEL`

$$k(x,x') = \frac{3}{56x_>^6} - \frac{1}{14}\frac{x_<}{x_>^7} + \frac{1}{40}\frac{x_<^2}{x_>^8}$$

$$
\begin{aligned}
p_{21} &= \frac{3}{56} & f_{21}(x) &= 1 & f_{31}(x) &= \frac{1}{x^6} \\
p_{22} &= -\frac{1}{14} & f_{22}(x) &= x & f_{32}(x) &= \frac{1}{x^7} \\
p_{23} &= \frac{1}{40} & f_{23}(x) &= x^2 & f_{33}(x) &= \frac{1}{x^8}
\end{aligned}
$$

Reciprocal power decay kernel with $n = 3$ and $m = 6$ ($M_2 = 3$)

`RECIPROCAL_POWER_N3_M6_KERNEL`

$$k(x, x') = \frac{1}{28x_>^7} - \frac{1}{20}\frac{x_<}{x_>^8} + \frac{1}{55}\frac{x_<^2}{x_>^9}$$

$$
\begin{aligned}
p_{21} &= \frac{1}{28} & f_{21}(x) &= 1 & f_{31}(x) &= \frac{1}{x^7} \\
p_{22} &= -\frac{1}{20} & f_{22}(x) &= x & f_{32}(x) &= \frac{1}{x^8} \\
p_{23} &= \frac{1}{55} & f_{23}(x) &= x^2 & f_{33}(x) &= \frac{1}{x^9}
\end{aligned}
$$

Exponential decay kernel with $n = 2$ ($M_2 = 2$)

`EXPONENTIAL_DECAY_N2_KERNEL`

$$k(x, x') = 4\frac{\mathrm{e}^{-\beta \mathrm{x}_>}}{\beta^3}\left[\beta(x_> - x_<) + 2\right]$$

$$
\begin{aligned}
p_{21} &= 4 & f_{21}(x) &= 2 - \beta x & f_{31}(x) &= \frac{\mathrm{e}^{-\beta \mathrm{x}}}{\beta^3} \\
p_{22} &= 4 & f_{22}(x) &= \beta & f_{32}(x) &= x\frac{\mathrm{e}^{-\beta \mathrm{x}}}{\beta^3}
\end{aligned}
$$

Exponential decay kernel with $n = 3$ ($M_2 = 5$)

`EXPONENTIAL_DECAY_N3_KERNEL`

$$k(x, x') = 18\frac{\mathrm{e}^{-\beta \mathrm{x}_>}}{\beta^5}\left[\beta^2(x_> - x_<)^2 + 6\beta(x_> - x_<) + 12\right]$$

$$f_{31}(x) = \frac{\mathrm{e}^{-\beta \mathrm{x}}}{\beta^5}$$

$$\begin{array}{lll} p_{21} = 18 & f_{21}(x) = 12 - 6\beta x & f_{32}(x) = x\dfrac{\mathrm{e}^{-\beta \mathrm{x}}}{\beta^5} \\ p_{22} = 18 & f_{22}(x) = 6\beta & \\ p_{23} = 18 & f_{23}(x) = (\beta x)^2 & f_{33}(x) = \dfrac{\mathrm{e}^{-\beta \mathrm{x}}}{\beta^5} \\ p_{24} = 18 & f_{24}(x) = -2\beta^2 x & \\ p_{25} = 18 & f_{25}(x) = \beta^2 & f_{34}(x) = x\dfrac{\mathrm{e}^{-\beta \mathrm{x}}}{\beta^5} \\ & & f_{35}(x) = x^2\dfrac{\mathrm{e}^{-\beta \mathrm{x}}}{\beta^5} \end{array}$$

Taylor spline kernel with $n = 2$ ($M_2 = 3$)

`TAYLOR_SPLINE_N2_KERNEL`

$$k(x, x') = 1 + x_< x_> + 2x_<^2 x_> - \frac{2}{3}x_<^3$$

$$\begin{array}{lll} p_{21} = -\dfrac{2}{3} & f_{21}(x) = x^3 - \dfrac{3}{2} & f_{31}(x) = 1 \\ p_{22} = 1 & f_{22}(x) = x & f_{32}(x) = x \\ p_{23} = 2 & f_{23}(x) = x^2 & f_{33}(x) = x \end{array}$$

Taylor spline kernel with $n = 3$ ($M_2 = 5$)

`TAYLOR_SPLINE_N3_KERNEL`

$$k(x, x') = 1 + x_< x_> + x_<^2 x_>^2 + 3x_>^2 x_<^3 - \frac{3}{2}x_> x_<^4 + \frac{3}{10}x_<^5$$

$$\begin{array}{lll} p_{21} = \dfrac{3}{10} & f_{21}(x) = x^5 + \dfrac{10}{3} & f_{31}(x) = 1 \\ p_{22} = -\dfrac{3}{2} & f_{22}(x) = x^4 & f_{32}(x) = x \\ p_{23} = 3 & f_{23}(x) = x^3 & f_{33}(x) = x^2 \\ p_{24} = 1 & f_{23}(x) = x & f_{34}(x) = x \\ p_{25} = 1 & f_{23}(x) = x^2 & f_{35}(x) = x^2 \end{array}$$