Chemical Structure Prediction Based on Machine Learning

Von der Fakultät Chemie der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Alexander Denzel

aus Tettnang

Hauptberichter	Prof. Dr. Johannes Kästner
Mitberichter	Prof. Dr. Oriol Vendrell
Prüfungsvorsitzender	Prof. Dr. Blazej Grabowski
Tag der mündlichen Prüfung:	23.01.2020

Institut für Theoretische Chemie
Universität Stuttgart

2020

Acknowledgement

The thesis would not have been possible if not for many people that supported me in a direct or indirect way. I want to sincerely thank all of them. Especially, and most of all I want to thank Johannes Kästner. He did not only provide a very free, flexible, and comfortable working environment but also contributed to this thesis through his frequent questions, comments, and ideas on the topic. I would also like to give special thanks to Oriol Vendrell for agreeing on becoming the second examiner as well as Blazej Grabowski for examining the thesis and taking the chair of the examination board.
During the time I worked on this thesis I got a lot of support from the helpful chemists in the group to compensate my lack of chemical knowledge. In particular I would like to thank our two spanish post-docs Germán Molpeceres de Diego and Sonia Álvarez Barcia who often answered my questions on chemistry and the respective software usage. Furthermore, Jan Meisner, Mark Pfeiffle and Max Markmeyer provided some helpful backgrounds on chemistry if I needed it. I would also like to thank April Cooper for the discussions we had on several machine learning topics. For mathematical support, I would like to thank Bernard Haasdonk for his very helpful comments on my work. His insight helped to improve the algorithms presented in this thesis. I also want to thank the rest of the institute for the nice discussions we had during tea time and lunch. They often helped to improved my capability of finding errors and solutions and were also a nice entertainment of course.

Outside of the university I would especially like to thank my parents for their constant moral and also financial support during my studies and the nice holidays I could spent at their place. Without them I would not have accomplished the presented work. Furthermore, I would like to thank Lena and my friends, especially Dave, Georg, Franky, Katrin, Malte, Phillip, Schölli, Thomas, and Tim that supported me in several non-scientific ways during the work for this thesis. Thanks guys!

Contents

Index of Abbreviations

AM1 Austin Model 1; semi-empirical method electronic structure calculations [27]

Å Ångström; unit of length; $1\text{Å} = 10^{-10}\text{m}$

DFT Density functional theory

E_h Hartree, the atomic unit of energy, $E_\text{h} = 4.359744650 \cdot 10^{-18}\text{J}$

GP Gaussian process

GPR Gaussian process regression

GPR-PES The GPR-based surrogate for the PES

GPRMEP GPR-based optimizer presented in this thesis to find a MEP

GPRTS GPR-based optimizer presented in this thesis to find a TS

IDPP Image dependent pair potential; a potential that is constructed only by geometrical reasoning that can be used for pre-optimization in the NEB method [76]

a_0 Bohr radius; atomic unit of length; $\text{a}_0 = 5.29 \cdot 10^{-11}\text{m}$

L-BFGS The limited memory version of the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)

MEP Minimum energy path; it is the path that connects two minima (reactant/product) which has the lowest energy of all such paths

ML Machine learning

NEB Nudged elastic band method, see section 2.5

P-RFO Partitioned rational function optimization, see section 2.3

PES The potential energy surface describing the electronic energy to a fixed position of the nuclei within the Born–Oppenheimer approximation

RMSD root-mean-square deviation; measures the difference of two geometries in a molecule; $\sqrt{\frac{\sum_{i=1}^{d}\left(x_1^i - x_2^i\right)^2}{d}}$ with x_j^i being the i-th coordinate of the vector representing a configuration $\mathbf{x}_j$ of the system and d the number of dimensions in the system

TS Transition structure, often also called transition state; it refers to a first-order saddle point on the PES that connects two minima (reactant/product); it is the point of highest energy on the MEP

Abstract

In this thesis, the machine learning method called *Gaussian process regression* (GPR) is used to implement different types of optimization algorithms for several kinds of chemically relevant, geometrical structures. In the introduction part, the necessity for fast geometry optimizers is argued for and some of the existing optimization algorithms are explained. After the introduction, a summary of the existing GPR theory is given, see section II. It is shown how Gaussian processes can be used as a regression scheme. Furthermore, its statistical properties are highlighted that allow, e.g., to estimate uncertainties. The explanations given in the literature are complemented, trying to make GPR understandable without large prior knowledge. The following part, see section III, is the core of this thesis and describes the author's own work in detail. It is shown that even without making use of the statistical properties, GPR is a reliable regression scheme for learning energies from *ab-initio* calculations. The GPR-based approximation of the potential energy surface is used as a surrogate model on which optimizations are performed. The results of these optimizations are used to estimate the desired geometry on the real energy surface. The surrogate is thereby iteratively improved by adding more and more training data and giving a better estimate for the desired geometry in later optimization steps until the optimizer is converged. It is shown in several test cases that the developed algorithms find the desired geometries faster and in many cases more reliably than it was possible before. All software developed in this thesis has been implemented in DL-FIND and is therefore open source. It is also interfaced to ChemShell, which makes it usable with several electronic structure codes. In the rest of the abstract, the key ideas of the author's work is summarized.

Implementation of Gaussian Process Regression

The presented implementation of GPR has some extra features compared to other implementations that improve its capabilities for geometry optimization. The implementation allows to arbitrarily incorporate data with different derivative information (up to second order), is sped up dramatically by an iterative implementation of the Cholesky decomposition, allows the handling of larger systems due to a multi-level scheme, and has implemented several other small features, see section 6, that make it better suited for geometry optimization than other available libraries.

Geometry Optimizier/Minimizer

The first developed optimizer shows how one can speed up minimization processes in theoretical chemistry by using GPR as a surrogate model. It is based on L-BFGS optimization on the surrogate model and requires only energy and gradient information. Two different overshooting processes are presented that overstep the estimated minimum on purpose. One which simply scales up the step taken by the optimizer and one which scales up the step in specific dimensions. These overshooting procedures are uniquely usable in surrogate-based optimizers as the one presented. They allow for large step sizes which can speed up optimizations. And if they overshoot the minimum too much, they still improve the regression capabilities of the surrogate model, which improves further estimates of the minimum. This is in contrast to classical optimizers in which such estimates are often just discarded. The resulting algorithm outperforms classical L-BFGS optimization, which also serves as a proof of concept that surrogate models can improve the performance of classical optimizers.

Transition State Search

The presented, new algorithm for searching first-order saddle points is based on using the P-RFO method on the surrogate model. The presented algorithm, in contrast to P-RFO, still only needs gradient information and does not require ab-initio Hessians. Some ideas of the minimizer are adapted to improve the performance of the transition state search. The algorithm is compared to classical P-RFO with updated Hessians and to the dimer method. The new algorithm outperforms both for the cases that were tested. It needs much fewer energy/gradient evaluations and therefore, easily compensates its computational overhead that arises due to the GPR model being used.

Minimum Energy Path Optimizer

A completely new method for optimizing minimum energy paths is presented. It is based on minimizing two different loss functions, one based on energy, one based on forces. Both loss functions are combined and, again, only energy and gradient evaluations from the electronic structure calculations are needed. The method is also based on a surrogate model created with GPR that can be reused in a following transition state search. This makes it possible to reliably find a transition state only from the two given minimum structures that it connects. Furthermore, the optimizer can guarantee that the discretization of the optimized path is evenly spaced. The method outperforms the classical NEB method in the presented test cases and makes the search for minimum energy paths and transition states faster than before.

Updating Hessians

In a rather straightforward approach, it is possible to use GPR to update Hessians using gradients. This is often done with classical update schemes like the one by Bofill. As a proof of concept it is shown that P-RFO can be clearly improved using a GPR-based update scheme for Hessians compared to the classical alternatives.

Kurzzusammenfassung

Im Rahmen dieser Arbeit wird Gaußprozess-Regression (GPR), eine Methode des maschinellen Lernens, verwendet, um verschiedene Typen von Optimierungsalgorithmen für chemisch relevante, geometrische Strukturen zu entwickeln. In der Einleitung wird die Motivation für die Entwicklung schneller Geometrieoptimierer erklärt und einige bisher existierende Optimierer vorgestellt. Nach der Einleitung folgt eine Zusammenfassung der Theorie zur GPR, siehe Teil II. Es wird gezeigt wie Gaußprozesse zur Regression verwendet werden können. Außerdem wird auf die statistischen Eigenschaften der Methode eingegangen, die es z.B. erlauben, Unsicherheiten zu quantifizieren. Dabei wird versucht, die Erklärungen aus der Literatur zu erweitern um sie verständlicher zu machen, auch ohne viele Kenntnisse der Thematik vorauszusetzen. Der darauffolgende Teil, siehe Teil III, ist der Kern dieser Dissertation und beschreibt die eigene Arbeit des Autors im Detail. Es wird gezeigt, dass GPR auch ohne den statistischen Hintergrund bereits eine verlässliche Methode zur Regression von Energien aus *ab-initio* Berechnungen ist. Die resultierende Regressionsfunktion, die die Energie nachbildet, wird als ein Modell für die echte Energiefunktion verwendet. Auf ihr werden Optimierungen durchgeführt, um die gewünschte Geometrie zur realen Energiefunktion abzuschätzen. Das Modell wird dabei iterativ, durch Zufügen einer wachsenden Zahl von Energien, die im Laufe der Optimierung berechnet werden, verbessert. Dies verbessert die Vorhersagequalität des Modells schrittweise, bis letztlich die Schätzung der realen Geometrie ausreichend gut ist um eine konvergierte Geometrie zu erhalten. An Hand diverser Testsysteme wird gezeigt, dass die entwickelten Optimierer es ermöglichen die gewünschten Geometrien schneller und oft auch zuverlässiger als zuvor zu finden. Alle in dieser Arbeit entwickelte Software wurde in DL-FIND implementiert und ist dementsprechend quelloffen. Außerdem ist eine Schnittstelle zu ChemShell entwickelt worden, die es ermöglicht, verschiedenste

Elektronenstrukturprogramme zu verwenden. Im Rest der Kurzzusammenfassung werden die zentralen Ergebnisse der vorliegenden Arbeit kurz erläutert.

Implementierung der Gaußprozess-Regression

Die vorgestellte Implementierung der GPR weist einige Besonderheiten auf, die sie von anderen GPR-Implementierungen unterscheidet und besonders geeignet für Geometrieoptimierung in der Chemie macht. Sie erlaubt z.B. die beliebige Verwendung von Trainingsdaten mit Ableitungen unterschiedlicher Ordnung (bis zur zweiten Ordnung) und wird signifikant beschleunigt durch die Verwendung einer iterativen Implementierung der Cholesky-Zerlegung. Weiterhin erlaubt sie die Handhabung größerer Systeme durch die Verwendung einer Methode, die mehrere Ebenen von Gaußprozessen kombiniert und hat einige weitere kleine Besonderheiten, siehe Abschnitt 6, durch welche sie besser für die Geometrieoptimierung geeignet ist als andere verfügbare Bibliotheken.

Optimierung von Minima

Der erste Optimierer, der im Laufe dieser Arbeit entwickelt wurde, zeigt, wie man Minimierungsprozesse in der theoretischen Chemie durch die Verwendung eines GPR-Modells beschleunigen kann. Er basiert auf einer L-BFGS Optimierung auf dem GPR-Modell und benötigt ausschließlich Energien und Gradienten. Zwei verschieden *overshooting* Methoden wurden implementiert, die bewusst über das vermutete Minimum hinausschießen. Eine dieser Methoden skaliert einfach die geschätzte Schrittweise durch einen Faktor größer als eins. Die zweite *overshooting* Methode führt das selbe durch, allerdings nur für ausgewählte Dimensionen des Schrittvektors. Diese beiden Methoden erlauben dem Algorithmus besonders große Optimierungsschritte auszuführen, die die Optimierung beschleunigen können. Sollten die entsprechenden Schritte zu weit über die gewünschte Struktur hinausschießen, so können die resultierenden Geometrien genutzt werden um das GPR-Modell in diesem Bereich zu verbessern. Folglich sind auch weitere Schätzungen der gewünschten Struktur deutlich besser möglich. Dies steht ganz im Gegensatz zur Vorgehensweise bei klassischen Optimierern, bei der solche Punkte häufig verworfen werden. Der präsentierte Algorithmus übertrifft die Effizienz des klassischen L-BFGS Optimierers. Weiterhin zeigt dies die grundsätzliche Möglichkeit, dass klassische Optimierer mit entsprechenden maschinellen Lernverfahren deutlich beschleunigt werden können.

Übergangszustandssuche

Der in dieser Arbeit präsentierte, neue Algorithmus für die Suche von Sattelpunkten erster Ordnung basiert auf der Idee eine P-RFO Optimierung auf dem GPR-Modell durchzuführen. Der vorgestellte Algorithmus braucht aber dennoch, im Gegensatz zu P-RFO, ausschließlich Energien und Gradienten und keine ab-initio Hesse-Matrizen. Einige Ideen des Minimierers werden auch in diesem Optimierer umgesetzt um seine Leistungsfähigkeit zu steigen. Der Optimierer wird sowohl mit klassischer P-RFO mit aktualisierten Hesse-Matrizen, wie auch mit der *dimer*-Methode verglichen. Der neue Optimierer übertrifft die klassischen Optimierer deutlich in den vorgestellten Testsystemen. Er braucht beträchtlich weniger Energie-/Gradientenauswertungen und kann dadurch die benötigte Rechenzeit für die Verwendung des GPR-Modells einfach kompensieren.

Optimierung des Pfades Geringster Energie

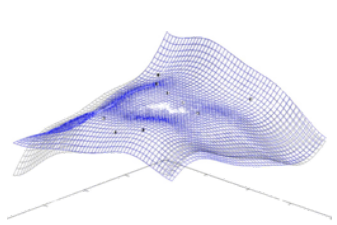
Eine komplett neue Methode der Optimierung für Pfade geringster Energie wird vorgestellt. Sie basiert auf der Minimierung zweier Kostenfunktionen, die eine basierend auf der Energie, die andere basierend auf Kräften. Beide Kostenfunktionen werden kombiniert. Wie bereits die anderen Optimierer, benötigt dieser Algorithmus lediglich Energien und Gradienten der ab-initio Berechnungen und keine Hesse-Matrizen. Der Optimierer basiert ebenso auf der Verwendung eines GPR-Modells, das jedoch in einer anschließenden Suche nach Übergangszuständen wiederverwendet werden kann. Das macht es möglich, verlässlich Übergangszustände zwischen zwei gegebenen Minima zu finden. Außerdem garantiert der Optimierer eine gleichmäßige Diskretisierung entlang des Pfades. Die Methode übertrifft die klassische NEB-Optimierung in den präsentierten Testsystemen und ermöglicht eine schnellere Suche von Pfaden geringster Energie als zuvor.

Aktualisierung von Hesse-Matrizen

Mit sehr geringem Aufwand ist es möglich GPR zur Aktualisierung von Hesse-Matrizen mit Hilfe von Gradienten zu verwenden. Dies wird üblicherweise mit klassischen Methoden wie der von Bofill durchgeführt. Im Rahmen eines kurzen Machbarkeitsnachweises wird gezeigt wie P-RFO durch die GPR-basierte Methode klar verbessert werden kann, verglichen mit klassischen Methoden zur Aktualisierung der Hesse-Matrizen.

Part I

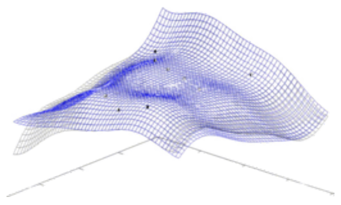
Introduction



1 Theoretical Chemistry and Geometry Optimization

The idea of theoretical chemistry is to explain and predict chemical phenomena by using theoretical methods. It includes rather intuitive concepts like chemical bonds, orbitals, valency, potential energies and other approaches that are intended to facilitate the human understanding of chemistry. On the other hand, it also makes use of rather abstract methods from mathematics and physics to allow for explicit calculation of several chemical properties. For example to explain the principles of chemistry on a very fundamental level one deploys the laws of quantum mechanics, which opens the field of *quantum chemistry*. The most central equation in this field is the Schrödinger equation, postulated first in 1926. In theory one can explain all chemical phenomena by solving the Schrödinger equation. Unfortunately, it is a partial differential equation whose solutions are not easily obtained. In fact, analytical solutions are only available for the smallest of all molecules, atomic hydrogen and the hydrogen molecular ion ${H_2}^+$. Nowadays, one can hope to approximate the solutions of other chemical systems using elaborate computer algorithms.

Therefore, a large part of research in theoretical chemistry is concerned with developing methods to approximate the solutions to the Schrödinger equation in a fast and reliable way. The most commonly used approximation is the Born–Oppenheimer approximation. It assumes that the motion of atomic nuclei and electrons can be treated separately since their masses are very different and, therefore, the time-scales of their motion is very different as well. Under the Born–Oppenheimer approximation one can solve the Schrödinger equation only for the movement of the electrons, also called the *electronic Schrödinger equation*, given a fixed configuration of the atomic nuclei. This yields a specific energy of the electronic structure



for the respective configuration of the nuclei. The resulting mapping from the configuration space of the nuclei to the electronic energy of the system is called the *potential energy surface* and abbreviated with PES in the rest of the thesis. Its evaluation for a single configuration of the nuclei requires to solve the electronic Schrödinger equation. Unfortunately, even solving the electronic Schrödinger equation in contrast to the full Schrödinger equation is computationally intensive, even with modern computer algorithms. Depending on the approximations made and on the chemical system considered, finding a single solution to this equation can last from seconds to days. In many cases the solution is completely out of reach and further approximations have to be made. The usage of computers to solve quantum chemistry problems touches another field of theoretical chemistry, namely *computational chemistry*. In computational chemistry one employs computer codes to obtain all sorts of chemical properties that are otherwise out of reach. And the probably most frequently desired property in computationally chemistry is the shape or configuration of a chemical system. To find these one has to perform a *geometry optimization*.

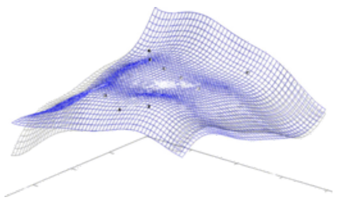
1.1 Geometry Optimization

The spatial arrangement of the atoms of a molecule or a chemical system is also called a *geometry*. It is mostly interesting to obtain chemically stable geometries and geometries that occur along chemical reaction pathways. The former correspond to minima on the PES and are therefore the structures that are most likely to exist in the physical world. The latter are particularly interesting if they are first-order saddle points on the PES. They can then be used in *transition state theory* to obtain insight on the rate of a chemical reaction to estimate how likely it is that a chemical reaction takes place and under which circumstances. One determines the eigenvalues of the Hessian of the PES at these geometries. The eigenvalues are used to calculate the vibrational partition function of the transition state. From that one can estimate reaction rate constants. In this framework the negative eigenvalue of the Hessian corresponds to a transition from the reactant to the product regime. In some cases it is also of interest to find a complete, classical reaction path, a *minimum energy path* (MEP). It explains the proceeding of a chemical reaction and is often used to facilitate the search for a first-order saddle point which is the point of highest energy on a MEP. In some theories, like *small curvature tunneling* the MEP is also used to calculate semi-classical reaction rates that also incorporate quantum effects like tunneling to a certain extent. Theoretical chemists that are, e.g., concerned with reaction kinetics spend a lot of time on finding the said geometries. To find a converged geometry can often be very tedious and might need a lot of human input. Therefore, developing faster and especially more stable algorithms for geometry optimizations can massively improve many future works in computational chemistry. The currently available algorithms are not necessarily stable in every case or require much computational power. Fore example, some algorithms require the calculation of Hessians of the PES. For many electronic structure calculations this is not feasible. Ideally, a geometry optimizer works as a black box-like system that only needs very little human input and still converges in a stable way without requiring information about the Hessian. It also should require as few energy and gradient evaluations of the PES as possible. In this thesis new approaches to attack this problem are presented that are based on the concepts of machine learning.

1.2 Machine Learning

There are several possible ways of defining *machine learning* (ML). In the following this is tried by highlighting the contrast to classical, non-ML algorithms.

In non-ML algorithms one usually implements known rules in algorithms. The resulting program operates on input data based on these rules and calculates the desired result. In computational chemistry traditional *ab initio* calculations are an example of such algorithms. The basic rules that are implemented are the laws of quantum mechanics. The algorithms apply these rules to given input data, e.g. a molecule, and calculate a resulting property, e.g. the energy of that molecule, based on the laws of quantum mechanics. Alternatively one can often think of classical programming as *model-based programming*: One defines a model for a specific problem, e.g. a mathematical description of a molecule, and implements



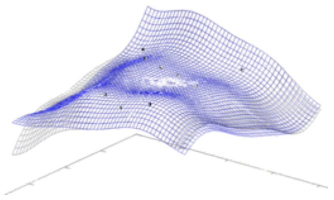
an algorithm that solves the equations of the model. The better the model, the more accurate the solution of the algorithm will be. This requires a huge effort in building adequate models and also the solution of the equations of the model can become tedious. This is often the case for problems in computational chemistry. Therefore, one can consider non-ML algorithms as *model-driven.*

In contrast to that ML algorithms can often be considered as *data-driven.* As a programmer, one does not set up a model to describe the desired system and one does not lay down rules like laws of physics. Instead, the programmer only provides data and lets the computer itself learn the underlying physical property. However, it does not learn the underlying laws of physics but only the results that are manifested in the learned physical property. In other words, ML can be viewed as a learning-by-example approach. This is pretty similar to human learning: We learn to recognize objects or words with examples simply by looking or hearing. From that we can abstract and build a more complex understanding of the world. In many cases ML algorithms are able to mimic this behavior and recognize objects, words, etc. in data.

The most prominently employed methods to achieve these goals are *neural networks.* Their possibility to abstract data on a high level and recent advances in computational capabilities helped neural networks and other ML methods to gain large popularity around scientists in many fields. But if it comes to data-driven models, neural networks are not always the method of choice. Neural networks require huge amounts of data to learn reliable information and their training process can become computationally expensive. Depending on the task that is to be performed, other ML methods might be more suitable.

In this thesis the focus lies on another data-driven ML method called *Gaussian process regression* that was first introduced in 1951 by Danie Krige. Therefore, it is often also called *kriging.* It is able to approximate a function with rather few samples of that function, i.e. with rather few *training points.* Therefore, it is extremely suitable for geometry optimization in chemistry: Only few training data are available; at the beginning often only one single data element.

In this thesis it is shown how Gaussian process regression can be used to optimize chemical geometries faster and often more reliable than before. But since ML is all about data, do not take my word for it...
Let the data speak for themselves.

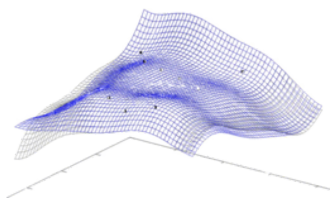


2 Commonly Used Optimization Algorithms in Theoretical Chemistry

In this part existing optimizers are reviewed that are used in theoretical chemistry. The main focus lies on L-BFGS [57] and P-RFO [10] since they are used in the developed optimizers. None of these algorithms were implemented or changed in the work for this thesis. They were simply used in the form of existing computer code implemented in DL-FIND [42].

2.1 Limited-Memory Broyden–Fletcher–Goldfarb–Shanno

There exist several methods to find stationary points of a function $f(x)$, i.e. points x_0 at which $\frac{d}{dx}f(x)|_{x=x_0} = 0$, or in multiple dimensions $\nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x_0}} = \mathbf{0}$. The most prominent one is Newton's method for optimization problems. However, it requires the second derivative of the function $f(x)$. Often, like for many cases in theoretical chemistry, the second derivatives are not available or very expensive to obtain. Quasi-Newton methods are modifications of the Newton method that are based on the same idea but do not use the Hessian explicitly. They only approximate or update the second derivative by analyzing multiple gradient vectors instead. L-BFGS is an example of such a method and stands for *Limited-Memory Broyden–Fletcher–Goldfarb–Shanno*. It is a slight modification suggested in 1980 by Nocedal [57] of the original BFGS method that was suggested independently by Broyden [18], Fletcher [32], Goldfarb [34], and Shanno [73] in 1970. The limited-



memory version has, as the name suggests, much lower memory requirement.

The update for the Hessian in BFGS is chosen in a way that it guarantees that the Hessian matrix stays symmetric and is positive definite. The BFGS-update of a Hessian H_k at position $\mathbf{x}_k$ to a Hessian H_{k+1} at position $\mathbf{x}_{k+1}$ is given by

$$H_{k+1} = H_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{H_k \mathbf{s}_k \mathbf{s}_k^T H_k^T}{\mathbf{s}_k^T H_k \mathbf{s}_k} \tag{2.1}$$

with $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ and $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$. For the algorithm to effectively minimize a function one does the following:

- Obtain a direction in which to jump by solving

$$H_k \mathbf{v} = -\nabla f(\mathbf{x}_k) \tag{2.2}$$

- Optionally one can perform a one-dimensional line search to find an acceptable step size in which to elongate in direction $\mathbf{v}$ to find the next point $\mathbf{x}_{k+1}$.
- Calculate the gradient at $\mathbf{x}_{k+1}$. Check for convergence.
- If not converged, update the Hessian according to equation (2.1)

These steps are repeated until convergence. Often one does not update the Hessian but the inverse of the Hessian since this is required to solve equation (2.2).

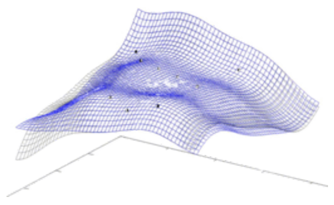
$$H_{k+1}^{-1} = \left(I - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) H_k^{-1} \left(I - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \tag{2.3}$$

The important thing to note for BFGS is that it guarantees a positive definite update of the Hessian and keeps it symmetric. The positive definiteness might be advantageous if one is close to a minimum. If the distance to the minimum is too large, this might become problematic since the real Hessian is not positive definite there. Also the optimizer cannot be used to find saddle points.

The memory requirement for the Hessian is often very large. To avoid the resulting memory problems the limited memory version of BFGS (L-BFGS) does not store the Hessian explicitly but only stores some vectors that are necessary for the update of the Hessian. These vectors consist of the $\mathbf{s}_k$ and $\mathbf{y}_k$ above. Then one defines the initial Hessian as a diagonal matrix, usually the identity matrix. It's memory demand is also linear in the dimension of the system. All updates of the Hessian or its inverse could then be calculated using a loop over the stored vectors but at no point in this procedure a full Hessian is calculated. The Hessian is updated only implicitly and a series of vectors is calculated that converges to the desired direction in which to jump. The number of vectors stored to update the Hessian is also limited to a number m which further decreases the memory requirement. Since these are primarily algorithmic details be referred to the original publication for a more detailed explanation [57]. For the present work it is sufficient to keep in mind that L-BFGS is designed to work for high-dimensional systems in regions of positive definite Hessians.

2.2 Trust Region and Acceptance of Steps

Usually L-BFGS can be combined with the concept of a *trust region*. There is only a certain region in which one trusts the results of the optimizer. It can be understood as a very simple version of a line search in which one simply scales down the step size to the radius of the trust region if the step size would be larger than that. In DL-FIND the trust radius is chosen based on the so-called *Wolfe-conditions* [81,82], proposed in 1969, which are often used to estimate whether the step size is adequately chosen, i.e. that the chosen step size decreases the function and the slope sufficiently. They build a simple estimate of choosing an acceptable trust region. When it is mentioned in later sections that *a trust radius based on energy decrease* is chosen, e.g. for section 7, this is the method referred to.
Furthermore, one usually only accepts steps that improve the result of the algorithm. A step that leads to a higher function value or a larger gradient may be a bad choice. The step is rejected and the trust region is reduced. This is similar to a line search in which one decreases the step size to find a better choice for it.



2.3 Partitioned Rational Function Optimization

A method to find first-order saddle points reliably is the P-RFO method. P-RFO stands for partitioned rational function optimization and is based on the idea of rational function optimization (RFO). The basic idea of RFO is based on the standard Newton-Raphson step

$$\Delta \mathbf{x} = -\sum_i (\mathbf{v}_i^T \mathbf{g}) \mathbf{v}_i / \lambda_i \tag{2.4}$$

with the gradient $\mathbf{g}$, the Hessian's eigenvalues λ_i, and the respective eigenvectors $\mathbf{v}_i$. The eigenvalues are shifted by a parameter γ so that the algorithm is walking uphill in the direction of the lowest eigenvector and downhill along all other eigenvectors.

$$\Delta \mathbf{x} = -\sum_i (\mathbf{v}_i^T \mathbf{g}) \mathbf{v}_i / (\lambda_i - \gamma) \tag{2.5}$$

In the original publication [10] it is shown that one can choose a certain rational function approximation of the function that leads to the same equation. This approximation allows to calculate a suitable value for λ in an iterative way. To find first-order saddle points the stability of the algorithm can be increased. It is shown that one can reformulate the problem into two separate optimization problems, one for maximizing in a certain direction $\mathbf{v}_k$ and one for minimizing in the other directions. Let the k-th eigenvector be the one corresponding to the smallest eigenvalue. One wishes to maximize the function in the direction of this eigenvector. One obtains two smaller partitioned RFO problems, hence the name P-RFO, with two shift parameters γ_p and γ_n. Both parameters are determined by solving an eigenvalue problem [10] and the resulting step vector is

$$\Delta \mathbf{x} = -\sum_i (\mathbf{v}_i^T \mathbf{g}) \mathbf{v}_i / (\lambda_i - \gamma_i) \tag{2.6}$$

in which $\gamma_i = \gamma_p$ for $i = k$ and $\gamma_i = \gamma_n$ for all other values of i. The P-RFO algorithm does the following:

- Diagonalize the Hessian
- Identify the eigenmode to be followed (the smallest one that is not a zero-

mode due to rotational and translational invariances)

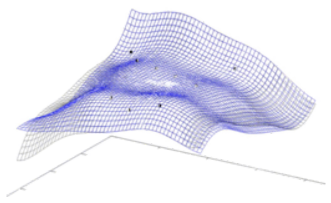
- Compute the shift parameters λ_p and λ_n, see [10]
- Compute the step $\Delta\mathbf{x}$ in equation (2.6) and potentially limit the step size
- Calculate the new gradient at the new position
- Check convergence and abort if the algorithm converged
- Recalculate or update the Hessian at that position

Important to remember for this thesis is that P-RFO allows the convergence to a first-order saddle point. It requires Hessians (or updated Hessians) and needs to diagonalize the Hessian which is all computationally expensive. For details on the algorithm, be referred to the literature [10,40].

2.4 Dimer Method

The dimer method [38,40,43] is a way of following the eigenvector corresponding to the smallest eigenvalue of the Hessian to a first-order saddle point, a so called *minimum mode following method* [47,59,62,83]. Approximating the minimum-mode can be done by rotating a dimer (dimer method) or building a Krylow-subspace (Lanczos method). Since only the dimer method is used in this thesis this approach is explained in the following. A dimer consists of two points in the configuration space that lie closely together. The basic idea is that one can approximate the minimum mode of the Hessian, i.e. the smallest eigenvalue and the respective eigenvector, by rotating a dimer around a fixed point and performing only gradient, not Hessian evaluations at the endpoints of the dimer. This is done by minimizing the rotational force $\mathbf{F}_{\text{rot}}$ acting on the dimer. To calculate this force one just needs the gradients $\mathbf{g}_1$ and $\mathbf{g}_2$ at the endpoints of the dimer to calculate the torque acting on these endpoints.

$$\mathbf{F}_{\text{rot}} = -(\mathbf{g}_1 - \mathbf{g}_2) + [(\mathbf{g}_1 - \mathbf{g}_2) \cdot \tau]\, \tau \tag{2.7}$$



When this optimization is done the dimer direction τ points in the direction of the minimum mode. One can then follow the force

$$\mathbf{F}_{\perp} = -\mathbf{g}_0 + 2(\mathbf{g}_0 \cdot \tau)\tau \tag{2.8}$$

with the gradient $\mathbf{g}_0$ at the midpoint of the dimer. This force drives the algorithm in a direction that increases the function value in the direction of the minimum mode τ but maximizes the function value in all other directions. Following this force up to a certain distance gives a translational step by which the dimer midpoint is translated. One then alternates between rotations and translational steps until one obtains a small gradient at the dimer midpoint. The big advantage of the dimer method compared to second order methods like P-RFO is that it is applicable to high-dimensional systems since it only uses gradients and no Hessian information. For this thesis it is important to keep in mind that the dimer method is a gradient-based saddle point search algorithm that often performs better than P-RFO when no analytical Hessians are available [43].

2.5 Nudged Elastic Band

The nudged elastic band method (NEB) optimizes a full path connecting two minima through a saddle point [14, 19, 21, 39, 41, 42, 61, 75]. This path is optimized by minimizing the function values (energies) at its discretization points. The path is constructed as a set of points along an initial guess for the path. These points, or *images*, are connected by artificial springs with a certain force constant k.
Let τ_i be the normalized local tangent vector at the image $\mathbf{x}_i$ (tangential to the path). The forces on image $\mathbf{x}_i$ are then the sum of two forces:

$$\mathbf{F}_i = \mathbf{F}_i^{\text{spring}} - \mathbf{F}_i^{\perp} \tag{2.9}$$

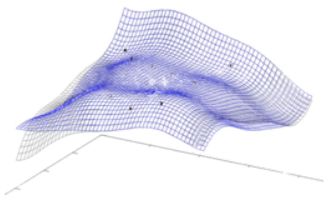
with the spring force

$$\mathbf{F}^{\text{spring}} = k\left(|\mathbf{x}_{i+1} - \mathbf{x}_i| - |\mathbf{x}_i - \mathbf{x}_{i-1}|\right)\tau_i \tag{2.10}$$

to the neighboring images and the part of the negative gradient of the function $E(\mathbf{x})$ that is perpendicular to the spring forces.

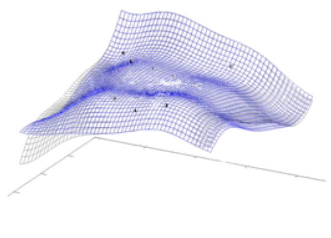
$$\mathbf{F}_i^{\perp} = \nabla E(\mathbf{x}_i) - \nabla E(\mathbf{x}_i)\tau_i \tag{2.11}$$

The forces on the images are then minimized to find the MEP. Often one also spawns a so-called *climbing image* after a few optimization steps. This climbing image is moved in the direction of increasing function value along the path but still minimized in all other directions. It is advantageous when trying to find a first-order saddle point and will probably converge to a point very close or in many cases even on the saddle point. Note that there are other path optimizers as an alternative to NEB [6,28,29,49,63] and many adaptions of the NEB method itself [14, 19, 21, 42, 61]. Important to remember for this thesis is that the NEB method is probably the most commonly used path optimization method which uses artificial spring constants to keep images on the path together and often gives a good estimate for saddle points.



Part II

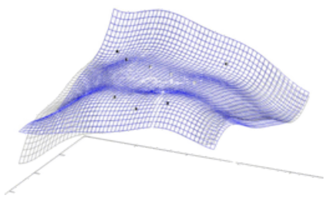
Theory of Gaussian Process Regression



3 Introduction to Gaussian Process Regression

In this part of the thesis the fundamental equations of Gaussian Process Regression (GPR) are explained in detail. This part is partly based on the book by Rasmussen and Williams, [67] mainly sections 4 and 5. In these sections their explanation of GPR is revised and complemented with some explanatory remarks. Additionally, the fundamental equations are thoroughly derived in the cases in which the book was felt to be not readily understandable or not thorough enough. It is avoided to use mathematical technicalities in this part of the thesis, especially the ones concerned with measure theory, if they are not necessary for understanding. In some cases, a more precise formulation is given in the footnotes for the mathematically interested reader.

Originally, GPR was a method for regression in geostatistics, introduced in 1951 by Danie Krige [46]. Nowadays, several views and interpretations of the basic interpolation technique of Gaussian Process Regression exist. Most simply one can interpret the interpolation equations as regularized regression. In this case the regression technique is often called *kernel ridge regression*. The interpolative equations can simply be derived by minimizing the Tikhonov-regularized residual. For a derivation of this, be referred to other sources [20]. But there is a more powerful interpretation of the interpolative equations in the context of *Bayesian statistics*. Also in the context of Bayesian statistics there are two kinds of interpretations to the formalism: The *weight-space view* and the *function-space view*. The explanations in this thesis are mainly focused on the weight-space view in which one derives the most probable weights to be used in the regression formalism. This view on GPR is explained in section 4 where GPR is derived completely



from the concept of linear regression. section 4 is also intended to clarify the idea of GPR as a kernel-based method in which one uses a feature space to represent the data and applies the kernel trick to allow for an infinite dimensional feature space. In section 5 the equations from GPR are derived from another perspective. This time starting from the more abstract definition of a Gaussian process. This also allows to explain GPR in the context of the function-space view in section 5.3 and clarifies the statistical background of GPR further. Note that both views, the weight-space view and the function-space view, will result in the same predictive equations and the same weights.

One advantage of the Bayesian interpretation of the regression formalism is the possibility to give uncertainty estimates which is used for the path optimizer described in section 9. Furthermore, the statistical view on GPR opens the possibility to optimize hyperparameters by the so-called *maximum likelihood* approach. This approach will be discussed only briefly since no benefits were found when using it for the algorithms developed in this thesis. The reasons for that will be discussed in section A.4.

There are many other regression techniques closely related to GPR such as *Reproducing Kernel Hilbert Spaces* (RKHS) [5, 69, 71] with regularization or *Support Vector Machines* (SVM) [23, 71] for regression. Both these methods are very similar to GPR in terms of their predictive equations. For example, one can interpret the GPR formalism in the framework of RKHS [4]. But these formalisms are not equivalent from their mathematical interpretation. And their statistical interpretation is not straightforward [3, 15]. These alternatives are not discussed in this thesis. The GPR framework is the most straightforward to yield all the interpretation possibilities needed for geometry optimization. Before deriving the theory of GPR, some mathematical basics are given.

3.1 Some Basics of Statistics

Before diving into the theory of GPR, one has to establish some nomenclature concerning the Gaussian distribution, also called *normal distribution*, and statistics in general.

A *random variable* is a variable that can take on values that are the outcome of some random phenomenon.[1] If the random phenomenon is the roll of a die, one could define a random variable that can take on the natural numbers from 1 to 6. But there are also continuous random variables: When GPR is used for regressing energies we encounter random variables that take on energy values. Capital letters, e.g. X, are used to refer to random variables and respectively $\mathbf{X}$ for random vectors (collections of random variables). The underlying random phenomenon in GPR is a *Gaussian process*, hence the name Gaussian process regression. This will be explained in section 5.2.

To quantify the possible values that any random variable can take on we define *probability density functions*. These are real-valued functions that describe how probable it is that a random variable takes on a certain value: When integrated over a certain set of possible outcomes it gives the probability of this set to take place. It can only take on non-negative values and it integrates to 1 over the whole space (it is certain that the random variable takes on one of the values of the space).[2] An example of such a probability density function is the Gaussian distribution.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{3.1}$$

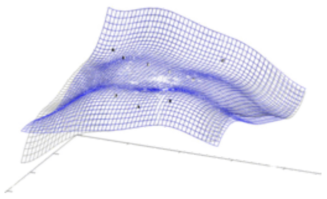
The most probable value of x is given by $\max_x f(x) = \mu$, the *mean* of the Gaussian. In figure 3.1 the mean is at $x = 0$. We also identify the mean with the so-called *expected value* $\mathbb{E}(\cdot)$ of the random variable. For a continuous random variable X that can take on the values x with a probability density $f(x)$, the expected value is defined via an integral.

$$\mathbb{E}(X) = \int_{\mathbb{R}} x f(x) dx \tag{3.2}$$

The term σ^2 of equation (3.1) is referred to as the *variance* of the distribution. The variance describes the expected value of the squared difference between the

[1]Formally, a random variable is a measurable function from a probability space in a measurable space.

[2]Formally, it must also be Lebesgue integrable.



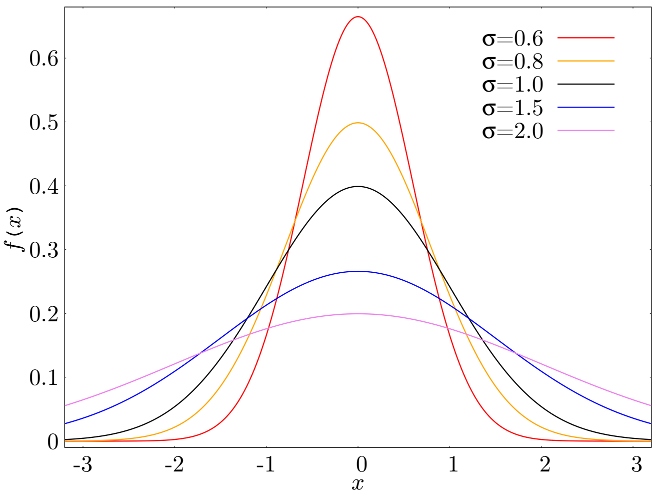


Figure 3.1: The Gaussian/normal distribution for different values of σ.

value of a random variable X and its expected value/mean.

$$\mathrm{Var}\left(X\right) = \mathbb{E}\left[\left(X - \mathbb{E}\left[X\right]\right)^2\right] \tag{3.3}$$

This yields an indication of uncertainty in the prediction. The higher the variance, the flatter the distribution. The flatter the distribution, the smaller the probability density at the mean, i.e. the probability that the random variable takes on the value of the mean is smaller. One can also write

$$X \sim \mathcal{N}(\mu, \sigma^2) \tag{3.4}$$

if a random variable X is distributed normally.

Sometimes a probability of an outcome is dependent on multiple variables which leads to higher dimensionality. In d dimensions one does not only require a single random variable but a random vector $\mathbf{X} = (X_1 \;\; X_2 \;\; \dots \;\; X_d)^T$, a list of d random variables which takes on the value of a vector $\mathbf{x} \in \mathbb{R}^d$. The covariance becomes the so-called *covariance matrix* Σ. A possible common distribution of the d random variables is then described by the *multivariate normal distribution*, the multidimensional form of the normal distribution.

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right) \tag{3.5}$$

One also writes

$$X \sim \mathcal{N}_d(\mu, \Sigma) \tag{3.6}$$

where the entries Σ_{ij} of the covariance matrix Σ are the covariances between the respective random variables.

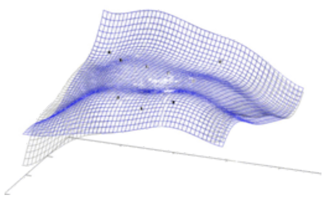
$$\begin{aligned} \Sigma_{ij} = \text{Cov}(X_i, X_j) &= \mathbb{E}\left[(X_i - \mathbb{E}(X_i))\,(X_j - \mathbb{E}(X_j))\right] \\ &= \mathbb{E}\left[(X_i - \mu_i)\,(X_j - \mu_j)\right] \end{aligned} \tag{3.7}$$

The μ_i are the entries of the mean $\mu = (\mu_1 \;\; \mu_2 \;\; ... \;\; \mu_d)^T$.
Note that $\text{Cov}(X_i, X_i) = \text{Var}(X_i)$. A positive covariance $\text{Cov}(X_i, X_j)$ means that high (low) values of X_i indicate mostly high (low) values of X_j or, to be precise, it measures the linear relation of two variables. In a simplified manner one can think of the covariance as a similarity measure of the two variables.

3.2 Bayes' Theorem

The statistical theory of GPR makes use of Bayes' theorem. The idea is to gradually improve a prior belief about the interpolated function with additional training data. Bayes' theorem yields the connection between the prediction of the function without any training data (prior) and the prediction of the function given the training data (posterior). In the following Bayes' theorem is explained.

Let $P(X)$ stand for a probability of the random variable X taking on a certain value. For example for a coin toss the probabilities would be $P(\text{Heads}) = P(\text{Tails}) = 1/2$. For a continuous random variable (for example when it takes on the value of an energy) the probability of a random variable taking on certain values is calculated by integrating the probability density function over the respective values. A probability denoted with $P(X|Y)$ means the probability of X taking on a certain value under the assumption that Y has taken on a certain value. One



often says *probability of* X *given* Y.

$$\text{posterior} = P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{marginal likelihood}} \tag{3.8}$$

H is a so called *hypothesis* whose probability may be affected by additional observations/evidence. In this thesis it might describe the probability of a certain energy. The most probable energy will then be the prediction of the regression model. This is done by finding the hypothesis yielding the highest probability.

The prior probability $P(H)$ is an estimate of the probability of the hypothesis H before the observation data/evidence is given. The posterior $P(H|E)$ is what we want to calculate in the end and gives us the probability of H given certain evidence E. The evidence E is the data from which the system may learn the properties of the real underlying probability of the hypothesis H. In our case the evidence might be computed energies of a chemical system. The factor which introduces the influence of the observed data on the hypothesis is the likelihood $P(E|H)$. The likelihood describes the probability of observing the evidence under the given hypothesis. In other words, it indicates the compatibility of the evidence with the hypothesis. The marginal likelihood $P(E)$ does not influence the process of finding the highest probability since it does not depend on H and can be considered as a normalization constant.[3]

[3]It is the marginalization of the common distribution of evidence and hypothesis over the hypothesis.

4 From the Linear Bayesian Model to Gaussian Process Regression

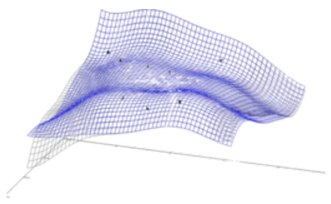
In this section the derivation of all the necessary equations for GPR is shown. The so-called *weight-space* view on GPR is clarified, which gives a probability distribution for the weights in the regression scheme. The weights with the highest probability are then the ones one chooses for predicting the target function (in this thesis that is the energy). This section clarifies why GPR is often viewed as a kernel-based method in which the feature space is used to represent the data and how the kernel trick is applied. The derivation starts with a simple linear model of regression that will be expanded step by step. It is shown how one can include some noise in the data, apply Bayes' theorem, and how one derives the most probable prediction of a function in this framework. Finally, the feature space will be introduced in order to apply the kernel trick.

4.1 Linear Regression with Noise

Let us consider n training points $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i \in R^d$. One further defines the $d \times n$ matrix $X = (\mathbf{x}_1\ \mathbf{x}_2\ \dots\ \mathbf{x}_n)$ and the vector $\mathbf{y} = (y_1\ y_2\ \dots\ y_n)^T \in R^n$. To derive the concept of Gaussian Process Regression one can start from the one-dimensional linear interpolation model.

$$f(x) = \mathbf{x}^T\ \mathbf{w} \tag{4.1}$$

At every training point x_i this yields a prediction $f(x_i)$ of the underlying function that one wants to regress. One does not identify $f(x_i) = y_i$ directly but assumes



a certain noise on the training data.

$$y_i = f(x_i) + \epsilon_i \tag{4.2}$$

The difference between the prediction $f(x_i)$ and the observation y_i, namely the noise ϵ_i is assumed to be given by an independent and identically normally distributed random variable X_ϵ.

$$X_\epsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{4.3}$$

From this assumption of independent noise and $\epsilon_i = y_i - \mathbf{x}^T\mathbf{w}$, following from equations (4.1) and (4.2), one can immediately give the probability density function of the likelihood $p(\mathbf{y}|X, \mathbf{w})$.

$$\begin{aligned} p(\mathbf{y}|X,\mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_\mathrm{n}} \exp\left[-\frac{\left(y_i - \mathbf{x}^T\mathbf{w}\right)^2}{2\sigma_\mathrm{n}^2}\right] \\ &= \frac{1}{\sqrt{2\pi}\sigma_\mathrm{n}} \exp\left[-\frac{\mid \mathbf{y} - X^T\mathbf{w} \mid^2}{2\sigma_\mathrm{n}^2}\right] \end{aligned} \tag{4.4}$$

The prior assumption on the weight vector (which is needed in a Bayesian setting) is simply another multivariate Gaussian distribution $\mathbf{w} \sim \mathcal{N}(0, \Sigma_\mathrm{w})$ with positive definite covariance matrix Σ_w.

$$p(\mathbf{w}) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_\mathrm{w} \mid}} \exp\left[-\frac{1}{2}\mathbf{w}^T\Sigma_\mathrm{w}^{-1}\mathbf{w}\right] \tag{4.5}$$

4.2 Applying Bayes' Theorem

One can now apply Bayes' theorem, equation (3.8), to get the most probable values for the weight vector given the training data. In other words, one searches for the probability density function of the posterior, $p(\mathbf{w}|X, \mathbf{y})$. For that one can ignore all values that are constant in the weights $\mathbf{w}$. Therefore, one drops the marginal likelihood $p(\mathbf{y})$. In the following, terms that are constant in $\mathbf{w}$ are neglected at several steps, indicated by the proportionality sign $\propto$.

$$\begin{aligned}
p(\mathbf{w}|X,\mathbf{y}) &\propto p(\mathbf{y}|X,\mathbf{w})\ p(\mathbf{w}) \\
&\propto \exp\left[-\frac{1}{2\sigma_{\mathrm{n}}^2}\left(\mathbf{y}-X^T\mathbf{w}\right)^T\left(\mathbf{y}-X^T\mathbf{w}\right)\ -\frac{1}{2}\mathbf{w}^T\Sigma_{\mathrm{w}}^{-1}\mathbf{w}\right] \\
&\propto \exp\left[-\frac{1}{2}\left(\frac{1}{\sigma_{\mathrm{n}}^2}\left[-\mathbf{w}^TX\mathbf{y}-\mathbf{y}^TX^T\mathbf{w}+\mathbf{w}^TXX^T\mathbf{w}\right]+\mathbf{w}^T\Sigma_{\mathrm{w}}^{-1}\mathbf{w}\right)\right] \\
&= \exp\left[-\frac{1}{2}\left(\mathbf{w}^T\left(\frac{1}{\sigma_{\mathrm{n}}^2}XX^T+\Sigma_{\mathrm{w}}^{-1}\right)\mathbf{w}-\frac{1}{\sigma_{\mathrm{n}}^2}\mathbf{w}^TX\mathbf{y}-\frac{1}{\sigma_{\mathrm{n}}^2}\left(X\mathbf{y}\right)^T\mathbf{w}\right)\right]
\end{aligned} \tag{4.6}$$

At this point one introduces the substitution $A := \frac{1}{\sigma_{\mathrm{n}}^2}XX^T+\Sigma_{\mathrm{w}}^{-1}$ for which $A = A^T$ and therefore, $A^{-1} = \left(A^T\right)^{-1} = (A^{-1})^T$. Furthermore, one multiplies by $AA^{-1} = \mathbb{1}$ and also by a new term that is not dependent on $\mathbf{w}$ to get

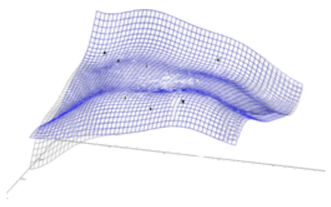
$$\begin{aligned}
p(\mathbf{w}|X,\mathbf{y}) &\propto \exp\left[-\frac{1}{2}\left(\mathbf{w}^TA\mathbf{w}-\mathbf{w}^TA\left(\frac{1}{\sigma_{\mathrm{n}}^2}A^{-1}X\mathbf{y}\right)-\frac{1}{\sigma_{\mathrm{n}}^2}\overbrace{\left(X\mathbf{y}\right)^T\left(A^{-1}\right)^T}^{\left(A^{-1}X\mathbf{y}\right)^T}A\mathbf{w}\right.\right. \\
&\qquad\left.\left.\underbrace{+\frac{1}{\sigma_{\mathrm{n}}^2}\left(A^{-1}X\mathbf{y}\right)^TA\frac{1}{\sigma_{\mathrm{n}}^2}A^{-1}X\mathbf{y}}_{\text{does not depend on }\mathbf{w}}\right)\right] \\
&= \exp\left[-\frac{1}{2}\left(\mathbf{w}^T-\frac{1}{\sigma_{\mathrm{n}}^2}\left(A^{-1}X\mathbf{y}\right)^T\right)A\left(\mathbf{w}-\frac{1}{\sigma_{\mathrm{n}}^2}A^{-1}X\mathbf{y}\right)\right] \\
&= \exp\left[-\frac{1}{2}\left(\mathbf{w}^T-\bar{\mathbf{w}}^T\right)A\left(\mathbf{w}-\bar{\mathbf{w}}\right)\right]
\end{aligned} \tag{4.7}$$

which is a normal distribution with a mean

$$\bar{\mathbf{w}} := \frac{1}{\sigma_{\mathrm{n}}^2}\left(A^{-1}X\mathbf{y}\right) \tag{4.8}$$

and its covariance matrix.

$$A^{-1} = \left(\frac{1}{\sigma_{\mathrm{n}}^2}XX^T+\Sigma_{\mathrm{w}}^{-1}\right)^{-1} \tag{4.9}$$



The predictive equation for the function value $f^* \equiv f^*(\mathbf{x}^*)$ at a point $\mathbf{x}^*$ with the mean $\bar{\mathbf{w}}$ is then

$$f^* = \frac{1}{\sigma_{\mathrm{n}}^2}\mathbf{x}^{*T}A^{-1}X\mathbf{y} \tag{4.10}$$

and one can also calculate the respective variance of this prediction (f^* is now considered to be a random variable that takes on the value of the prediction of the linear model).

$$\begin{aligned}
\mathrm{Var}\left[f^*\right] &= \mathrm{Var}\left[\mathbf{w}^T\mathbf{x}^*\right] \\
&= \mathrm{E}\left[(\mathbf{w}^T\mathbf{x}^* - \mathbf{x}^{*T}\bar{\mathbf{w}})(\mathbf{w}^T\mathbf{x}^* - \mathbf{x}^{*T}\bar{\mathbf{w}})\right] \\
&= \mathrm{E}\left[\mathbf{w}^T\mathbf{x}^*\mathbf{w}^T\mathbf{x}^* - \mathbf{w}^T\mathbf{x}^*\mathbf{x}^{*T}\bar{\mathbf{w}} - \mathbf{x}^{*T}\bar{\mathbf{w}}\mathbf{w}^T\mathbf{x}^* + \mathbf{x}^{*T}\bar{\mathbf{w}}\mathbf{x}^{*T}\bar{\mathbf{w}}\right] \\
&= \mathrm{E}\left[\mathbf{x}^{*T}\mathbf{w}\mathbf{w}^T\mathbf{x}^* - \mathbf{x}^{*T}\mathbf{w}\bar{\mathbf{w}}^T\mathbf{x}^* - \mathbf{x}^{*T}\bar{\mathbf{w}}\mathbf{w}^T\mathbf{x}^* + \mathbf{x}^{*T}\bar{\mathbf{w}}\bar{\mathbf{w}}^T\mathbf{x}^*\right] \\
&= \mathbf{x}^{*T}\mathrm{E}\left[\mathbf{w}\mathbf{w}^T - \mathbf{w}\bar{\mathbf{w}}^T - \bar{\mathbf{w}}\mathbf{w}^T + \bar{\mathbf{w}}\bar{\mathbf{w}}^T\right]\mathbf{x}^* \\
&= \mathbf{x}^{*T}\mathrm{E}\left[(\mathbf{w} - \bar{\mathbf{w}})(\mathbf{w} - \bar{\mathbf{w}})^T\right]\mathbf{x}^* \\
&= \mathbf{x}^{*T}\mathrm{Var}\left[\mathbf{w} - \bar{\mathbf{w}}\right]\mathbf{x}^* \\
&= \mathbf{x}^{*T}A^{-1}\mathbf{x}^*
\end{aligned} \tag{4.11}$$

Since f^* is calculated by a linear combination of normally distributed random variables (linear combination of the weights $\mathbf{w}$) it is also normally distributed.

$$f^* \sim \mathcal{N}\left(\frac{1}{\sigma_{\mathrm{n}}^2}\mathbf{x}^{*T}A^{-1}X\mathbf{y}, \mathbf{x}^{*T}A^{-1}\mathbf{x}^*\right) \tag{4.12}$$

4.3 Introducing the Feature Space

One can represent the training points in the so called feature space rather than with their coordinates $\mathbf{x}_i$. This is done because a representation in this space is usually much more powerful. Many regression tasks cannot be performed linearly of course. But the transformation into a high-dimensional feature space allows non-linear regression. Optimally, the problem becomes very simple (and linear) if expressed in the feature space. This is a common trick in machine learning: The representation of the input data is changed in such a way that the computer can handle the data easier. This is often done with highly nonlinear functions. In modern deep learning approaches to image recognition the representation is changed layer by layer from a simple color information of the pixels to a representation in form of edges, forms, objects, scenes and even meaning of the picture. In a very general way one defines the feature space by some (up to here undefined) function $\phi(\mathbf{x})$ that maps from R^d to R^N. One also defines the matrix $\Phi \equiv \Phi(x)_{ij} := \phi_i(x_j)$ in which ϕ_i represents the element of ϕ in the i-th dimension. One can do the same calculations as above but only substituting $\mathbf{x}$ with $\phi(\mathbf{x})$. From equation (4.12) follows

$$f^* \sim \mathcal{N}(\frac{1}{\sigma_{\mathrm{n}}^2}\phi(\mathbf{x}^*)^T\tilde{A}^{-1}\Phi\mathbf{y}, \phi(\mathbf{x}^*)^T\tilde{A}^{-1}\phi(\mathbf{x}^*)) \tag{4.13}$$

with the newly defined $N \times N$ matrix $\tilde{A} = \frac{1}{\sigma_{\mathrm{n}}^2}\Phi\Phi^T + \Sigma_{\mathrm{w}}^{-1}$. In principle these are already the predictive equations for GPR. In the following this equation is reformulated to make it applicable for a large feature space. However, at this point it becomes clear that GPR can be viewed as a linear regression method with a possibly infinite number of basis functions. Note that it is necessary to invert this $N \times N$ matrix. Therefore, an algorithm based on this formulation scales cubicly in the number of dimensions of the feature space (if one inverts the matrix exactly). Since the feature space is usually chosen to be high dimensional this formulation is impractical. One can change to a different representation of this equation that implies the inversion of an $n \times n$ matrix where n is the number of training points. To do that one separately reformulates the mean and the variance of the above expression. With the definition of $\tilde{A}$ one gets the following equations.

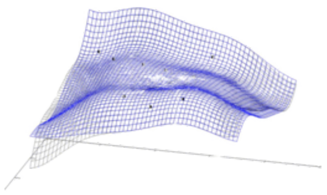
$$\frac{1}{\sigma_{\mathrm{n}}^2}\Phi\left(\Phi^T\Sigma_{\mathrm{w}}\Phi + \sigma_{\mathrm{n}}^2\,\mathbb{1}\right) = \tilde{A}\Sigma_{\mathrm{w}}\Phi \tag{4.14}$$

Multiplying $\tilde{A}^{-1}$ from the left and $\left(\Phi^T\Sigma_{\mathrm{w}}\Phi + \sigma_{\mathrm{n}}^2\mathbb{1}\right)^{-1}$ from the right this yields

$$\tilde{A}^{-1}\frac{1}{\sigma_{\mathrm{n}}^2}\Phi = \Sigma_{\mathrm{w}}\Phi\left(\Phi^T\Sigma_{\mathrm{w}}\Phi + \sigma_{\mathrm{n}}^2\mathbb{1}\right)^{-1} \tag{4.15}$$

and multiplying $\phi(\mathbf{x}^*)^T$ from the left and $\mathbf{y}$ from the right results in

$$\frac{1}{\sigma_{\mathrm{n}}^2}\phi(\mathbf{x}^*)^T\tilde{A}^{-1}\Phi\mathbf{y} = \phi(\mathbf{x}^*)^T\Sigma_{\mathrm{w}}\Phi\left(\Phi^T\Sigma_{\mathrm{w}}\Phi + \sigma_{\mathrm{n}}^2\mathbb{1}\right)^{-1}\mathbf{y} \tag{4.16}$$



which gives the new mean in a different representation. Applying the Woodbury Matrix Identity (Lemma A.1.2) with the substitutions $A = \Sigma_{\mathrm{w}}^{-1}$, $U = V^T = \Phi$ and $C = \sigma_{\mathrm{n}}^2 \mathbb{1}$ one immediately obtains

$$\begin{aligned}&\phi(\mathbf{x}^*)^T \left[\tilde{A}\right]^{-1} \phi(\mathbf{x}^*) = \phi(\mathbf{x}^*)^T \left[\frac{1}{\sigma_{\mathrm{n}}^2}\Phi\Phi^T + \Sigma_{\mathrm{w}}^{-1}\right]^{-1} \phi(\mathbf{x}^*) = \\ &\phi(\mathbf{x}^*)^T \left[\Sigma_{\mathrm{w}} - \Sigma_{\mathrm{w}}\Phi\left(\sigma_{\mathrm{n}}^2\mathbb{1} + \Phi^T\Sigma_{\mathrm{w}}\Phi\right)^{-1}\Phi^T\Sigma_{\mathrm{w}}\right]\phi(\mathbf{x}^*)\end{aligned} \tag{4.17}$$

Substituting $K := \Phi^T\Sigma_{\mathrm{w}}\Phi$ one can formulate the probability distribution as follows.

$$\begin{aligned}f^* \mid \mathbf{x}^*, X, \mathbf{y} \sim \mathcal{N}\Big(&\phi(\mathbf{x}^*)^T\Sigma_{\mathrm{w}}\Phi\left(K + \sigma_{\mathrm{n}}^2\mathbb{1}\right)^{-1}\mathbf{y}, \\ &\phi(\mathbf{x}^*)^T\Sigma_{\mathrm{w}}\phi(\mathbf{x}^*) - \phi(\mathbf{x}^*)^T\Sigma_{\mathrm{w}}\Phi\left(\sigma_{\mathrm{n}}^2\mathbb{1} + K\right)^{-1}\Phi^T\Sigma_{\mathrm{w}}\phi(\mathbf{x}^*)\Big)\end{aligned} \tag{4.18}$$

This formulation is more favorable when there are fewer training points than number of dimensions in the feature space. The larger the feature space the larger the function space one can describe with it. Therefore, one usually defines an infinitely large feature space and I continue with the formulation of equation (4.18).

4.4 The Kernel Trick

There is a very common way to simplify equation (4.18) recognizing that all the terms involving x^* or the training points X (implicitly in $\Phi \equiv \Phi(x)_{ij} = \phi_i(x_j)$) are of one of the following forms: $\phi(\mathbf{x}^*)\Sigma_{\mathrm{w}}\Phi$, $\phi(\mathbf{x}^*)^T\Sigma_{\mathrm{w}}\phi(\mathbf{x}^*)$ or $K = \Phi^T\Sigma_{\mathrm{w}}\Phi$. All these forms completely consist of expressions of a particular form.

$$k(x, x') := \phi(\mathbf{x})^T\Sigma_{\mathrm{w}}\phi(\mathbf{x}') \tag{4.19}$$

Since Σ_{w} is assumed to be positive definite it can be decomposed as $\Sigma_{\mathrm{w}} = LL^T$ and with the definition of $\psi = L^T\phi$ one can write k as a simple inner product.

$$k(x, x') = \psi(\mathbf{x})^T\psi(\mathbf{x}') \tag{4.20}$$

This function $k(x, x')$ is called the *kernel* or *covariance function*. It is not necessary to calculate the scalar product of the functions ψ explicitly if one simply defines the covariance function as a function of x and x'. Therefore, the complete procedure of transferring the data points into the feature space and building the scalar product of the functions that define that space is reduced to a simple function evaluation: The evaluation of the kernel. This procedure is referred to as the *kernel trick* and is often used when dealing with a feature space (especially when it is high dimensional). Note that the covariance function (kernel) as the result of an inner product must be a positive definite, symmetric bilinear form. One can define (similar to the matrix $K \equiv K_{ij} = k(x_i, x_j)$) the vector $\mathbf{k}^* := \Phi^T \Sigma_\mathrm{w} \phi(\mathbf{x}^*)$ with its elements $k_i^* = \phi(\mathbf{x_i})^T \Sigma_\mathrm{w} \phi(\mathbf{x}^*) = k(x_i, x^*)$. Furthermore, saying that $\mathbf{w}$ is the solution to

$$(K + \sigma_\mathrm{n}^2 \mathbb{1})\mathbf{w} = \mathbf{y} \tag{4.21}$$

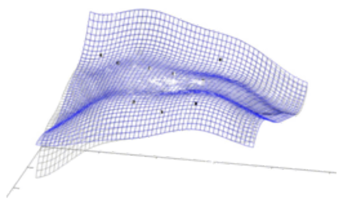
and defining $\mathbf{c}$ as the solution to

$$\left(\sigma_\mathrm{n}^2 \mathbb{1} + K\right) \mathbf{c} := \Phi^T \Sigma_\mathrm{w} \phi(\mathbf{x}^*) = \mathbf{k}^* \tag{4.22}$$

one gets

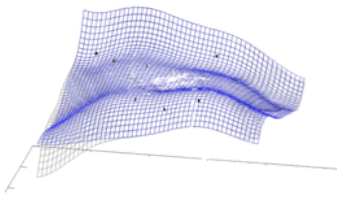
$$f^* \mid \mathbf{x}^*, X, \mathbf{y} \sim \mathcal{N}\Big(\mathbf{k}^* \mathbf{w}, k(x^*, x^*)^T - \mathbf{k}^* \mathbf{c}\Big) \tag{4.23}$$

as the predictive equations. One can calculate the mean and the variance of this distribution by inverting the matrix $(K + \sigma_\mathrm{n}^2 \mathbb{1})$, i.e. solving equations (4.21) and (4.22). Given the covariance function, the most probable guess for the function value at x^* is given by the mean of the distribution (4.23). The variance of the distribution can be viewed as an uncertainty measure for the prediction. These are the final predictive equations for Gaussian process regression.



5 Gaussian Process Regression from a Statistical Perspective

In this section another approach to derive GPR will be demonstrated. One can consider GPR as a statistical methodology in the context of so-called *statistical inference*. In statistical inference one assumes that observed data is sampled from an underlying probability distribution. One tries to describe this distribution and ultimately use the properties of this distribution in order to infer information from it. Bayesian inference is a method of statistical inference that is based on the application of Bayes' theorem to include information about observations to a prior belief about the underlying distribution. When additional data, i.e. more observations, are provided the underlying probability distribution is updated with the additional data. This chapter is intended to show the statistical perspective on GPR deriving it from the definition of a Gaussian process. This also allows the interpretation of GPR in the function-space view.



5.1 Defining a Gaussian Process

Definition 5.1.1. *"A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution." [67]*

The underlying probability distribution for GPR is described by a certain *stochastic process* that is called a *Gaussian process* (GP). A stochastic process is a collection of random variables $f(x)$, $x \in X$ with an index set X that can also be $\mathbb{R}^d$.[1] Giving the joint probability distribution for every finite subset of random variables $Y = \{f(x_1), ..., f(x_k)\}$ to any index set $X = \{x_1, ..., x_k\}$ specifies the kind of the stochastic process. For example a Gaussian process (GP) is a special stochastic process for which the distributions over a finite subset is given by a joint multivariate Gaussian distribution. The Gaussian process gives rise to the name of GPR.

A Gaussian process that describes a real process $f(x)$ can be specified by only two features:

- the mean function
$$m(x) = \mathbb{E}[f(x)] \tag{5.1}$$
which is the expectation value of $f(x)$ and

- the covariance function
$$k(x, x_n) = \mathbb{E}\Big[\big(f(x) - m(x)\big)\big(f(x_n) - m(x_n)\big)\Big] \tag{5.2}$$
which is a measure of the correlated change of the values of f at the two locations x and x_n. The covariance function includes an assumption about the interpolated function. For example, a certain continuity and differentiability.

[1] If the index set is multi-dimensional, for example $\mathbb{R}^d$, one usually refers to the stochastic object as a *random field*, or a *Gaussian random field* when the joint probability distribution for a finite subset is Gaussian. Also the expression *spatial random field* is commonly used since $\mathbb{R}^d$ can refer to a physical space rather than a time. Note that stochastic processes (and GPs) are usually defined with an index set referring to the one-dimensional time axis. So with this notion, one can call a GP a Gaussian random field in one dimension. Nevertheless, in this work a Gaussian random field is simply referred to as a GP since it is common in the theoretical chemistry literature (and in other fields as well) to not make this distinction.

5.2 Gaussian Processeses as a Method of Bayesian Inference

In this section, some additional understanding of the statistical properties of GPR is established. It is explained how it can be understood as a method of Bayesian inference.

It is shown that by assuming a common Gaussian distribution of a finite number of random variables (definition of a GP), conditioning on the training data, and then assuming noise on the training data one arrives at the same predictive equations as in the derivation of GPR from the linear model in section 4.

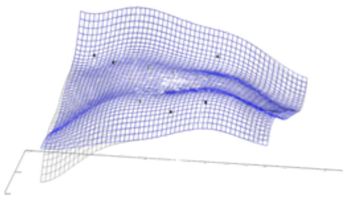
In the present work, a GP is used for regression of an energy surface. That means that a scalar function is predicted, given some training data: For that one has to determine the distribution

$$p(y_*|\mathbb{T}, \mathbf{x}_*) \tag{5.3}$$

of the scalar output y_*, the prediction of the energy at position $\mathbf{x}_*$ given the set of N training points $\mathbb{T} = (\mathbf{x}_i, y_i|i = 1, ..., N)$ which represent the configurations of the chemical system. Note that one does not have to distinguish between training and test points in the definition of a GP. All points are considered to be part of the same distribution: One introduces $N + 1$ random variables $Y_1, ..., Y_N, Y_*$ to model the underlying function's values at the respective inputs $\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_*$. The only thing that is modeled explicitly are the covariances between outputs. In agreement with the definition of a GP one assumes a multivariate Gaussian distribution on these variables.

$$p(y_1, ..., y_N, y_*|\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_*) \propto \exp\left(-\frac{1}{2}\tilde{\mathbf{y}}^T\Sigma^{-1}\tilde{\mathbf{y}}\right) \tag{5.4}$$

The column vector $\tilde{\mathbf{y}}$ contains all outputs $\tilde{\mathbf{y}} = (y_1, ..., y_N, y_*)$ and the matrix Σ is defined as the covariance matrix between the inputs $\Sigma_{mn} = k(\mathbf{x}_m, \mathbf{x}_n)$ for all $m, n = 1, ..., N + 1$. This distribution is considered as a *prior* since it has not included any information about the values of the training data y_i. In other words, no information about the function one wants to approximate is included. In most



cases it is sensible to chose priors that specify strong covariance of random variables that are close to each other, meaning that their respective coordinate vectors have small Euclidean distances. Note that the covariance function has not been specified yet. In the following, the matrix Σ will be rewritten in a manner that one can clearly distinguish between the values corresponding to training data $y_1, ..., y_N$ and the testing value y_*.

$$\Sigma = \begin{bmatrix} K & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \tag{5.5}$$

The matrix K refers to the covariance matrix of the training data with itself while $\mathbf{a}$ is the N dimensional vector of covariances of the test data (at the point $\mathbf{x}_*$) and the training data. Consequently, the scalar b is the covariance of the testing data with itself. In the following subsection it is shown that conditioning on the training data and the subsequent inclusion of noise yields the predictive equations of GPR.

5.2.1 Conditioning on Noise-Free Observations

In the end the probability density of the test value y_* given the training data $(\mathbf{x}_i, y_i), i = 1, ..., N$ has to be derived. Therefore, one must condition the above distribution (5.4) on the training values, i.e. one builds $p(y_*|y_1, ..., y_N, \mathbf{x}_1, ..., \mathbf{x}_*)$. For this purpose, one splits the vector $\tilde{\mathbf{y}}$ of the complete data set in the components one wants to condition on (putting the training data in a new N dimensional vector $\mathbf{y}$) and the test data y_*.

$$\tilde{\mathbf{y}} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \\ y_* \end{bmatrix} =: \begin{bmatrix} \mathbf{y} \\ \\ y_* \end{bmatrix} \tag{5.6}$$

Then one uses Lemma (A.1.1) on the Σ matrix from equation (5.5).

$$\Sigma^{-1} = \begin{bmatrix} K & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix}^{-1} = \begin{bmatrix} \mathbb{1} & -K^{-1}\mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} K^{-1} & \mathbf{0} \\ \mathbf{0}^T & \left(b - \mathbf{a}^T K^{-1}\mathbf{a}\right)^{-1} \end{bmatrix} \begin{bmatrix} \mathbb{1} & \mathbf{0} \\ -\mathbf{a}^T K^{-1} & 1 \end{bmatrix} \tag{5.7}$$

The exponent of equation (5.4) becomes

$$\begin{aligned}
&-\frac{1}{2}\tilde{\mathbf{y}}^T\left[\Sigma^{-1}\right]\tilde{\mathbf{y}} \\
&= -\frac{1}{2}\begin{bmatrix}\mathbf{y}\\ y_*\end{bmatrix}^T\left[\Sigma^{-1}\right]\begin{bmatrix}\mathbf{y}\\ y_*\end{bmatrix} \\
&= -\frac{1}{2}\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}^TK^{-1}\mathbf{a}+y_*\end{bmatrix}^T\begin{bmatrix}K^{-1} & \mathbf{0}\\ \mathbf{0}^T & \left(b-\mathbf{a}^TK^{-1}\mathbf{a}\right)^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}^TK^{-1}\mathbf{a}+y_*\end{bmatrix} \\
&= \mathbf{y}^TK^{-1}\mathbf{y}+\left(y_*-\mathbf{y}^TK^{-1}\mathbf{a}\right)\left(b-\mathbf{a}^TK^{-1}\mathbf{a}\right)^{-1}\left(y_*-\mathbf{y}^TK^{-1}\mathbf{a}\right)
\end{aligned} \tag{5.8}$$

which yields

$$y_* \sim \mathcal{N}(\mathbf{y}^TK^{-1}\mathbf{a},\, b-\mathbf{a}^TK^{-1}\mathbf{a})\ . \tag{5.9}$$

The query value y_* has a Gaussian distribution. The prior of equation (5.4) has been restricted to fit the training data. This will become more clear in section 5.3.

5.2.2 Inclusion of Noise

Unfortunately, the observed values are normally not noise-free. Assuming an independent, Gaussian noise ϵ on every observed value with mean 0 and variance σ_n

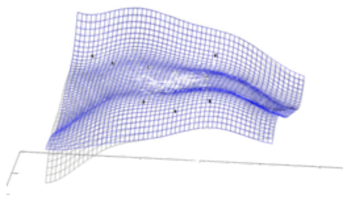
$$\epsilon \sim \mathcal{N}(0,\, \sigma_n^2) \tag{5.10}$$

one gets a different covariance between the observation values

$$\mathrm{Cov}(y_p, y_q) = \mathrm{Cov}(f(\mathbf{x}_p)+\epsilon_p, f(\mathbf{x}_a)+\epsilon_q) \tag{5.11}$$

in which $f(\mathbf{x}_i), i = 1, ..., N$ are the exact values of the underlying function and y_i the distorted observations. The value ϵ_i is the noise on observation y_i. Using the linearity of the covariance one gets

$$\mathrm{Cov}(y_p, y_q) = \underbrace{\mathrm{Cov}(f(\mathbf{x}_p), f(\mathbf{x}_q))}_{k(\mathbf{x}_p,\mathbf{x}_q)} + \underbrace{\mathrm{Cov}(f(\mathbf{x}_p), \epsilon_q)}_{=0} + \underbrace{\mathrm{Cov}(\epsilon_p, f(\mathbf{x}_q))}_{=0} + \underbrace{\mathrm{Cov}(\epsilon_p, \epsilon_q)}_{=\delta_{pq}\sigma_n^2} \tag{5.12}$$



because of the independence of the noise terms ϵ_i. Now one only has to substitute the old covariance matrix K with the new one $\tilde{K}$ which includes the noise.

$$\tilde{K}_{mn} = K_{mn} + \delta_{mn}\sigma_n^2 \tag{5.13}$$

Using this substitution all equations from above apply for the case of noisy observations (interpreting $\mathbf{y}$ as the observed, noisy values) especially equation (5.9). The most probable y_* according to equation (5.9) is the mean $\bar{y}_*$ of the distribution. Including noise in the training data this yields

$$\bar{y}_*(\mathbf{x}_*) = \mathbf{y}^T \tilde{K}^{-1}\mathbf{a} = \sum_{n=1}^{N} w_n k(\mathbf{x}_*, x_n) \tag{5.14}$$

with *weights*

$$w_n = \sum_{m=1}^{N} y_m \tilde{K}_{mn}^{-1} \tag{5.15}$$

and are (using the symmetry of K) the solution of the linear system

$$\tilde{K}\mathbf{w} = \mathbf{y}\ . \tag{5.16}$$

This defines the predictive equations of GPR. Inverting the matrix $\tilde{K}$ explicitly or solving the above linear system yields weights for the prediction equation (5.14). Note that from an algorithmic point of view the noise values σ_n^2 that are included in $\tilde{K}$ can be viewed as regularization parameters that make the inversion/solution more stable by enforcing a more diagonally dominant $\tilde{K}$. It is also interesting to notice that one can also give an estimate of the uncertainty of the prediction using the covariance of the distribution of y_*. Since it is

$$\mathrm{Var}(X) = \mathrm{Cov}(X, X) \tag{5.17}$$

the variance of the observation is (using equation (5.9) with the new covariance matrix $\tilde{K}$)

$$\begin{aligned} \mathrm{Var}(y_*) &= \mathrm{Cov}(y_*, y_*) \\ &= b - \mathbf{a}^T \tilde{K}^{-1} \mathbf{a} \\ &= k(\mathbf{x}_*, \mathbf{x}_*) - \sum_{n=1}^{N} k(\mathbf{x}_n, \mathbf{x}_*) c_n \end{aligned} \tag{5.18}$$

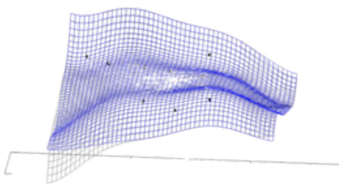
with

$$c_n := \sum_{m=1}^{N} \tilde{K}^{-1}_{nm} a_m \tag{5.19}$$

so that $\mathbf{c} = (c_1, ..., c_N)$ is the solution to the following linear system.

$$\tilde{K}\mathbf{c} = \mathbf{a} \tag{5.20}$$

Note that one assumes only noise on the training data, not on the predicted value y_*. Again, by solving an $N \times N$ linear system one gets the variance of the prediction for the test data. This can be an important uncertainty estimate since $\mathrm{Var}(y_*) = E\left((y_* - E\left(y_*\right))^2\right)$. The higher the variance, the higher the uncertainty in the prediction of the energy. This can give an indication where more training points might be needed in order to decrease uncertainties in the regression scheme.



5.3 GPR in the Function Space

In the following it is clarified why GPR can also be understood as a method that chooses functions from a prior-defined function space and restricts the function-space to only these functions which go through the training data. This section also tries to clarify the statistical view on GPR using as few mathematical requirements as possible. Mathematically more precise notions and references are given in the footnotes.

Stochastic processes have an underlying probability space. Loosely speaking a probability space consists of the possible outcomes of the process/experiment with a respective probability measure that assigns each possible outcome a probability.[2] The possible outcomes of this probability space can be understood as functions. In this thesis these functions $f(\mathbf{x})$ represent possible potential energy surfaces, i.e. functions from $\mathbb{R}^d$ to $\mathbb{R}$.[3] Therefore, a stochastic process can be seen as a way of statistically weighting a function space. And the most probable function is the prediction of the process when it is used for regression.

The probability distribution of these functions is not necessarily directly defined. Stochastic processes are usually defined by giving the joint probability distribution over a finite set of random variables that can take on the function's values at discrete points.[4] For a Gaussian process this distribution is given by a multivariate normal distribution. A multivariate normal distribution is completely specified by giving the mean values (most probable values/energies at the *considered points*) and the covariance matrix (these are the covariances between the *considered points*). Note that the *considered points* can be training points from which the energy values are known, but also testing points at which one wants to infer the energy. The evaluation of the GP at a test point is done by assuming a

[2]Formally a probability space consists of the sample space Ω, a σ-algebra $\mathbb{F}$ on Ω and the respective probability measure P.

[3]Formally these are functions from the given index set T (in this thesis this is the set of coordinates) to a measurable space Z with a given measure $\mathbb{Z}$. The space Z is called the *state space* and its elements are the values that the stochastic process can take. In this thesis these values are values of energy.

[4]The existence of the stochastic process is guaranteed by Kolmogorov's existence theorem. More details can be found in the literature [16].

common distribution of all considered points and then conditioning over the training points and marginalizing over the other test points.[5] In the prior one does not have any training data. The most probable function is defined by the mean of the prior. The statistical weighting of the functions in the underlying function space is given by the covariance function of the prior. One can see examples of such functions in figure 5.1 on the left. In the posterior, the conditioning of the distribution over the training data defines a function space that is restricted to fulfill the given training data, see figure 5.1 on the right. In the rest of this section it is explained how the sampling of the functions in figure 5.1 was done. This will further clarify how the function space in a stochastic process can be understood.

The prior is completely defined by the covariance function $k(\cdot,\cdot)$ and the mean of the prior. The mean of the prior is called $\bar{y}_{\text{pr}}$. It can also be a vector, but in this thesis only scalar functions are considered ($\bar{y}_{\text{pr}} \in \mathbb{R}$). The prior mean is a guess of how the underlying function one wants to regress might look like. Let $\mathbb{X}_*$ be a set of coordinates at which one wants to evaluate the prior. Let $\mathbf{Y}_*$ be a random vector containing the random variables at the test points in the set $\mathbb{X}_*$. The prior is given by the following probability distribution.

$$\mathbf{Y}_*|\mathbb{X}_* \sim \mathcal{N}\left(\bar{y}_{\text{pr}}, K(\mathbb{X}_*, \mathbb{X}_*)\right) \tag{5.21}$$

Consider the set of test data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_{N_{\text{test}}}, y_{N_{\text{test}}})$ with unknown y_i and $\mathbf{x}_i \in \mathbb{X}_*$. To plot functions of the prior one wants to evaluate the prior's possible function values y_i at the argument values $\mathbf{x}_i$. The covariance matrix $K(\mathbb{X}_*, \mathbb{X}_*)$ is of dimension $N_{\text{test}} \times N_{\text{test}}$. Its elements are the covariances of the random variables Y_i (contained in the vector $\mathbf{Y}$), namely $K_{ij}(\mathbb{X}_*, \mathbb{X}_*) = \text{Cov}(Y_i, Y_j)$. The Y_i are random variables that can take on the values of the energies, $\mathbf{y}_i$, at the test points. Their covariances are calculated by the covariance function, i.e.

[5] Considering two random variables A and B that have a known joint distribution, e.g. a normal distribution, the marginalization of A over B means that the value of A is evaluated by averaging over all possible outcomes of B. For a Gaussian distribution the marginalization is particularly easy. If the common distribution is defined by $(A, B) \sim \mathcal{N}(\mathbf{m}, \Sigma)$, the marginalization of this distribution over B is given by $A \sim \mathcal{N}(m_1, \Sigma_{11})$ where Σ_{11} is the respective upper-left submatrix of Σ and m_1 the respective component of $\mathbf{m}$. This property is also called the *marginalization property* and simply means that the examination of a larger set of test data does not change the distribution of the smaller set.

$K_{ij}(\mathbb{X}_*, \mathbb{X}_*) = k(\mathbf{x}_i, \mathbf{x}_j)$. The covariance function between $\mathbf{x}_i$ and $\mathbf{x}_j$ measures the covariance of the respective random variables Y_i and Y_j of the stochastic process. Note that the prior does not contain training data. The distribution only contains points at which one wants to evaluate the prior.

The prior defines a function space in which every function is assigned a probability. The most probable function is the mean of the prior, $\bar{y}_{\text{pr}}$. To get an impression of how the other functions might look like one can sample some functions from the function space of the prior (and of the posterior). To sample functions from such a function space one has to find a function that is different from $\bar{y}_{\text{pr}}$ but still satisfies the conditions of the prior. That means that the function still has an expectation value of $\bar{y}_{\text{pr}}$ and a covariance matrix given by $K(\mathbb{X}_*, \mathbb{X}_*)$ for all test points. To do that (for a Gaussian process) one defines a N_{test}-dimensional random vector $\mathbf{U}$ in which all random variables are normally distributed $\mathbf{U} \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$. Let the covariance matrix $K \equiv K(\mathbb{X}_*, \mathbb{X}_*)$ be decomposed by the Cholesky decomposition $K = LL^T$ and let $\bar{\mathbf{y}}_{\text{pr}} = (\bar{y}_1, \bar{y}_2, ..., \bar{y}_{N_{\text{test}}})$ be the vector containing all evaluations of the prior mean at the test points. Furthermore, let $\mathbf{u}$ be a possible outcome of the random vector $\mathbf{U}$. Then one can define a new function[6] by $\mathbf{y}_{\mathbf{U}} = \bar{\mathbf{y}}_{\text{pr}} + L\mathbf{u}$ which is different from $\bar{\mathbf{y}}_{\text{pr}}$ by a random vector. This function is contained in the function space defined by the prior because it still fulfills the conditions of the prior at the test points: The expectation value (mean) is still $\bar{\mathbf{y}}_{\text{pr}}$ since the expectation value of $\mathbf{U}$ is $\mathbf{0}$. And the covariance matrix is still K.

$$\begin{aligned}
&\mathrm{Cov}[\bar{\mathbf{y}}_{\text{pr}} + L\mathbf{U}, \bar{\mathbf{y}}_{\text{pr}} + L\mathbf{U}] \\
&\qquad = \mathbb{E}[(\bar{\mathbf{y}}_{\text{pr}} + L\mathbf{U} - \overbrace{\mathbb{E}[\bar{\mathbf{y}}_{\text{pr}} + L\mathbf{U}]}^{\bar{\mathbf{y}}_{\text{pr}}})(\bar{\mathbf{y}}_{\text{pr}} + L\mathbf{U} - \overbrace{\mathbb{E}[\bar{\mathbf{y}}_{\text{pr}} + L\mathbf{U}]}^{\bar{\mathbf{y}}_{\text{pr}}})^T] \\
&\qquad = \mathbb{E}[(L\mathbf{U})(L\mathbf{U})^T] = \mathbb{E}[L\mathbf{U}\mathbf{U}^T L^T] \\
&\qquad = L\mathbb{E}[\mathbf{U}\mathbf{U}^T]L^T = L\mathbb{1}L^T = K
\end{aligned} \tag{5.22}$$

[6] vector containing the set of evaluations of this function at the training points

With this method one can sample a random function from the prior. One can do the same with the posterior distribution, compare equation (5.14). Given some training points $\mathbb{X}_{\mathrm{tr}}$ with their respective function values contained in the vector $\mathbf{y}_{\mathrm{tr}}$ one obtains the following posterior distribution.

$$\begin{aligned}\mathbf{Y}_*|\mathbb{X}_*, \mathbf{y}_{\mathrm{tr}}, \mathbb{X}_{\mathrm{tr}} \sim \mathcal{N}\big(&K(\mathbb{X}_*, \mathbb{X}_{\mathrm{tr}})K(\mathbb{X}_{\mathrm{tr}}, \mathbb{X}_{\mathrm{tr}})^{-1}\mathbf{y}_{\mathrm{tr}}, \\ &K(\mathbb{X}_*, \mathbb{X}_*) - K(\mathbb{X}_*, \mathbb{X}_{\mathrm{tr}})K(\mathbb{X}_{\mathrm{tr}}, \mathbb{X}_{\mathrm{tr}})^{-1}K(\mathbb{X}_{\mathrm{tr}}, \mathbb{X}_*)\big)\end{aligned} \tag{5.23}$$

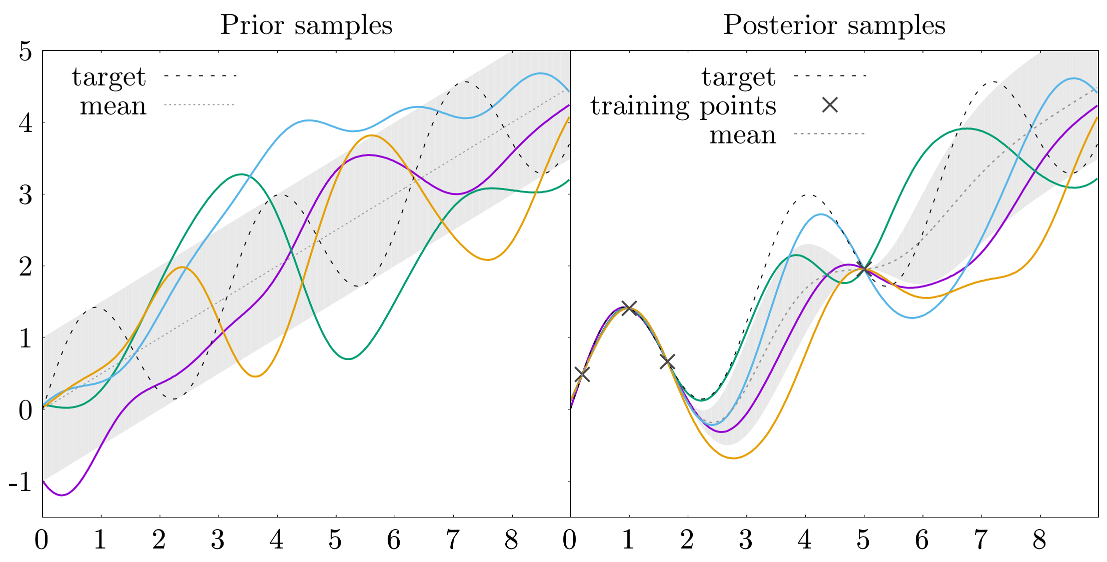
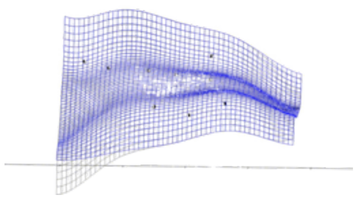


Figure 5.1: Sampled functions from the prior and the posterior to approximate the target function $f(x) = \frac{x}{2} + \sin(2x)$. The prior mean is set to $m(x) = \frac{x}{2}$. The grey area depicts the mean $+/-$ the variance of the prediction. For this example this area corresponds to a 68% confidence region. This means that 68% of all function values considering all functions in the prior's function space are inside this area. The black crosses are the training points that are included in the posterior. In the posterior the function space of the prior is restricted to the functions that go through the training points.



5.4 The Chosen Covariance Functions

The covariance function is the central element of GPR since it defines the prior function space (together with the mean of the prior). That means that it defines the function space used for regression. Therefore, it also defines the properties of the resulting surrogate model for the PES one wants to regress. In theoretical chemistry one usually assumes the PES to be smooth, i.e. it has continuous derivatives of all orders which are of physical interest. In this thesis only second order information of the regressed PES is needed. Therefore, the regression of the PES only has to be two times continuously differentiable with respect to the atomic coordinates. Two commonly used covariance functions that result in a GP like that were implemented in this work, the squared exponential covariance function and a special case of the Matérn covariance function. To get a good understanding of covariance functions it is often helpful to look at their *spectral density*. The spectral density can be understood as the measure of how much certain frequencies contribute to the description of the GP by the covariance function. *Bochner's theorem* [67,77] yields a description for the covariance function in terms of its spectral density. It states that a complex-valued function k on $\mathbb{R}^d$ is the covariance function of a random process[7] on $\mathbb{R}^d$ if and only if it can be represented as

$$k(\mathbf{r}) = \int_{\mathbb{R}^d} \mathrm{e}^{2\pi i \mathbf{s}\cdot\mathbf{r}} d\mu(\mathbf{s}) \tag{5.24}$$

where μ is a positive finite measure. If μ has a density $S(\mathbf{s})$,[8] S is called the spectral density corresponding to k and one can interpret the spectral density as

[7]More mathematically rigorous: k is the covariance function of a weakly stationary mean square continuous complex-valued random process. *Weakly stationary* means that the mean of the process is constant and that the covariance function $k(\mathbf{x}_m, \mathbf{x}_n)$ is only a function of $\mathbf{r} = \mathbf{x}_m - \mathbf{x}_n$, i.e. one can write $k(\mathbf{r})$. *Mean square continuous* means that the process X is *continuous in the mean square sense* at all points $\mathbf{x}$, i.e. that $\lim_{h\to 0} \mathbb{E}X(\mathbf{x}+\mathbf{h}) - X(\mathbf{x})^2 = 0$ [22].

[8]With the Radon–Nikodym theorem the existence of a density of a measure μ with respect to another measure ν (in this case this is the Lebesgue measure) can be shown: The requirements for its existence is that the measure ν is absolutely continuous with respect to μ, i.e. that $\mu(A) = 0$ implies $\nu(A) = 0$ for every measureable set A [30]. This is the case for all the covariance functions presented.

a result of a Fourier transformation.

$$k(\mathbf{r}) = \int_{\mathbb{R}^d} \mathrm{e}^{2\pi i \mathbf{s}\cdot\mathbf{r}} S(\mathbf{s}) d\mathbf{s} \tag{5.25}$$

$$S(\mathbf{s}) = \int k(\mathbf{r}) \mathrm{e}^{-2\pi i \mathbf{s}\cdot\mathbf{r}} d\mathbf{r} \tag{5.26}$$

That can give some intuition for the behavior of the covariance function. If higher frequencies are present, the covariance function allows to sample fast changes of the training data and the resulting GP becomes less smooth. In the following, two of the most common covariance functions are introduced, the squared exponential and the Matérn covariance function. Subsequently, a few hints are given why the Matérn covariance function is preferable for interpolating energies in theoretical chemistry. Note that all covariance functions $k(\mathbf{x}_m, \mathbf{x}_n)$ presented are only functions of the Euclidean norm $r := |\mathbf{x}_m - \mathbf{x}_n|$ of the distance between the two vectors. One calls these kind of covariance functions *isotropic*.

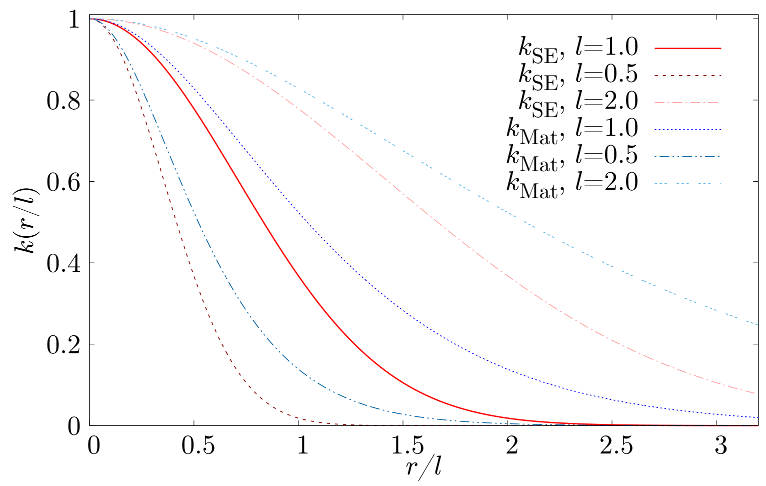
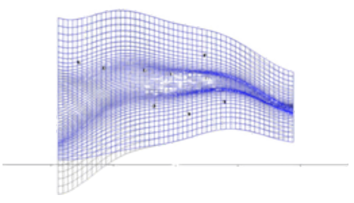


Figure 5.2: Comparing the squared exponential covariance function and the Matérn covariance function (with $\nu = 5/2$) for different length scales l.



5.4.1 The Squared Exponential Covariance Function

The most commonly used covariance function in the machine learning community is the squared exponential covariance function.

$$k_{\mathrm{SE}}(r) = \sigma_{\mathrm{f}}^2 \exp\left(-\frac{r^2}{2l^2}\right) \tag{5.27}$$

The parameter σ_{f} is just a scaling factor and will be set to $\sigma_{\mathrm{f}} = 1$ for the rest of this thesis. The parameter l determines a characteristic length scale parameter that determines how strongly the covariance function decreases with distance r. It describes the range on which the target function might show relevant information. Looking at the curves in red in figure 5.2 one sees how l influences the squared exponential covariance function. The spectral density of the squared exponential covariance function is

$$S_{SE}(s) = \left(2\pi l^2\right)^{d/2} \mathrm{e}^{-2\pi^2 l^2 s^2} \tag{5.28}$$

in which d stands for the dimensionality of the system.

5.4.2 The Matérn Covariance Functions

The Matérn covariance functions [52] are a class of functions with a parameter ν with which one can determine the smoothness of the resulting Gaussian process. If $\nu > k$ for $k \in \mathbb{N}$, the process is k-times differentiable in the mean square sense. The class of the Matérn covariance functions looks like this.

$$k_{\mathrm{Mat}_\nu}(r) = \sigma_{\mathrm{f}}^2 \frac{2^{(1-\nu)}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^\nu K_\nu\left(\frac{\sqrt{2\nu}r}{l}\right) \tag{5.29}$$

The parameter σ_{f} is again set to $\sigma_{\mathrm{f}} = 1$ in the rest of this thesis. The function K_ν is the modified Bessel function of the second kind, see Ref. 1 and Γ is the gamma function. The parameter l is (like for the squared exponential covariance function) a length scale parameter. For an example with $\nu = 5/2$, see figure 5.2. Note, however, that the length scale has a different influence on the behavior of the covariance function than for the squared exponential covariance function. Therefore, it also has a different influence on the resulting process: A less smooth function can have a higher length scale and still cope with faster changes in the target function. The practically most useful covariance functions of the Matérn class are the ones with $\nu = n + 1/2$, $n \in \mathbb{N}$. In this case the functions have a much easier form.

$$k_{\mathrm{Mat}_{1/2}}(r) = \sigma_{\mathrm{f}}^2 \exp\left(-\frac{r}{l}\right) \tag{5.30}$$

$$k_{\mathrm{Mat}_{3/2}}(r) = \sigma_{\mathrm{f}}^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(-\frac{\sqrt{3}r}{l}\right) \tag{5.31}$$

$$k_{\mathrm{M}}(r) = k_{\mathrm{Mat}_{5/2}}(r) = \sigma_{\mathrm{f}}^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left[-\frac{\sqrt{5}r}{l}\right] \tag{5.32}$$

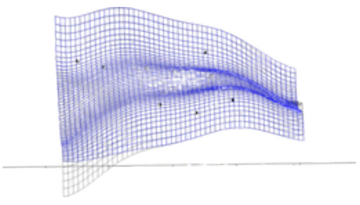
Higher values of ν yield results that are very similar to the results of the squared exponential covariance functions. Therefore, they are usually neither considered in the literature nor in the present work. For simplicity the function $k_{\mathrm{Mat}_{5/2}}$ is abbreviated with $k_{\mathrm{M}}(r)$ since this is the covariance function that will be predominantly used in this thesis. The spectral density of the Matérn class is given by the following expression

$$S_{\mathrm{Mat}_\nu}(s) = \frac{2^d \pi^{d/2} \Gamma(\nu + d/2)(2\nu)^\nu}{\Gamma(\nu) l^{2\nu}} \left(\frac{2\nu}{l^2} + 4\pi^2 s^2\right)^{-(\nu + d/2)} \tag{5.33}$$

in which d is the dimensionality of the system.

5.4.3 Comparing Squared Exponential and Matérn Covariance Functions

In this section it is clarified why the Matérn covariance function with $\nu = 5/2$ is preferred throughout this thesis. First of all, one clearly wants a PES that is at least two times continuously differentiable. That only leaves the option with $\nu = 5/2$ of the Matérn class. The second option is the widely used squared exponential covariance function that is called k_{SE} in the following. To show the problem with k_{SE} for interpolating energies the behavior in figure 5.3 is discussed in the following.



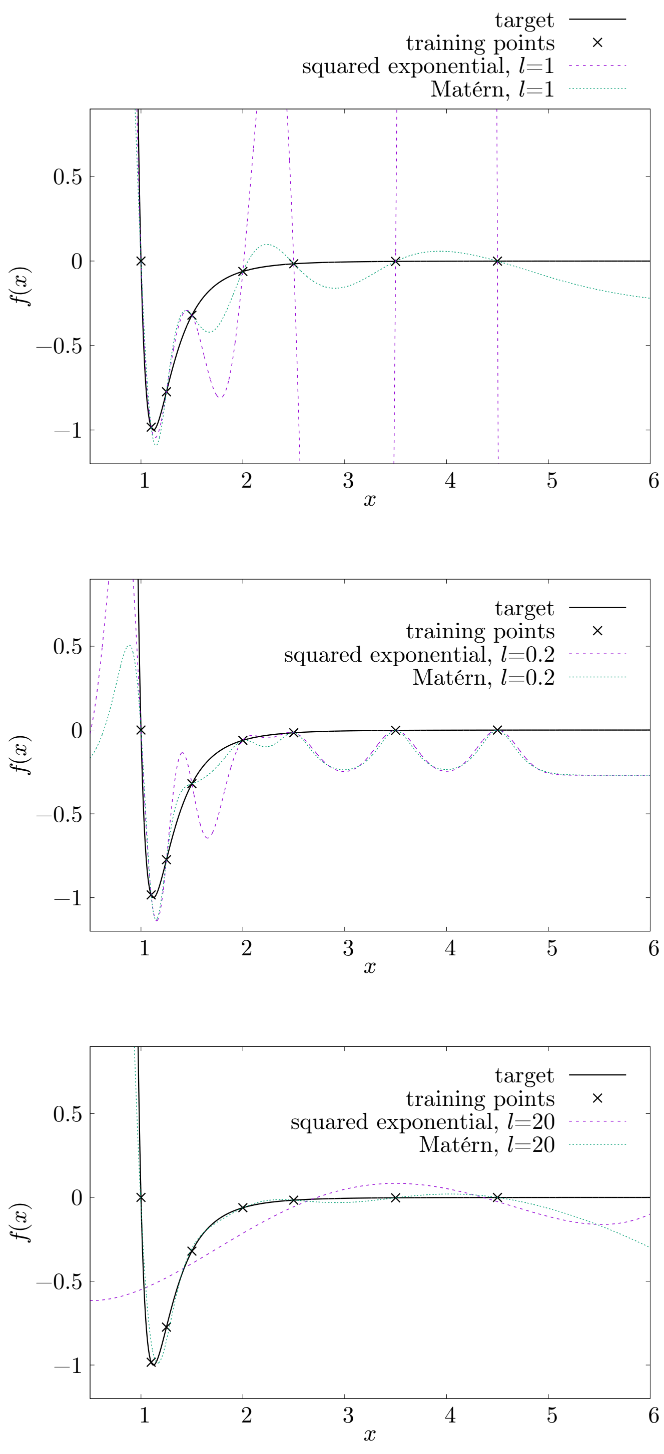
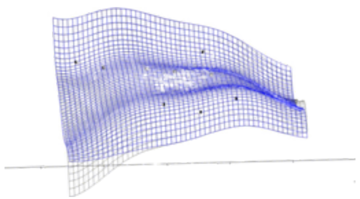


Figure 5.3: GPR of the Lennard-Jones potential regressed using k_{SE} and k_{M} with different length scales.

A Lennard-Jones (LJ) potential (the *target*) is regressed using both k_{SE} and k_{M}. The training points used for that regression are depicted with black crosses. As one can see, k_{SE} with $l = 1$ has problems to cope with the fast changes in the LJ potential. The small area around the minimum leads to strong oscillations at points lying further to the right. By choosing a smaller length scale, e.g. $l = 0.2$, one sees that the oscillations get weaker but the function is represented well exactly only at the training points. Going to a larger length scale of $l = 20$ one sees that k_{SE} is not able to reproduce the function in any meaningful way anymore.
To find the reasons for that one has to consider two things. Firstly, k_{SE} is completely smooth, i.e. all derivatives are continuous. That puts a very strong restriction on the resulting GP. Secondly, one can have a closer look at the spectral densities for one-dimensional systems, see figure 5.4.[9] As one can see the high frequencies in k_{SE} rapidly decline in contrast to the ones in k_{Mat_ν}. For $l = 20$ there are basically only very low frequencies present. This explains why k_{SE} cannot represent the LJ-potential using $l = 20$. The faster changes in the area of the minimum are simply not representable anymore. Using a smaller $l = 0.2$ the higher frequencies are sufficiently represented to easily fit the training data at any point, but the low-frequency terms should be much more dominant to give a non-local representation.

Comparing k_{SE} to $k_{\mathrm{M}} \equiv k_{\mathrm{Mat}_{5/2}}$, which is also used in figure 5.3, one clearly sees that k_{M} performs much better. One still obtains oscillations with $l = 1$ (although they are much smaller) and the very local behavior for $l = 0.2$. But for a larger length scale, i.e. $l = 20$, k_{M} performs exactly as desired. Actually, $l = 20$ is the length scale that was used for the optimizers. A hint for the good behavior can again be found in the graph of the spectral densities, see figure 5.4. Although low-frequencies clearly dominate the covariance function, the presence of much higher frequencies (even for the largest length scale $l = 20$) allows for more flexibility and in consequence the ability to represent the target function adequately. As one can see, the Matérn covariance functions with smaller values for ν represents a larger amount of higher frequencies which also corresponds to the ability to represent discontinuities: Remember that the approximation functions (the predictions of

[9] One can also determine the smoothness of the covariance function by its spectral density/moments of the spectral density [58].



the PES) using GPR with $k_{\text{Mat}_{3/2}}$ are only one time continuously differentiable and are only continuous with $k_{\text{Mat}_{1/2}}$. One can also see, that the Matérn covariance functions seem to become more similar to k_{SE} for higher values of ν. And in fact, for $\nu \to \infty$ one obtains k_{SE}.
Finally, one can conclude that $k_{\text{Mat}_{5/2}}$ has a good trade-off between a presence of high frequencies to allow the sampling of chemical potentials and a decay of higher frequencies that prevents discontinuities. Simply experimenting with different kernels in the algorithms of this thesis suggested the same: The performance of all algorithms presented is consistently better when using $k_{\text{Mat}_{5/2}}$ rather than k_{SE}.

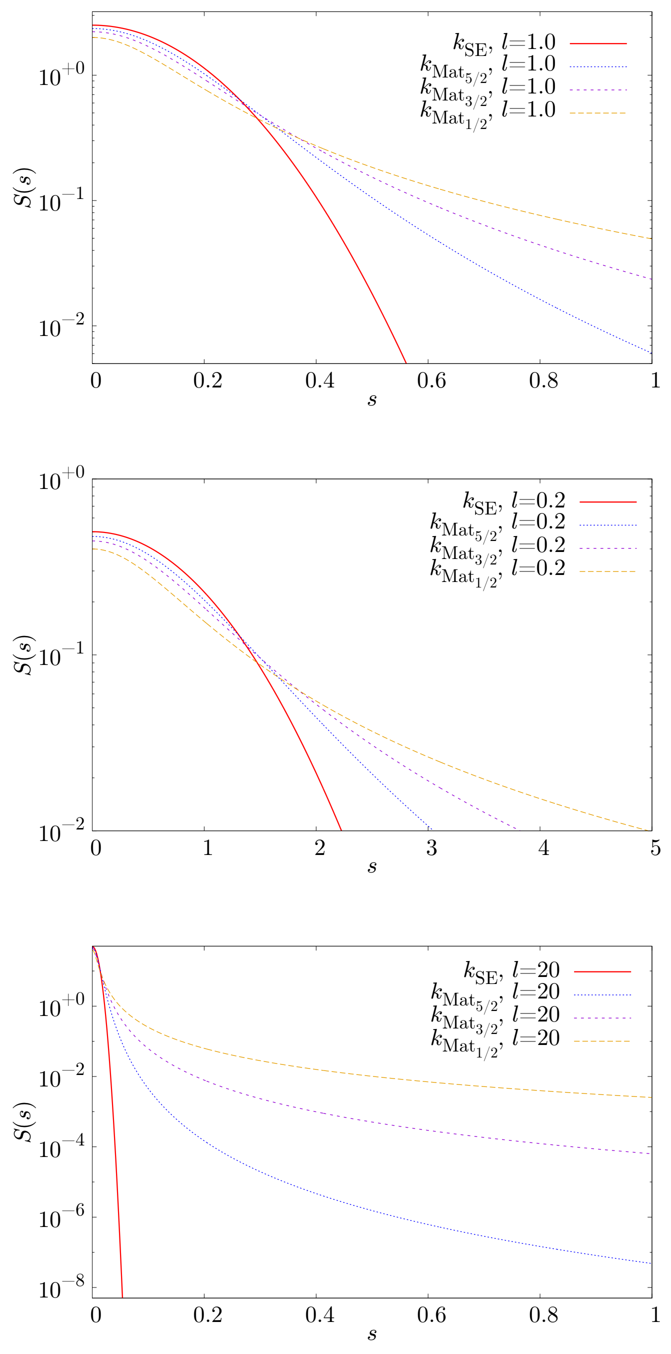
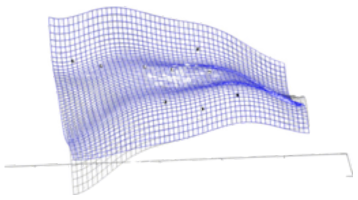


Figure 5.4: Spectral densities of the squared exponential covariance function and three Matérn covariance functions ($\nu = 1/2, 3/2, 5/2$) on a logarithmic scale for different values for l .



5.5 Derivative Information

In this section it is described how one can include gradients $\mathbf{g}_n$ at the data points $\mathbf{x}_n$ in the GPR scheme. The quality of the GP increases with the amount of derivative information one includes in the training process, see figure 5.5. The uncertainty decreases and the mean (the prediction of the GP) converges to the target function. Since gradient information are often cheap to obtain in theoretical chemistry compared to the potential benefit they provide, one wants to use them to construct the GP surrogate for the PES. In this section elements of vectors are written with superscript, indices are written in the subscript.

$$\mathbf{v}_n = (v_n^1, v_n^2, ..., v_n^d) \in \mathbb{R}^d$$

The derivative of a GP is again a GP and one can use certain derivative information for inference with a GP, see Ref. 2, 22, 50, 67 for details. As can be seen in the aforementioned references, e.g. Ref. 22, the covariance function must fulfill sufficient smoothness assumptions and the GP must be differentiable in the *mean square sense*.[10] The covariance functions used in this thesis have the necessary smoothness requirements for the resulting GPs to be two times differentiable in the mean square sense [67]. This allows not only to infer information about the second derivative of the GP-surface but also to train the GP with obtained derivative information. In this thesis mostly first-order derivatives (gradients) are used to train the GP-surface.[11] But still information up to the second-order derivatives (Hessians) are inferred from the GP-surface. The necessary equations for that procedure are shown in the following. Some formulations/sentences of this section

[10] Some more detailed information from Ref. 22: A GP $X(\mathbf{x})$ is differentiable in the mean square sense with respect to the component x^i of the point $\mathbf{x} = (x^1, ..., x^d) \in \mathbb{R}^d$ if $\lim_{h_1,h_2\to 0} E\left[\frac{X(\mathbf{x}+h_1\mathbf{u}_i)-X(\mathbf{x})}{h_1} - \frac{X(\mathbf{x}+h_2\mathbf{u}_i)-X(\mathbf{x})}{h_2}\right]^2 = 0$ with $\mathbf{u}_i$ being the unit vector along direction i. From this definition follows [50] that a GP is differentiable in the mean square sense if and only if (a) the mean value $E[X(\mathbf{x})]$ is differentiable and (b) $\partial^2 k(\mathbf{x}_m, \mathbf{x}_n)/\partial x_m^i \partial x_n^i$, the covariance function of $\partial X(\mathbf{x})/\partial x^i$, exists and is finite at all points $\mathbf{x}_m = \mathbf{x}_n$. The following holds for higher order derivatives. The derivative $\partial^\nu X(\mathbf{x})/\partial x^{i_1}...\partial x^{i_\nu}$ exists for all $\mathbf{x} \in \mathbb{R}^d$ in the mean square sense if and only if $\partial^{2\nu} k(\mathbf{x}_m, \mathbf{x}_n)/\partial x_m^{i_1}...\partial x_m^{i_\nu}\partial x_n^{i_1}...\partial x_n^{i_\nu}$ exists and is finite at all points $\mathbf{x}_m = \mathbf{x}_n$.

[11] Note that it is also possible to train a GP with Hessians if the covariance function is smooth enough. In fact the presented covariance functions allow that.

are based on the author's previously published work, especially the work on the path optimizer [24].

A prior estimate $E_{\text{prior}}(\mathbf{x})$ of the PES is defined that is an estimate of the desired PES before including any real energy calculations (the training data) in the GPR-scheme. GPR then only approximates the distance of $E_{\text{prior}}(\mathbf{x})$ to the real PES. The energy prediction of such a GP trained on energy and gradient information can be done as follows.

$$E(\mathbf{x}) = \sum_{n=1}^{N} w_n k(\mathbf{x}, \mathbf{x}_n) + \sum_{n=1}^{N}\sum_{i=1}^{d} v_n^i \frac{dk(\mathbf{x}, \mathbf{x}_n)}{dx_n^i} + E_{\text{prior}}(\mathbf{x}) \tag{5.34}$$

Predicting the energy like that allows the usage of energy and gradient information at the training points. The elements w_n and the vectors $\mathbf{v}_n$ (with elements v_n^i) can be obtained by solving the following linear equation.

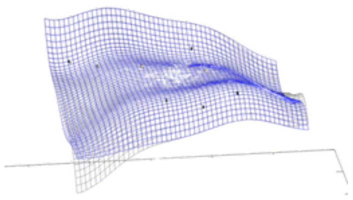
$$K \begin{bmatrix} w_1 \\ \vdots \\ w_N \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \begin{bmatrix} E_1 \\ \vdots \\ E_N \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_N \end{bmatrix} - \begin{bmatrix} E_{\text{prior}}(\mathbf{x}_1) \\ \vdots \\ E_{\text{prior}}(\mathbf{x}_N) \\ \nabla E_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} \\ \vdots \\ \nabla E_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_N} \end{bmatrix} \tag{5.35}$$

The matrix K is called the covariance matrix and has the form

$$K = \begin{bmatrix} k(\mathbf{x}_m, \mathbf{x}_n) + \sigma_{\text{e}}^2 \delta_{mn} & \frac{dk(\mathbf{x}_m, \mathbf{x}_n)}{dx_n^i} \\ \frac{dk(\mathbf{x}_m, \mathbf{x}_n)}{dx_m^i} & \frac{d^2k(\mathbf{x}_m, \mathbf{x}_n)}{dx_m^i dx_n^j} + \sigma_{\text{g}}^2 \delta_{mn}\delta_{ij} \end{bmatrix} \tag{5.36}$$

in which $\delta_{mn}(\delta_{ij})$ is the Kronecker delta. The parameter σ_{e} (σ_{g}) describes noise that one assumes on the energy (gradient) data. In the derivation of GPR in previous sections it was introduced as Gaussian random noise. That noise could be from numerical errors or even methodological errors of the used electronic structure method.

The method also allows to include Hessians in the regression scheme. For these one also has a noise parameter σ_{h}. How this can be done is described in the



appendix, section A.2. For all the presented optimizers in this thesis only energy and gradient information are used (no Hessian information) to build the GPR representation of the PES that is called GPR-PES from now on. One can calculate gradients of the Gaussian process as follows.

$$\frac{d}{dx^k}E(\mathbf{x}) = \sum_{n=1}^{N} w_n \frac{d}{dx^k}k(\mathbf{x},\mathbf{x}_n) + \sum_{n=1}^{N}\sum_{i=1}^{d} v_n^i \frac{d^2 k(\mathbf{x},\mathbf{x}_n)}{dx^k dx_n^i} + \frac{d}{dx^k}E_{\text{prior}}(\mathbf{x}) \tag{5.37}$$

Hessians can be evaluated by further differentiation of this equation.

$$\frac{d^2}{dx^k dx^l}E(\mathbf{x}) = \sum_{n=1}^{N} w_n \frac{d^2}{dx^k dx^l}k(\mathbf{x},\mathbf{x}_n) + \sum_{n=1}^{N}\sum_{i=1}^{d} v_n^i \frac{d^3 k(\mathbf{x},\mathbf{x}_n)}{dx^k dx^l dx_n^i} + \frac{d^2}{dx^k dx^l}E_{\text{prior}}(\mathbf{x}) \tag{5.38}$$

The equations to calculate the variance also changes when one includes derivative information. Let $\tilde{E}_m$ be the random variable that takes on the value of the energy at a molecular configuration $\mathbf{x}_m$. The variance for a GP trained on energy and gradient information is given by the following equation.

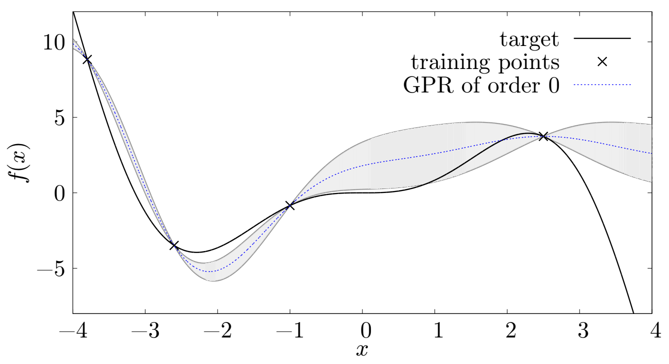
$$\text{Var}\left[\tilde{E}_m\right] = k(\mathbf{x}_m,\mathbf{x}_m) - \mathbf{a}^T\mathbf{c} \tag{5.39}$$

The vector $\mathbf{c} = (c_{e1}, ..., c_{eN}, \mathbf{c}_{g1}, ..., \mathbf{c}_{gN})$ (in which $\mathbf{c}_{gi} = (c_{gi}^1, ..., c_{gi}^d)$) is the solution of the linear system

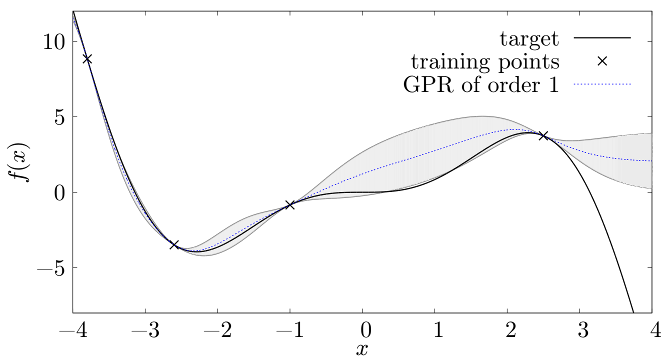
$$K\mathbf{c} = \mathbf{a} \tag{5.40}$$

with the covariance matrix K from equation (5.36).
The vector $\mathbf{a} = (a_{e1}, ..., a_{eN}, \mathbf{a}_{g1}, \mathbf{a}_{gN})$ (in which $\mathbf{a}_{gi} = (a_{gi}^1, ..., a_{gi}^d)$) consists of the elements $a_{ei} = k(\mathbf{x}_i, \mathbf{x}_m)$ and $\mathbf{a}_{gi} = \nabla_{\mathbf{x}_i} k(\mathbf{x}_m, \mathbf{x}_i)$ for all $i = 1, ..., N$.

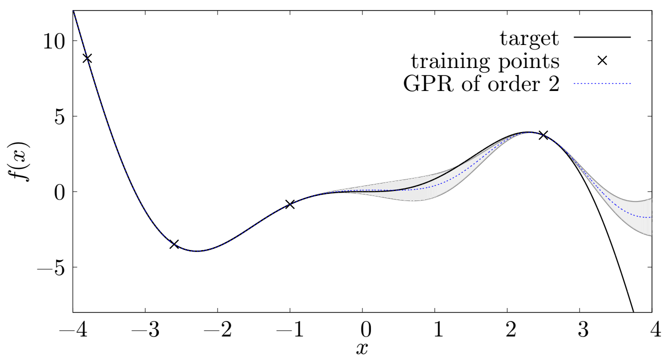
In the appendix, section A.3, the derivatives of the two used covariance functions are shown that are necessary to implement GPR for two times differentiable surfaces.



(a) GPR using only the function value at the training points.

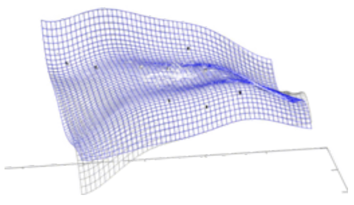


(b) GPR using the function value and the gradient at the training points.



(c) GPR using the function value, the gradient, and the Hessian at the training points.

Figure 5.5: Comparing different *orders* of GPR (the order of the derivatives included). The area between the mean +/− two times the standard deviation is shown in grey.



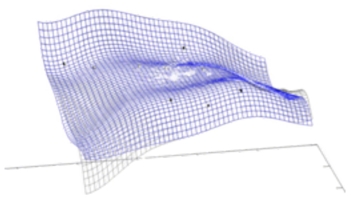
5.6 Maximum Likelihood Estimation for Geometry Optimizers

In the appendix, section A.4, the *maximum likelihood estimation* for optimizing the hyperparameters in GPR is reviewed. Here, however, it is explained why this method is not used in the work for this thesis. Some studies in theoretical chemistry suggest that optimization of the maximum likelihood method can increase the performance of GPR [44, 70]. For the geometry optimizers presented in this thesis, this method was found to generally increase the number of energy evaluations needed until convergence. Several tests for the optimizers were performed with and without optimizing the hyperparameter for the length scale, l. Optimization of the hyperparameter was done after every step of the optimization and until one obtains a fully optimized hyperparameter or a maximum change of the hyperparameter is reached. In general a worsening of the results was obtained compared to simply choosing a fixed value for the length scale. Especially for very fast and small optimizations with only few training points a fixed length scale is clearly superior to the maximum likelihood approach.

The main reason for this is that maximum likelihood optimization, as a statistical method, requires a lot of training points before it yields meaningful results. However, the geometry optimizers often handle only a few training points, at the beginning even only one. The resulting information content of the likelihood is very limited for geometry optimization. Furthermore, one has to consider that the algorithms choose new training points based on their current approximation of the PES. This approximation, however, is dependent on the hyperparameters that are currently used. Changing the hyperparameters might result in a situation in which other training points should have been preferred to built the GPR-PES, based on the predictions using the new hyperparameters.
This mutual dependency of the hyperparameters and the choice of the training points might decrease the efficiency of the maximum likelihood approach. In fact one assumes that the training points are sampled independently of each other which is clearly not the case in geometry optimization. Furthermore, it is bad for the performance of the presented GPR implementation itself since the Cholesky

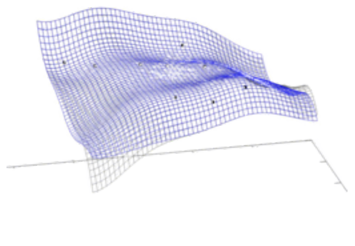
decomposition of the covariance matrix has to be recalculated completely after changing a hyperparameter. The iterative version of the Cholesky decomposition, introduced in section 6.1, cannot be used. Therefore, using the maximum likelihood approach can increase the overhead of the optimizer.

In the first geometry optimizer presented the length scale parameter is adapted nevertheless. This is done in a systematic way, but not based on maximum likelihood optimization, and also only very rarely. The details on that are explained in the chapters on the respective optimizers. However, the usage of hyperparameter optimization by maximizing the marginal likelihood can be beneficial GPs trained on a larger number of training data.



Part III

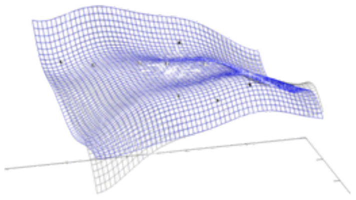
Optimization Algorithms Based on Gaussian Process Regression



This part of the thesis shows the main results/developments achieved in the thesis. Some distinct features of the specialized GPR implementation are highlighted and the developed optimizers are explained in detail. Small discussion parts, specific to each of the three optimizers, are provided. A larger discussion section at the end of this part is concerned with all the presented optimizers.

All the optimizers that are presented could be called *surrogate optimizers* since they built a surrogate for the PES that is called GPR-PES. Classical optimization is then performed on the GPR-PES to obtain a new guess for the desired structure on the real PES. The GPR-PES is then improved by adding a new training point to it, i.e. the GPR-PES is built in an iterative way. Three different algorithms to optimize minimum structures, saddle points, and minimum energy paths are explained in detail. All three algorithms are benchmarked on several test systems. Thereby, 25 test systems are employed for all three optimizers: The test set suggested by Baker [8] contains comparably small systems but they can be expected to be (at least partially) quite challenging for geometry optimizers. Furthermore, at least one larger test system is used to benchmark the algorithms. Visualizations of the used test systems are given in the respective chapters. Since all three optimizers are already published, some of the explanations are taken from the respective articles [24–26] although most of it was rewritten for this thesis.
Finally, one last algorithm, yet unpublished, is presented very briefly that allows accurate updating of Hessians via GPR.
All the algorithms presented in the thesis are open source and available in the DL-FIND library.

All physical properties are expressed in atomic units (Bohr for positions and distances, Hartree for energies), unless other units are specified.

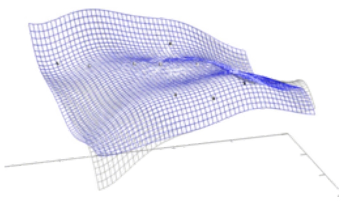


6 Features of the GPR Implementation

In the work for this thesis a Fortran implementation of GPR was created. In this section specific details of this implementation are explained. New developments are explained that improve the straightforward implementation of the GPR theory to make GPR well suited for geometry optimization in chemistry.

6.1 Overcoming the Cubic Scaling of GPR

As discussed before the standard way of solving the linear system in GPR is using the Cholesky decomposition. It is a way of exactly solving a linear system of the form $A\mathbf{x} = b$ with a Hermitian, positive-definite matrix A. When A is a $d \times d$ matrix the Cholesky decomposition of A to a lower triangular matrix L scales with $\mathcal{O}(d^3)$. It is then $A = LL^*$ or simply $A = LL^T$ for real numbers. In this thesis only real-valued matrices have to be considered. Solving the linear equation with the decomposed matrix scales only with $\mathcal{O}(d^2)$. Therefore, the cubic scaling of the Cholesky decomposition limits the number of training points that can be used in GPR. In geometry optimization one optimally makes very few energy evaluations. One also includes gradient information of the training points into the GP. This is mainly done because the gradient is often comparably cheap to evaluate at a certain configuration of the system when the energy at that configuration is evaluated already. The value of the information encoded in the gradient clearly justifies the small computational overhead. The usage of gradient information in GPR increases the size of the covariance matrix to $N(d + 1) \times N(d + 1)$ if N is the number of training points and d the number of dimensions in the system. In this case, the Cholesky decomposition scales cubically with $N(d + 1)$. Luckily,



Cholesky decomposition can be understood as an iterative process in the sense that the decomposition matrix L can be constructed row-wise. If one reorders the covariance matrix of equation (5.35) (the same can be done when using Hessian information as well) in a way that the rows (and columns) concerning the same training point are next to each other, one can easily add new training points to the covariance matrix by adding new rows (and columns). Therefore, one can reuse the existing covariance matrix to construct the new one. In a similar way one can reuse the older decomposition L of the covariance matrix and add new rows to it. The addition of a new training point to the decomposition does only scale with $\mathcal{O}(N(d+1)^2)$. Since the evaluation of the GP already scales with $\mathcal{O}([N(d+1)]^2)$ the overall scaling of GPR can be considered quadratic when one iteratively builds the GP-surface. Only when one evaluates Hessians on the GPR-PES this scaling worsens. Evaluating Hessians on a GPR-PES that was trained based on energies and gradients scales with $\mathcal{O}(d^3N)$. If one trains the surface with Hessians, the scalings naturally becomes worse.

6.2 Multi-level GPR

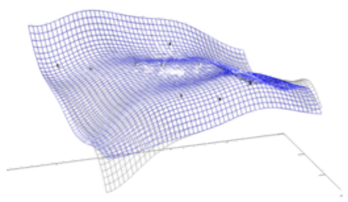
Another problem arising from the poor scaling of GPR is memory restrictions. Whether one builds the covariance matrix in an iterative way or not, see section 6.1, the memory demand is still the same, and it scales cubically: In a system with 300 dimensions, using double-precision, and including gradient information the memory demand for the covariance matrix for 10 training points is 36MB, for 100 training points it is $\sim$ 3.6GB, for 300 training points it is $\sim$ 32.6GB. The GPR algorithm becomes impracticable for a larger number of training data. A possible way to overcome this problem is to use a *multi-level approach* that was presented in the author's first paper on geometry optimization [25].

The simplest way to bring the scaling down would be to simply use only, e.g., the last 50 training points. That would make the scaling independent of the length of the optimization history. A much better way is to use the older training points to build another GP: Let N be the number of training points. N increases by one every step along the optimization. As soon as $N = N_{\text{max}}$ one can take the oldest

m training points to build a separate GP called GP_1. The remaining $N_{\text{max}} - m$ training data are used to correct the prediction of GP_1: GP_1 is used as the prior mean function for the GP with the newer training data. The resulting GP with the new training points and GP_1 as a prior mean function is called GP_0. Newer training points are added to GP_0. As soon as GP_0 holds N_{max} training points again, rename GP_1 to GP_2, use the oldest m training points of GP_0 to build a new GP called GP_1 with GP_2 as a prior mean function. The resulting PES prediction of GP_1 is used as a prior mean function for GP_0 with the remaining $N_{\text{max}} - m$ newest training data. This process is repeated every time GP_0 holds N_{max} training points. The number of GPs increases but the overall scaling (number of computations and required memory) becomes constant with respect to the length of the optimization history. This makes GPR-based optimization also applicable in higher-dimensional systems that would otherwise be out of reach. In the example above, a system with 300 dimensions, setting $N_{\text{max}} = 60$ and $m = 10$, the algorithm only needs $\sim$ 3GB of memory space for 300 training points, not 32.6GB. In Ref. 25, 26 it was demonstrated that the capabilities of GPR for geometry optimization are not significantly diminished by using the multi-level approach.

6.3 Summary and Further Details of the Implementation

- The Cholesky decomposition is done in an iterative way in the sense that one can easily add new training points to an existing GP without increasing the computational scaling of the algorithm, see section 6.1.
- A *multi-level approach* to GPR to overcome further scaling problems due to memory constraints has been implemented, see section 6.2.
- The Hessian evaluation of GPR is parallelized row-wise using OpenMP. The variance evaluation in GPRMEP is parallelized (it must be evaluated at several points).
- Since the Hessians of the PES are symmetric, only the lower half of the Hessians is used for training. This reduces the size of the covariance matrix



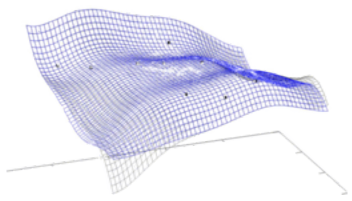
considerably.

- The covariance matrix is stored in a linearized fashion: Only the lower half of the matrix is saved and the remaining entries of the rows are all lined up one after another to give a vector. This allows for fewer memory demand and still fast iteration over its elements.

- The GPR implementation itself is programmed in a modular way. That means that a data type called *gpr_type* exists and the respective module called *gpr_module* contains all private and public functions operating on that type. The necessary parameters for optimization on GPR are collected in a separate data type called *optimizer_type*. The implementations of the geodesic approach [85], the curve interpolation procedures based on GPR, and the path optimization algorithms to find MEP that we call *GPRMEP* later on, are done in a completely object-oriented way.

- GPR was implemented with several kinds of possible functions for the prior mean.
  - The mean value of all energy values in the GP is taken as a constant prior mean.
  - To that mean value one can add/subtract a constant that can be manually defined.
  - Another GP can be taken as an offset. This is used in the multi-level scheme, see section 6.2.
  - A linear interpolation between the first and the last added training point is chosen as a prior mean. This is used, for example, to interpolate a curve based on GPR (as an alternative to splines).
  - A Taylor expansion (of order zero, one, or two) around an arbitrary point $\mathbf{x}_T$ can be used as a prior mean. One should, however, include the point $\mathbf{x}_T$ and its known derivatives as training data as well to guarantee that the GP accurately describes this point. This is used in the GPR-based update of Hessians.

- The implemented GPR procedures allow to arbitrarily include points of different *order*, i.e. points with different amount of derivative information.
- Optimization of the hyperparameters by maximizing the likelihood is implemented, see section A.4. However, it is recommended to avoid this procedure for geometry optimization since it prohibits to build the GP in an iterative way which has computational disadvantages.

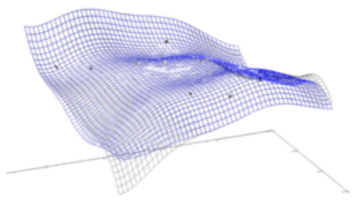


7 Minimization Algorithm

In this chapter it is explained in detail how geometry optimization can be improved using GPR. This chapter is mainly based on Ref. [25]. However, the results vary slightly from the paper, as I replaced the multi-level approach, see section 6.2, by the iterative implementation of the Cholesky decomposition, see section 6.1. The explanations given in this chapter are often similar to Ref. [25].

The basic idea of the GPR optimizer is to use the energy and gradient information of the PES to build a GP surrogate, the GPR-PES. The algorithm searches for a minimum on this GPR-PES to estimate a minimum on the real PES. The energies and gradients at this estimate are evaluated and included in the GPR-PES. Then a new search for a minimum on the GPR-PES is started. This process is repeated until the optimizer can be considered to be converged.

So far, this is similar to other optimizers with different surrogate models based on Taylor expansions instead of GPR [74, 84]. In the following, the optimization procedure is explained in detail. Section 7.2.3 explains the algorithm on an example run of the resulting algorithm in the one-dimensional Lennard–Jones potential.



7.1 Convergence Criteria

In order to define convergence for the optimizer the standard convergence criteria of DL-FIND for the step size and the gradient are used: The Euclidean norm of the step vector, and the gradient vector, as well as the maximum entry of both vectors have to drop below a certain threshold. The step vector is the vector that points from the last estimate to the new estimate of the minimum. It describes the proceeding of the optimization run. Given a single tolerance value, δ, the convergence criteria in DL-FIND are

$$\max_i(g_i) < \delta_{\max(g)} := \delta \tag{7.1}$$

$$\frac{|\mathbf{g}|}{d} < \delta_{|g|} := \frac{2}{3}\delta \tag{7.2}$$

$$\max_i(s_i) < \delta_{\max(s)} := 4\,\delta \tag{7.3}$$

$$\frac{|\mathbf{s}|}{d} < \delta_{|s|} := \frac{8}{3}\delta \tag{7.4}$$

where $|\mathbf{g}|$ ($|\mathbf{s}|$) is the Euclidean norm of the gradient (step vector), and $\max_i(g_i)$ ($\max_i(s_i)$) its maximum entry. The variable d stands for the number of dimensions in the system. If these four criteria are fulfilled, the algorithm is considered to be converged. Note that convergence is tested for the gradient on the underlying ab-initio data rather than the GPR surrogate.

7.2 The Optimization Algorithm

In the first step the GPR-PES is built with only a single energy and gradient evaluation at the starting point, $\mathbf{x}_0$. In later steps, all obtained energies and gradients from the N training points are used to build the GPR-PES and then find the minimum, $\mathbf{x}_N^{\text{GPmin}}$, on the GPR-PES. The evaluation of the GPR-PES is very cheap, especially compared to the evaluation of the PES via electronic structure calculations. Therefore, the search for $\mathbf{x}_N^{\text{GPmin}}$ can be carried out very fast with an arbitrary optimization method, like for instance a L-BFGS optimizer [48], as has been used in this thesis.

Usually the search for a minimum on the GPR-PES is started at the last training point that was included in the GPR-PES. In rare cases this choice of the starting point can go wrong: If the direction along the optimization changes by more than a 90 degree angle, or if the absolute value of the gradient gets larger, the optimization is restarted multiple times. A minimum search is started at each of the 10% of training points with lowest energies. The lowest minimum found is the next $\mathbf{x}_N^{\text{GPmin}}$. The obvious optimization step, $\mathbf{s}'_N$, after one has obtained $N \geq 1$ training points would be to take the step vector to position $\mathbf{x}_N^{\text{GPmin}}$ as the next guess for our minimum on the PES.

$$\mathbf{s}'_N = \mathbf{x}_N^{\text{GPmin}} - \mathbf{x}_{N-1} \tag{7.5}$$

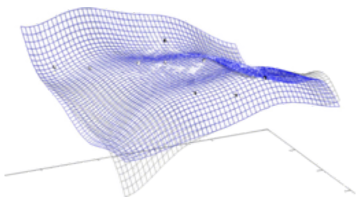
The first training point, $\mathbf{x}_0$, is defined as the starting point of the optimization, and $\mathbf{x}_{N-1}$ is the last estimate of the PES minimum after obtaining $N-1$ additional training points. This already yields a functional optimizer, but its performance is rather poor due to a well known problem: GPR and other machine learning techniques are not capable of accurate regression in regions where only few training points or none at all are given. Simply put, in one dimension *interpolation works much better than extrapolation.* An iterative optimization as described above is obviously largely based on extrapolation since it most likely makes a step away from known training points. By overshooting the estimated minimum on purpose, the problem of finding the minimum is presented as an interpolation rather than an extrapolation.

7.2.1 Overshooting

The first optimization step is carried out as described by equation (7.5), and the first step is defined as $\mathbf{s}_1 = \mathbf{s}'_1$. From the second step onward, the cosine of the angle between the last optimization step, $\mathbf{s}_{N-1}$, and the estimated new step, $\mathbf{s}'_N$, is determined as

$$\alpha_N = \frac{(\mathbf{s}_{N-1}, \mathbf{s}'_N)}{|\mathbf{s}_{N-1}||\mathbf{s}'_N|} \tag{7.6}$$

with $(\cdot, \cdot)$ being the Euclidean dot product. The closer α_N is to 1, the smaller the angle becomes. If α_N is smaller than 0, the direction of the optimization is changed by more than a 90 degree angle. If it is close to -1, the direction is completely



inverted. As soon as

$$\alpha_N > 0.9, \tag{7.7}$$

the confidence in taking a step in the correct direction is fairly high and the initially estimated $\mathbf{s}'_N$ is scaled up to obtain the next optimization step

$$\mathbf{s}_N = \lambda(\alpha_N)\mathbf{s}'_N \tag{7.8}$$

for $N \geq 2$, introducing the scaling factor

$$\lambda(\alpha_N) = 1 + (\lambda_{\max} - 1)\left(\frac{\alpha_N - 0.9}{1 - 0.9}\right)^4 \tag{7.9}$$

with a maximum scaling factor of $\lambda_{\max}$ so that $1 \leq \lambda(\alpha_N) \leq \lambda_{\max}$. This scaling factor is depicted in figure 7.1.

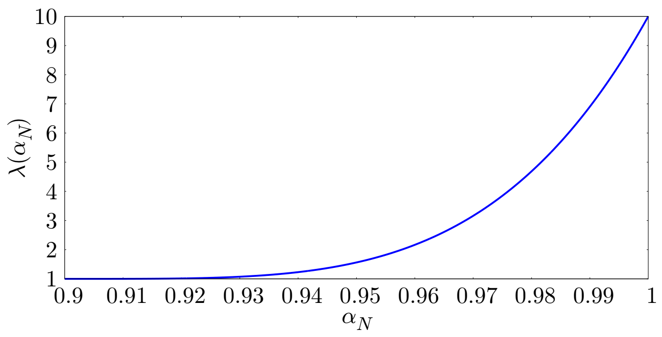


Figure 7.1: The scaling factor for overshooting the step, see equation (7.9), with $\lambda_{\max} = 10$ is plotted against α_N.

To avoid large overshooting in the area around the actual minimum of the PES this scaling factor is only applied if the estimated step, $\mathbf{s}'_N$, does not satisfy the convergence criteria of the maximum step entry in equation (9.28). Furthermore, close to convergence the maximum scaling factor is limited through

$$\lambda_{\max} = \left(1 + \tanh\left(\beta^2 - 1\right)\right)\frac{\tilde{\lambda}_{\max} - 1}{2} + 1 \tag{7.10}$$

in which

$$\beta = \max_i(s'_i)/\delta_{\max(s)} \tag{7.11}$$

is the ratio of the maximum entry of $\mathbf{s}'$ and $\delta_{\max(s)}$, the convergence criterion for the maximum step entry from equation (9.28). The variable β indicates how close the algorithm is to convergence with respect to the maximum entry of the step vector. If $\beta \leq 1$, the convergence criterion is met. No scaling occurs in this region. See figure 7.2 for a plot of $\lambda_{\max}$ against β.

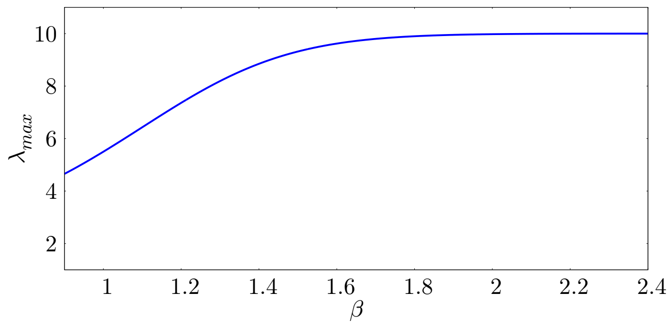
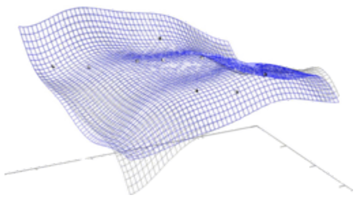


Figure 7.2: The limitation of the scaling factor, see equation (7.10), is plotted against the variable β, see equation (7.11). The upper limit of $\lambda_{\max}$ is set to $\tilde{\lambda}_{\max} = 10$.

To some extent, the limitation of the highest possible scaling factor, $\tilde{\lambda}_{\max}$, to $\lambda_{\max}$ via equation (7.10) is intended to guarantee a smooth transition into the region of convergence. However, keeping $\lambda_{\max}$ above a threshold of at least $\tilde{\lambda}_{\max}/2$ at the point where the convergence criterion is met does not seem to hinder convergence. Consequently, $\lambda_{\max} \approx \tilde{\lambda}_{\max}/2$ is kept in the area of convergence. The value of $\tilde{\lambda}_{\max}$ is chosen to be 5 at the beginning of the optimization. It is increased by 5%, if the overshooting procedure according to equation (7.9) is performed more than once in a row, i.e. that the criteria of equation (7.7) are satisfied for two consecutive steps in the optimization procedure. At the end of each optimization step, the estimated step size, $|\mathbf{s}_N|$, is limited by the maximum step size, $s_{\max}$, that is set externally for the optimization procedure. This is similar to all other optimizers in DL-FIND.



7.2.2 Separate Dimension Overshooting

In several systems one often observes that a few dimensions seem to converge very slowly, while the convergence in the other dimensions has already been accomplished. This is especially the case for longer optimization runs. One reason for this may be that only one length scale parameter, l, in equation (5.32) is used, and no different length scales for different dimensions are assumed. On the other hand, it is not easy to find suitable parameters for every dimension, and introducing more parameters makes the optimizer become more prone to chance.

Alternatively, one can make use of the fact that the correct solution to the optimization problem can be overshot quite a bit. The *separate dimension overshooting* is introduced by considering every dimension independently of the others. If the optimizer has monotonically changed the value of the coordinate in this dimension over the last 20 optimization steps, a one dimensional GP is built to represent the optimization along this single coordinate. This GP approximates the value of the corresponding coordinate with respect to the number of the steps taken: The position of the training points for this GP is simply the number of the steps along the optimization procedure, so its training points are equidistant and it interpolates the value of the considered coordinate at the respective step. On this GP the next maximum/minimum is searched assuming it could be a good guess for the dimension's value at the real minimum of the PES. Thereby, the coupling of the different coordinates is ignored. To give the optimization procedure time to explore the omitted coupling, the separate dimension overshooting is suspended for 20 optimization steps after it was performed.

In order to restrict the overshooting to a reasonable regime, the limit of the separate dimension overshooting is restricted by a factor of 4 compared to the originally estimated step without any overshooting. Furthermore, the separate dimension overshooting is only applied if it suggests higher overshooting than the scaling factor in equation (7.9) and if the convergence criterion for the maximum step entry from equation (9.28) is not satisfied. Also this overshooting procedure is finally limited by the maximum step size, $s_{\max}$, allowed for the optimization procedure.

7.2.3 The Algorithm in One Dimension

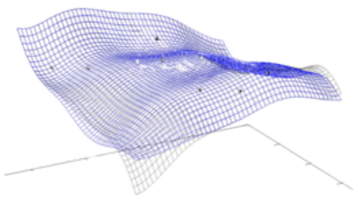
In this section the overall optimization process is explained with a simple one-dimensional example PES $E(x)$ illustrated in figure 7.3.

- *Step* 1: To begin with, the GPR-PES is built with the energy and gradient information from the starting point. The minimum on the GPR-PES is found. It is shown by a green circle. This is the next estimate for the PES minimum, and the energy and gradient of the PES is calculated at that position, indicated by the arrow pointing to the real PES.

- *Step* 2: After evaluating the energy and gradient at the estimate from the last step, the next GPR-PES is now built with two training points. In this example, the new minimum of the GPR-PES leads the algorithm in the same direction as in Step 1. Therefore, the estimated step size is scaled up in the overshooting procedures described above. The overshooting to a more distant point is indicated by the tilted arrow which points to the next estimate at which the energy and the gradient are calculated. If the estimated step size is now larger than the externally set maximum step size $s_{\max}$, we scale the step down to a step size of $s_{\max}$.

- *Step* 3: The minimum on the new GPR-PES, with now three training points, leads to a step in the opposite direction of the last step. Therefore, no overshooting is performed.

- *Step* 4: The next estimate for the minimum is close enough to the last estimated minimum, therefore the convergence criteria for the step size are satisfied. Calculating the gradient at estimate 4, also shows that the convergence criteria for the gradient are satisfied. The optimizer is completely converged.



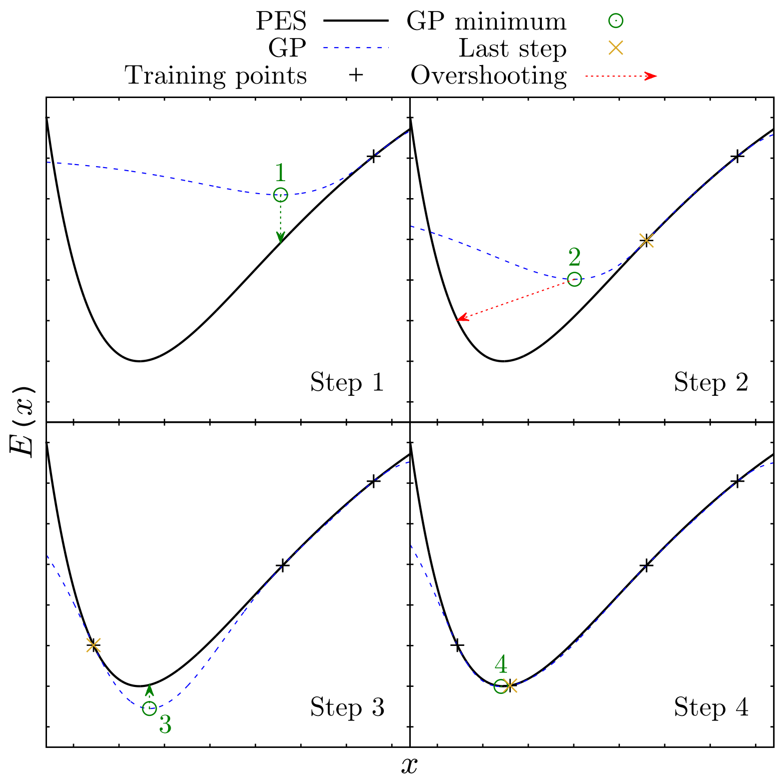


Figure 7.3: The basic idea of the GPR optimizer in the case of a Lennard–Jones potential as a simple example for a PES.

The limitation of the step size, called $s_{\max}$, lies roughly between 0.5 and 1 *a.u.* This prevents the overshooting process from shooting in a region outside of the domain in which the chosen electronic structure calculations are valid.

7.2.4 Parameters

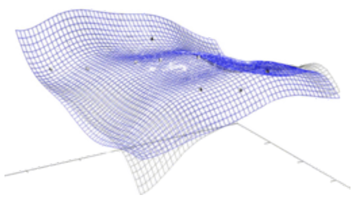
For all results presented in this section the Matérn covariance function of equation (5.32) is used. The parameter σ_{f} is set to $\sigma_{\mathrm{f}} = 1$ since it does not influence the result. The only other parameter in the covariance function is l. It is set to $l = 20$ at the beginning of the optimization. A dynamic approach is chosen: Every step along the optimization on which the gradient has become larger, instead of smaller, $1/l^2$ is increased by 10% of its current value. This leads to a shrinking characteristic length scale along the optimization. This means that the steps predicted by the GPR optimizer will become smaller and the training points closer. A smaller characteristic length scale is also advisable towards the end of an optimization procedure, as one often needs more careful steps when approaching the minimum. The changing of the hyperparameter l is turned off when the covariance matrix becomes too large (what *too large* means depends on the memory available on the computing system). In this case the iterative Cholesky can be used and the algorithm stays efficient. The noise parameters, see equation (5.36), are chosen to be $\sigma_\mathrm{e} = \sigma_\mathrm{g} = 10^{-7}$ which is a compromise between the smallest possible value, and numerical stability tested on various test systems. It is also noteworthy that the Matérn covariance function is rather insensitive to changes of the σ parameters. One can also use the *maximum likelihood principle* to optimize the parameters, see section A.4. Nevertheless, in our test cases the presented dynamic approach shows much better results.

The algorithm also includes an offset in the form of the prior mean function, see equation (5.34). Far away from any training points, the GPR-PES slowly converges to this function. This fact is exploited in the optimizer: The prior mean is chosen to be a constant that is much higher than the energy values observed in the system. This will restrict the optimization to a reasonable area around the observed training points, and guarantees that a minimum on the GPR-PES can be found at any time. The prior mean for the minimization procedure is chosen to be

$$E_\mathrm{mean} = \max_i E_i + 10 \tag{7.12}$$

The value of E_mean can change with an increasing amount of training points and is reevaluated if new training points are added to the GPR-PES.

Just like in L-BFGS optimizations, the maximum step size, s_max, is the only parameter that has to be specified by the user. The parameters in the overshooting schemes were chosen to provide reasonable performance on the Baker test set shown in section 7.3. The sensitivity of the performance on these parameters is small.



7.3 Applications/Benchmarks

The optimization algorithm was tested on several systems. First, a set of 25 test systems suggested by Baker was chosen [8]. The starting points of the optimization were chosen following Ref. 8. They are close to a TS on the Hartree–Fock level. In contrast to Ref. 8 the semi-empirical AM1 [27] method is used for the electronic structure calculations. The resulting minimum structures are shown in figure 7.4. These test systems are in the following referred to as IDs 1 to 25. Note that the structures with ID 23 and 24 start at different geometries but end up in the same minimum. Additionally, a more realistic test case is set up: a part of a previously investigated molybdenum amidato bisalkyl alkylidyne complex [72] that includes 41 atoms, see figure 7.5. Electronic structure calculations are carried out with the BP86 functional [12, 60] in the def2-SVP basis set [80].

The optimization runs on this *molybdenum system* are given the IDs 26, 27, and 28. For run 27, a different starting point from those in runs 26 and 28 was chosen. Runs 26 and 28 begin at the same starting point and only differ in the chosen convergence criteria: The convergence criteria were chosen according to equations (9.26 – 9.29) with $\delta = 4.5 \cdot 10^{-4}$ for the run on the molybdenum system with IDs 26 and 27. The stricter criterion, $\delta = 1 \cdot 10^{-4}$, was chosen for the run on the molybdenum system with ID 28, and $\delta = 3 \cdot 10^{-4}$ was set for the *Baker systems* with IDs 1 to 25.

The maximum step size was set to 5 $a.u.$ (never reached) for L-BFGS since it yields the best performance. The maximum step size for the GPR optimizer was set to 0.5 $a.u.$ for the Baker systems and 1 $a.u.$ for the molybdenum system runs. The number of steps in the L-BFGS memory is chosen to be 50 for the molybdenum system, and equal to the number of dimensions in the Baker systems. That is the default setting in DL-FIND. The L-BFGS optimizer in DL-FIND also employs a variable trust radius based on energy decrease, see section 2.2 [42]. All presented calculations are performed in Cartesian coordinates. The GPR optimizer is in principle able to handle other coordinates, but the adaptations of the algorithm needed to perform well with these are not trivial. Especially the optimal length scale parameter l may be different in every dimension. This is not possible with the presented implementation.

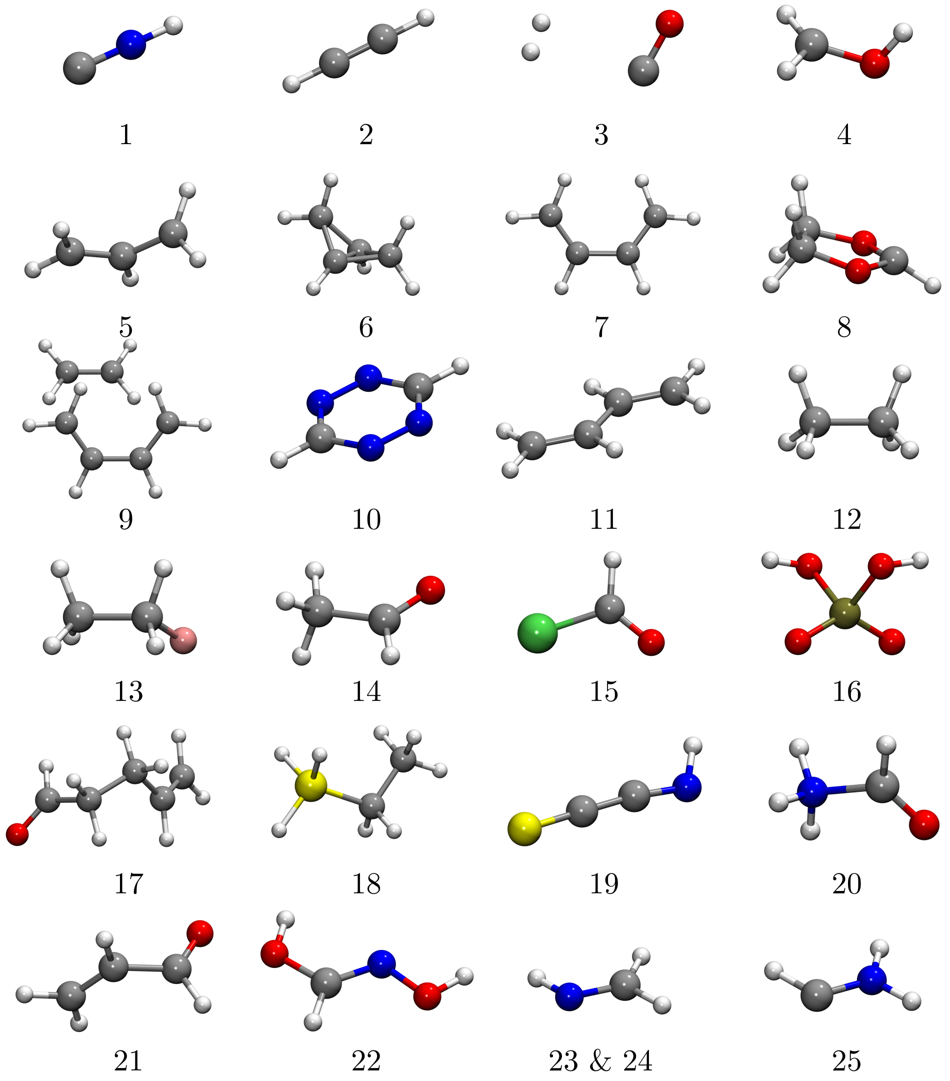
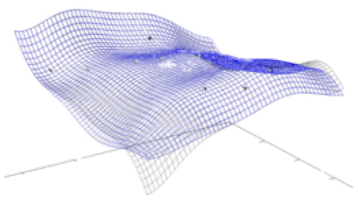


Figure 7.4: The minima of the test systems suggested by Baker [8]. The structures with ID 23 and 24 are the same.



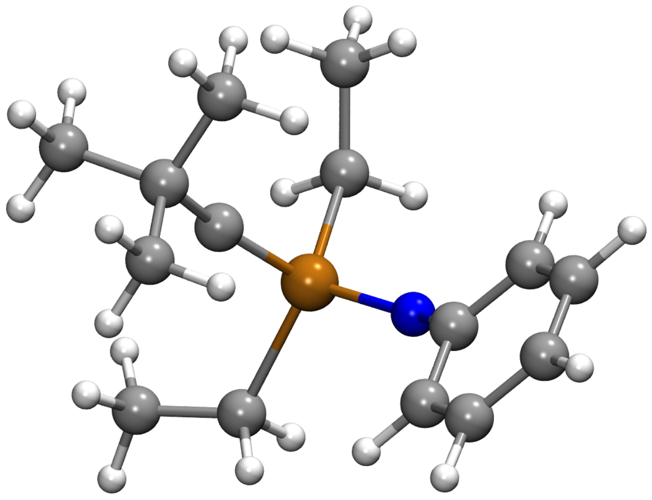
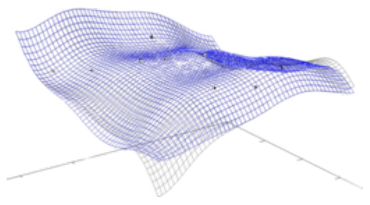


Figure 7.5: The minimum structure of the molybdenum amidato bisalkyl alkylidyne complex found by our GPR optimizer in the run with ID 26. Molybdenum is depicted in golden brown, nitrogen in blue, carbon in grey and hydrogen in white.

Rigorously proving the convergence order of the optimizer is rather difficult if not impossible. Instead, a comparison between the GPR optimizer and the superlinearly converging L-BFGS optimizer of DL-FIND is shown in table 7.1. The number of steps both optimizers take until convergence is compared as well as the obtained minima according to their energy and the RMSD value of their geometries. In the respective paper [25] a further comparison to the steepest descent and the conjugate gradient methods is shown. They perform much worse than L-BFGS. One can also find results of the optimization runs on the Baker test set using DFT instead of AM1. The DFT-based benchmarks in Ref. 25 are consistent with the ones presented here: The electronic structure method does not seem to have a strong influence on the optimization performance.

Table 7.1: A comparison of the L-BFGS and the GPR optimizer in the presented test systems, sorted by the number of dimensions d. The number of steps required until convergence is given for the L-BFGS and the GPR optimizer, Δsteps is their difference, Δenergy is the energy difference between the minima in Hartree, RMSD denotes the root-mean-square deviation of their atomic positions in Ångström.

	steps					
d	GPR	L-BFGS	Δsteps	Δenergy	RMSD	ID
123	120	251	131	2.00×10^{-11}	2.34×10^{-1}	26
123	96	223	127	5.02×10^{-8}	2.94×10^{-1}	27
123	166	436	270	-2.18×10^{-4}	2.77×10^{-1}	28
48	73	83	10	8.47×10^{-9}	1.43	9
42	94	104	10	2.29×10^{-7}	2.92×10^{-2}	17
33	29	39	10	1.93×10^{-6}	1.44×10^{-3}	18
30	26	27	1	-3.32×10^{-7}	2.34×10^{-3}	6
30	58	62	4	-4.17×10^{-8}	5.48×10^{-3}	7
30	38	42	4	-1.45×10^{-4}	3.08×10^{-4}	8
30	35	37	2	-6.40×10^{-9}	4.03×10^{-3}	11
24	29	31	2	-2.62×10^{-7}	9.16×10^{-4}	5
24	13	15	2	1.67×10^{-8}	6.63×10^{-5}	10
24	15	23	8	1.09×10^{-7}	1.19×10^{-4}	12
24	14	19	5	2.52×10^{-7}	3.09×10^{-4}	13
24	19	22	3	6.20×10^{-10}	3.83×10^{-4}	21
21	16	23	7	-7.08×10^{-2}	3.36×10^{-4}	14
21	30	80	50	-2.49×10^{-6}	8.76×10^{-1}	16
21	22	25	3	-3.68×10^{-7}	1.96×10^{-4}	20
21	19	22	3	-3.50×10^{-6}	3.54×10^{-4}	22
15	12	14	2	-1.45×10^{-8}	9.73×10^{-5}	4
15	13	26	13	-9.58×10^{-8}	4.57×10^{-3}	19
15	14	19	5	1.75×10^{-7}	2.82×10^{-4}	23
15	19	23	4	-1.03×10^{-7}	2.15×10^{-4}	24
15	25	29	4	-5.16×10^{-8}	3.52×10^{-4}	25
12	16	26	10	1.19×10^{-7}	3.49×10^{-4}	2
12	18	46	28	-3.97×10^{-4}	3.59×10^{-1}	3
12	12	13	1	-6.79×10^{-4}	2.06×10^{-5}	15
9	15	19	4	-4.10×10^{-4}	3.49×10^{-4}	1



The GPR optimizer yields good results. In most cases it is faster than the L-BFGS optimizer. Some qualitative differences can be observed. In the case of system 16 the GPR optimizer finds a different minimum than L-BFGS. One of these two different minima represents the reactant, the other one the product of this system. This happens because the optimization starts in the vicinity of the TS. System 3 and 9 show high RMSD values between the structures while their energy differences vanish: These are bimolecular reactions in which the minimum region of the separated molecules is flat using AM1. In the case of the molybdenum system with ID 26 and 27 the GPR optimizer finds a minimum that is higher in energy, and a little closer to the starting point. The minima look similar with slightly different torsions in the aliphatic groups. For the molybdenum system with ID 28 stricter convergence criteria are applied and the GPR optimizer is significantly faster. Towards the end of the optimization the convergence of L-BFGS is mostly hindered by larger predicted step sizes. All obtained minima look chemically plausible and (except for system 16) similar comparing the results of L-BFGS and the GPR optimizer.

For some of the test cases the Euclidean norm of the gradient versus the number of steps taken in the optimizer is shown. This is done for four of the biggest test cases from the Baker test set in figure 7.6 and the runs on the molybdenum system with IDs 26 and 27 (the ones with different starting points) in figure 7.7. The higher fluctuations of the GPR optimizer compared to the L-BFGS optimizer are due to the overshooting procedures, see section 7.2. They are not necessarily a sign of bad performance of the algorithm. The high overshooting is intentional and decreases the overall number of steps needed in almost all cases. The convergence criteria concerning the step size are usually fulfilled later than the ones concerning the gradient. This explains the large amount of steps that L-BFGS uses in the molybdenum system, although, the convergence criterion for the gradients are already met. Furthermore, the L-BFGS optimizer discards a lot of steps when the energy increases along the optimization, especially towards the end. This can be seen when the norm of the gradient stays constant for some time: No actual step is taken. The discarding of steps is not necessary for the GPR optimizer.

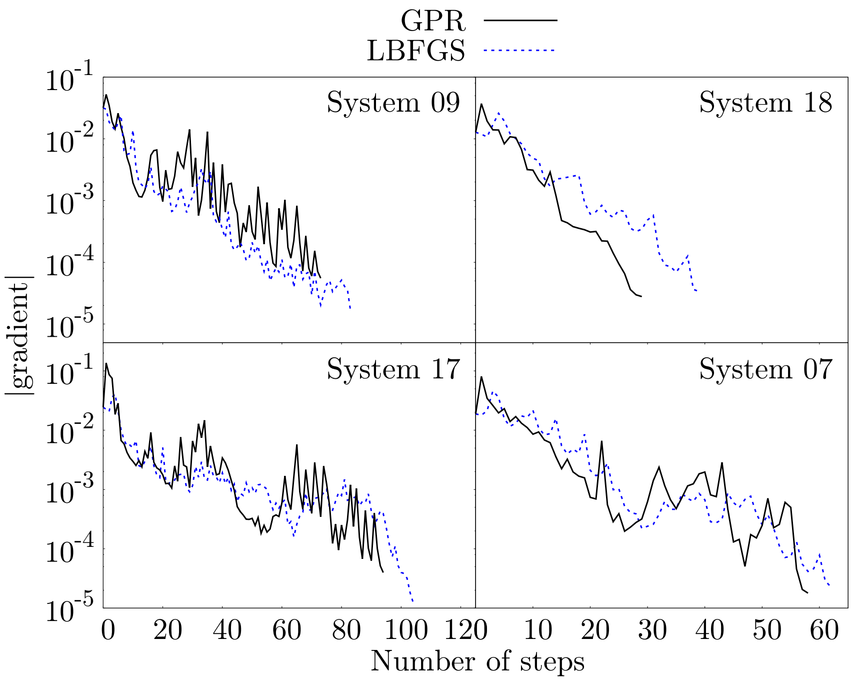
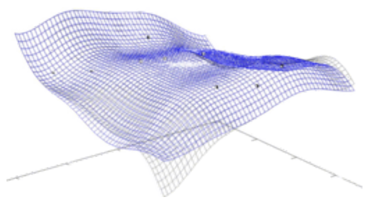


Figure 7.6: The Euclidean norm of the gradient with respect to the number of steps taken by the L-BFGS and the GPR optimizer in four of the biggest systems from the Baker test set.



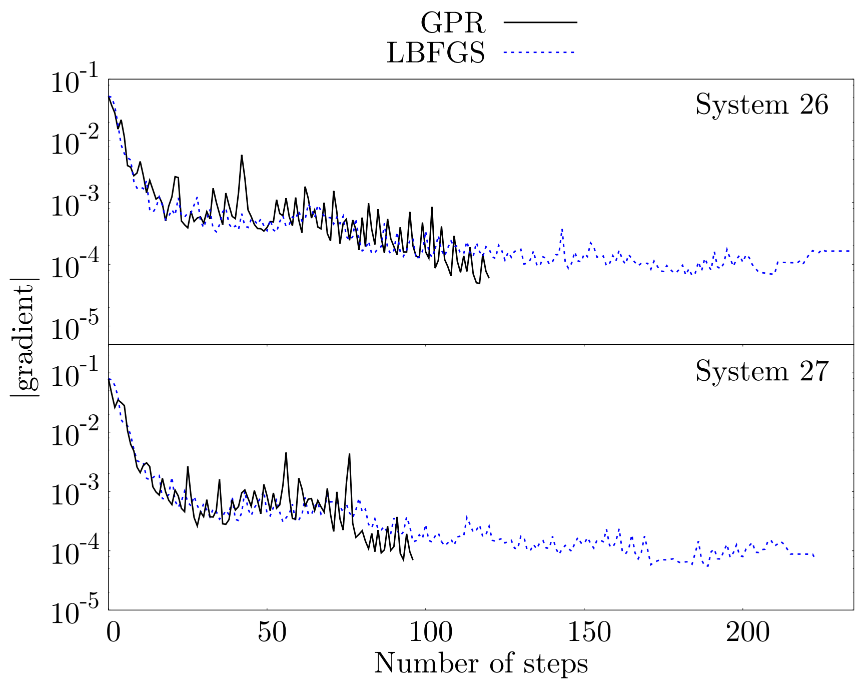
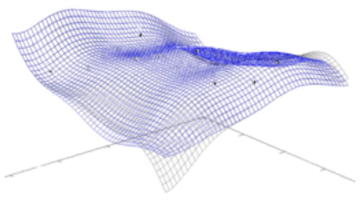


Figure 7.7: The Euclidean norm of the gradient with respect to the number of steps taken by the L-BFGS and the GPR optimizer in the molybdenum system with two different starting points.

7.3.1 Timing

The GPR optimizer is more demanding than classical optimizers in terms of computational power per step. The algorithm, including methods like the iterative Cholesky approach, see section 6.1, and the multi-level approach, see section 6.2, scales quadratically with the number of steps and dimensions of the system. Consequently, the optimizer takes more time per step than the L-BFGS optimizer. This can best be seen in our biggest test case, the molybdenum system (IDs 26 to 28): The GPR optimizer took about 40% of the overall computational time. The L-BFGS optimizer procedures take around 1%. The additional overhead is easily compensated by faster convergence in the test case. In the longest run with ID 28, the GPR optimization lasted around 77 minutes in total, while the L-BFGS optimization took around 105 minutes. The runs were all performed on an Intel i5-4570 quad-core CPU. If more accurate electronic structure methods are chosen than the DFT method used here and for calculating correspondingly smaller systems, only a negligible overhead from the GPR optimizer can be expected.



7.4 Discussion

The overall result of the presented benchmark indicates a good performance of the GPR optimizer. The big advantage of the GPR optimizer over traditional optimizers is that it can do comparably large steps without compromising the efficiency. If a step overshoots the minimum or even yields a higher energy structure than before, this generally improves the performance of the optimizer. It can easily use the obtained information from an overestimated step to build a more conservative step afterwards. This results in the possibility of doing large steps without compromising the efficiency of the optimizer. This speeds up the optimization procedure as can be seen in the largest test case.

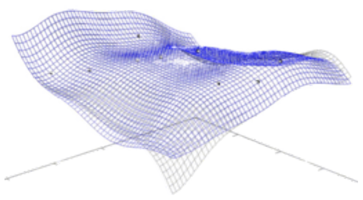
The scaling of the computational requirements for the optimizer is quadratic in the number of training points and the dimension of the system. The same holds for the memory demand, especially for the covariance matrix. Therefore, GPR is less promising in higher dimensional systems. However, since we can employ the multi-level approach to overcome memory problems to a certain extent, the optimizer is feasible for quite large systems. In the investigated systems the scaling was not a problem. Even in the 123-dimensional molybdenum system GPR performs well. Nevertheless, in general the GPR optimizer is only recommended for systems with less than a thousand atoms.

In summary, it can be said, that the presented optimizer outperforms the superlinearly converging L-BFGS optimizer on the presented test systems. In addition, using the step size limit as the only external parameter makes it easily applicable.

8 Saddle Point Search Algorithm

In this chapter it is explained in detail how one can improve the search for first-order saddle points. This chapter is largely based on the author's previous work on the respective topic [26].

Finding a first-order saddle point on the PES is particularly interesting because a reaction path from one minimum structure to another proceeds through such a saddle point. A *transition state* is formally a surface that separates the two minima. The lowest-energy point on this surface is a first-order saddle point and is called a *transition structure*. Often one also uses the term *transition state* for the *transition structure*. In the following it will be abbreviated with TS. Finding TSs is one of the most central tasks in computational chemistry. In transition state theory, it builds the basis for the calculation of classical reaction rates or even some non-classical reaction rates like in small-/large-curvature tunneling [31,51,78]. The most prominent algorithm to find saddle points is probably Newton's method. This method simply finds a point on the PES with a vanishing gradient. Therefore, the algorithm might find minima or higher-order saddle points. It is a second-order method which means that it needs Hessian evaluations in every optimization step. Another second-order method that converges explicitly to first-order saddle points is the partitioned rational function optimization (P-RFO) [7, 10], see section 2.3. Its results are usually very reliable but the need for Hessian calculations makes it often impracticable. So-called *minimum-mode following methods* [62] like the dimer method [38] or the Lanczos method [47, 83] only need gradient information to find TSs. This property makes them much faster and therefore often preferable to second-order methods like P-RFO [40]. They usually need more steps until convergence but compensate that by the fact that no Hessians have to be calculated. The basic idea of the mentioned optimizers are shortly explained in section 2. In



this thesis another alternative that is purely gradient-based was developed.

It uses, very similarly to the geometry optimizer of the previous chapter, a GPR-surrogate for the PES. Then, like in minimum-mode following methods, the algorithm converges the minimum-mode, see section 8.1.2. When the minimum-mode is converged P-RFO optimization is performed on this surrogate, see section 8.1.3. There are other algorithms to find TSs based on GPR than the one presented here. For example GPR was used to improve the nudged elastic band method (NEB) [39, 44, 45, 55] which is often used for TS searches. In the next chapter, an algorithm that is more comparable to the GPR-based NEB method will be shown. In this chapter, a surface walking algorithm is presented that does not need to create complete reaction paths but only iteratively finds a TS. It was also shown that one can optimize TSs using GPR with numerically calculated gradients (no analytical gradients are necessary) [70].

Finding a good first guess for the TS to start the optimization can often be challenging. Possible methods to find an initial guess are the image dependent pair potential (IDPP) [76] that was originally developed to give an initial guess for the NEB method. Another way of finding a starting guess based on GPR will be discussed in section 9.

Similar to the previous chapter, the algorithm is explained in detail, present some benchmarks, and discuss the properties of the algorithm. A few explanations are partially the same that are given in Ref. [26].

8.1 The Optimization Algorithm

Similar to P-RFO the TS search based on GPR, abbreviated GPRTS, uses a surrogate model to find a saddle point. Instead of a rational function, as in P-RFO, GPRTS uses GPR as a surrogate, called GPR-PES. The GPR surface is iteratively improved until the minimum-mode of the Hessian is converged. Then a full P-RFO search on the GPR-PES is performed, i.e. the full Hessian is calculated on the GPR-PES at every step of the P-RFO search. In contrast to a direct P-RFO search, no Hessian information of the electronic structure calculations are needed. Only energy and gradient information of the electronic structure calculations are used to build the GPR-PES. This will speed up the optimization overall. The convergence criteria are exactly the same as for the minimizer presented in the previous chapter, see section 7.1. They are only concerned with the length of the gradient and the step size and can be used for finding stationary points of any kind.

8.1.1 Parameters

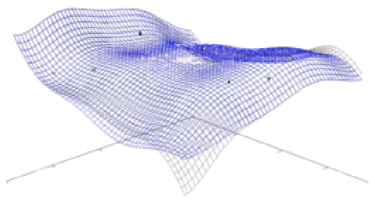
Also in this optimizer only the Matérn covariance function of equation (5.32) is used. The parameter σ_f is chosen to be 1. and l is set to $l = 20$. The noise parameters of equation (5.36) are set to $\sigma_\mathrm{e} = \sigma_\mathrm{g} = 10^{-7}$ again. The prior mean, E_prior, see equation (5.34), is set to the mean value of all training points.

$$E_\mathrm{prior}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} E_i \tag{8.1}$$

This can lead to confusion since it was stated before that the prior mean function is chosen before having knowledge of the training points. What is actually meant is that E_prior is chosen before the training points are used to fit the GP, i.e. to construct the linear system of equation (5.35).
Their remains one additional parameter, the maximum step size, s_max, which has to be specified by the user. It restricts the optimization to step sizes that are physically meaningful. Typically values between 0.1 and 1.0 are reasonable for s_max.

8.1.2 Converging the Minimum-Mode

Before a P-RFO optimization on the GPR-PES can be carried out, one has to have a reasonable representation of the Hessian at that point. The criterion by which it is decided whether the Hessian is adequately represented by the GPR-PES is that the minimum-mode, the eigenvector to the lowest eigenvalue of the Hessian, is converged. This is very similar to a dimer rotation in the dimer method. Starting at an initial guess for the TS, $\mathbf{x}_0^\mathrm{trans}$, the following procedure explains the convergence of the minimum-mode at that point. In later steps along the optimization procedure the minimum-mode is optimized at different points $\mathbf{x}_j^\mathrm{trans}$.



For minimum-mode optimizations at these points, $\mathbf{x}_0^{\text{trans}}$ can be substituted by $\mathbf{x}_j^{\text{trans}}$ in the following.

1. Calculate the energy and the gradient at the point $\mathbf{x}_0^{\text{trans}}$ and also at the point
$$\mathbf{x}_1^{\text{rot}} = \mathbf{x}_0^{\text{trans}} + \frac{\Delta}{|\mathbf{v}_0|}\mathbf{v}_0 \tag{8.2}$$
with $\mathbf{v}_0$ arbitrarily chosen. It works very well to choose $\mathbf{v}_0 = (1\ 1\ ...\ 1)^T$ which means just a translation by 1 of the whole molecule in all 3 spatial dimensions. The energy and gradient are the same for this translated structure. Therefore, the same energy as at $\mathbf{x}_0^{\text{trans}}$ is used for that training point and it is immediately included in the GPR scheme without additional electronic structure calculation. Only one energy and gradient calculation is then needed for this step. Choosing random numbers for the entries of $\mathbf{v}_0$ yielded worse results. The parameter Δ is set to $\Delta = 0.1$. Let $i = 1$.

2. Evaluate the Hessian $H_i(\mathbf{x}_0^{\text{trans}})$ of the GPR-PES at the point $\mathbf{x}_0^{\text{trans}}$.

3. Compute the smallest eigenvalue of H_i and the corresponding eigenvector, $\mathbf{v}_i$.

4. As soon as
$$\left| \frac{\mathbf{v}_i \cdot \mathbf{v}_{i-1}}{|\mathbf{v}_i| \cdot |\mathbf{v}_{i-1}|} \right| > 1 - \delta_{\text{rtol}} \tag{8.3}$$
the transition mode is assumed to be converged, this procedure is terminated, and $\mathbf{x}_0^{\text{trans}}$ is moved on the PES as described in section 8.1.3.

5. If the transition mode is not converged, calculate the energy and gradient at the point
$$\mathbf{x}_{i+1}^{\text{rot}} = \mathbf{x}_0^{\text{trans}} + \frac{\Delta}{|\mathbf{v}_i|}\mathbf{v}_i \tag{8.4}$$
and include the results to build a new GPR-PES. Increment i by one and go back to step 2.

8.1.3 Performing an Optimization Step on the GPR-PES

After following the procedure of section 8.1.2 the minimum mode is converged. In agreement with minimum-mode following methods, one can assume that enough Hessian information is available and one can now move on the PES to proceed to a TS. The points that are a result from movement on the PES are called $\mathbf{x}_j^{\text{trans}}$. They correspond to the midpoints in the procedure in section 8.1.2. A user-defined parameter, $s_{\max}$, is used to limit the steps from $\mathbf{x}_j^{\text{trans}}$ to the next point $\mathbf{x}_{j+1}^{\text{trans}}$. These steps can never be larger than $s_{\max}$.

Starting at a point $\mathbf{x}_j^{\text{trans}}$ the next point, $\mathbf{x}_{j+1}^{\text{trans}}$, is found as follows.

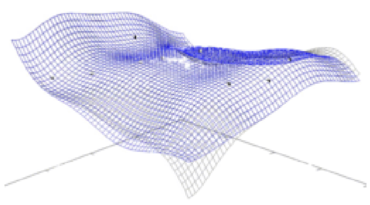
1. Find the saddle point on the GPR-PES using a P-RFO optimizer. This optimization on the GPR-PES is stopped if one of the following criteria is fulfilled.

    - The step size of the optimization is below $\delta_{\max(s)}/50$.
    - A negative eigenvalue is found (smaller than -10^{-10}) and the highest absolute value of all entries of the gradient on the GPR-PES drops below $\delta_{\max(g)}/100$.
    - The Euclidean distance between the currently estimated TS and the starting point of the P-RFO optimization is larger than $2s_{\max}$.

    If none of these are fulfilled after 100 P-RFO steps, a dimer translation is used to find a guess for the TS. For the dimer translation one uses the same convergence criteria as for P-RFO but the optimization is also stopped if the Euclidean distance between the currently estimated TS and the starting point is larger than $s_{\max}$.

2. Overshoot the estimated step, see section 8.1.4. The result of the overshooting is called $\mathbf{x}_{j+1}^{\text{trans}}$.

3. Calculate the energy and gradient at $\mathbf{x}_{j+1}^{\text{trans}}$ and include them in the GPR scheme to build a new GPR-PES.



If the point $\mathbf{x}_{j+1}^{\text{trans}}$ is not the converged solution, one converges the minimum-mode again as described in the previous section. These procedures are alternated until convergence is reached.

8.1.4 Overshooting

The overshooting procedures introduced in section 7.2.1 and also section 7.2.2 are exactly the same that are used for GPRTS here. The original step is overshot by multiplying the step with a certain scaling factor, see section 7.2.1. If the value of a coordinate along all optimization steps, $\mathbf{x}_i^{\text{trans}}$, changed monotonically for 20 steps in a row, the separate dimension overshooting is applied, see section 7.2.2. The resulting step is always limited by s_{max}. The overshooting procedures are applicable to minimization as well as to saddle point search. The argument still stays the same: Using a surrogate model like GPR allows for a very aggressive optimization by overshooting the guess for the saddle point. A step that is too far can still be included in the GPR-PES to improve prediction quality in following steps.

8.1.5 Starting Point from Minima

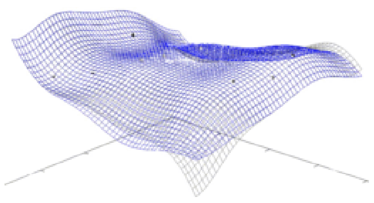
In the published article on GPRTS [26] a way of starting the TS search from known reactant and product is shown. The basic idea is that one can start from an initial approximation of the minimum energy path (MEP). In this case the initial path to start GPRTS should not do any energy calculations (in contrast to the algorithm presented in the next chapter which also optimizes MEPs). Assume that this path, called *initpath* in the following, is discretized with M images $\mathbf{x}_i^{\text{initpath}}$, $i = 1, ..., M$. GPRTS can take this path and searches for a maximum of the energy on it: The algorithm calculates energy and gradient of the real PES at the point $\mathbf{x}_j^{\text{initpath}}$ with $j = M/2$ for even and $j = (M-1)/2$ for odd M. This is the very first calculation of energies on the real PES. Then the scalar product beten the gradient γ_j at the point $\mathbf{x}_j^{\text{initpath}}$ and the vector to the next image on the *initpath* is calculated.

$$\beta = \gamma_j \cdot (\mathbf{x}_{j+1}^{\text{initpath}} - \mathbf{x}_j^{\text{initpath}}) \tag{8.5}$$

If $\beta > 0$, the energy and gradient at $\mathbf{x}_{j+1}^{\text{initpath}}$ is calculated. Otherwise the energy and gradient is calculated at $\mathbf{x}_{j-1}^{\text{IDPP}}$. In this way, the algorithm is likely to transit in the direction of the maximum in the energy. This is repeated until the direction along the NEB is changed and the algorithm would go back to a point $\mathbf{x}_{\text{best}}^{\text{initpath}}$ at which the energy is already known. The point $\mathbf{x}_{\text{best}}^{\text{initpath}}$ is most likely a good guess for the TS and is chosen as the starting point for GPRTS. Naturally, all energy and gradient information acquired in the described procedure are included in the construction of the GPR-PES. This is the same GPR-PES that is used and then improved by the following GPRTS procedure. This improves the quality of the GPR-PES compared to starting simply with a single energy calculation at $\mathbf{x}_{\text{best}}^{\text{initpath}}$. In the original article the *initpath* was constructed by optimizing a NEB in the IDPP. However, there is a further alternative approach based on differential geometry. The method is based on the idea to optimize a path that is a geodesic [85]. This approach was implemented in DL-FIND during the work for this thesis. Therefore, both possibilities to find the *initpath* are compared in the following. They are abbreviated with GPRTS/IDPP and GPRTS/geodesic. Note however, that the next chapter offers an alternative to that which includes real optimization of the MEP based on GPR.

8.2 Applications/Benchmarks

To benchmark GPRTS, 27 test systems were chosen. The first 25 are the ones in the test set by Baker [8]. The same starting points are chosen as for the benchmark of the minimizer, i.e. close to a TS on the Hartree–Fock level. Also the same electronic structure method as for the minimizer is used, namely AM1 [27]. The TSs obtained by GPRTS are depicted in figure 8.1. However, the shown TSs with IDs 10 and 15 are found by the dimer method. This is because the TS implied by the starting point in system 10 is only found by the dimer method and for system 15 GPRTS does not find a TS.



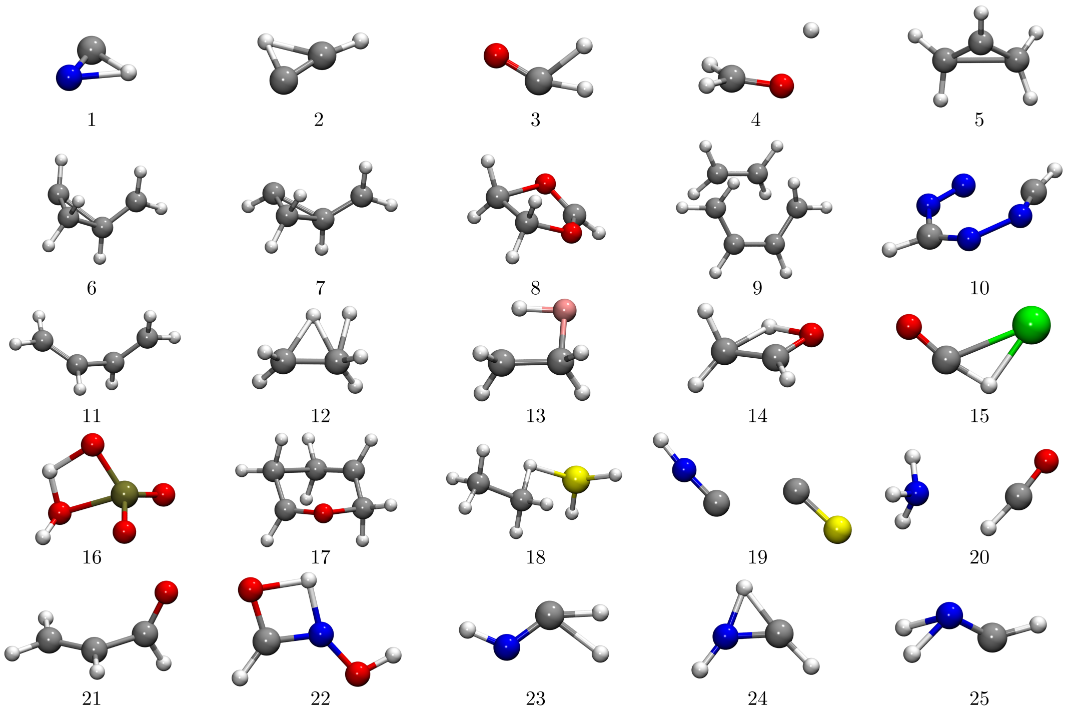
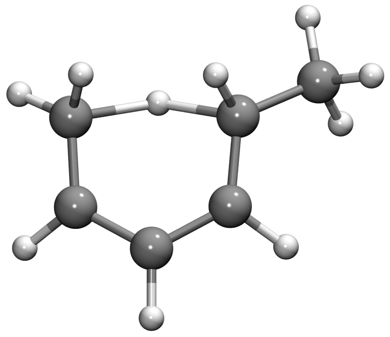
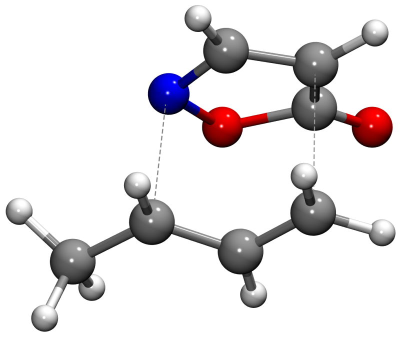


Figure 8.1: The TSs of the test systems suggested by Baker determined by the GPRTS method [8]. Only the TSs of the systems 10 and 15 are optimized by the dimer method.

Furthermore, two test systems on DFT level (BP86 functional [12, 60] in 6-31G* basis set [37]) are chosen: System 26 describes an intramolecular [1,5] H shift of 1,3(Z)-hexadiene to 2(E),4(Z)-hexadiene as investigated in Ref. 53, see figure 8.2a. System 27 describes an asymmetric allylation of a simple isoxazolinone as investigated in Ref. 68, see figure 8.2b. In the following two different benchmarks are presented. The first one comparing GPRTS to the dimer method and P-RFO, all starting from the same initial point. The second one comparing the performance of GPRTS with a starting guess from the NEB approximation in the IDPP (GPRTS/IDPP) and with a starting guess from the geodesic approach (GPRTS/geodesic).



(a) System 26, the [1,5] H shift of 1,3(Z)-hexadiene to 2(E),4(Z)-hexadiene [53]

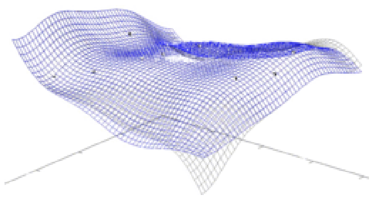


(b) System 27, an asymmetric allylation of a simple isoxazolinone. The dotted lines indicate the formed/broken bonds [68]

Figure 8.2: TSs found by GPRTS for test system 26 and 27.

8.2.1 Benchmarking GPRTS

All optimizations here are performed with a maximum step size of $s_{\max} = 0.3$ and a tolerance value $\delta = 3 \times 10^{-4}$. P-RFO uses the Bofill update mechanism [17] every 50 steps. The initial Hessians are built by central difference approximation via gradients in system 1 to 25. In systems 26 and 27 analytical Hessians are available. In the following *energy evaluation* automatically implies evaluation of the energy and the gradient. Table 8.1 shows the number of energy evaluations that GPRTS, the dimer method, and P-RFO require to yield a converged TS. The resulting TSs are compared by their respective energy differences and RMSD values. P-RFO uses 6 analytically calculated Hessians in system 27 that are not counted as energy evaluations.

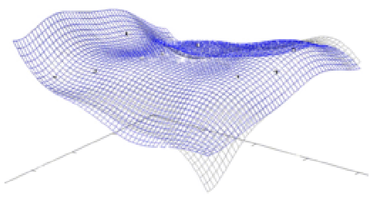


Some values of the obtained TSs in table 8.1 might need some explanation.

- System 3: P-RFO finds a structure in which H_2 is abstracted.
- System 6: The structure found by the dimer method has slightly different angles of the attached H atoms.
- System 7: P-RFO shows an opening of the ring.
- System 10: P-RFO finds a closed, symmetric ring structure. GPRTS finds a structure corresponding to separation of N_2. The dimer method finds the correct TS, shown in figure 8.1.
- System 11: The structure found by P-RFO is planar and has no negative eigenvalue.
- System 14: The dimer method finds the mirror image of the TS that the other methods find.
- System 18: The dimer method finds a structure in which both C atoms, the Si atom, and the transferred H atom are all in one plane. In the results of the other methods the dihedral angles are slightly different.
- System 20: P-RFO finds a slightly different distance and angle of the two separated parts.
- System 22: P-RFO finds a different angle of the attached OH group so that the structure is not planar as depicted in figure 8.1.
- System 26: P-RFO finds (after 924 steps) a different structure in which no H atom is transferred but the angles are changed: The distance of the H atom that should be transferred is almost 3 Å, the dihedral angle of the planes spanned by the two involved CH_2 structures is around 60°. The structure is too different from the desired TS to be considered as an acceptable TS for this rather simple reaction.

Table 8.1: A comparison of GPRTS to dimer and P-RFO in the test systems, sorted by the number of dimensions *d*. The number of required energy calculations until convergence is shown, as well as the energy difference (E_h) and RMSD values (Å) of the resulting structure compared to the structure found by GPRTS. Convergence problems of the electronic structure method are marked with *err*, non-convergence after 1000 energy evaluations is marked with *nc* (*not converged*), and if a run produces completely a unrealistic TS, it is marked with *false*.

	Energy evaluations			Energy differences		RMSD values		
d	GPRTS	dimer	P-RFO	dimer	P-RFO	dimer	P-RFO	ID
48	70	151	*false*	-6.4×10^{-6}		4.1×10^{-4}		26
57	88	231	262	-4.0×10^{-6}	-1.6×10^{-5}	5.5×10^{-3}	1.0×10^{-2}	27
48	42	171	*nc*	9.6×10^{-7}		2.0×10^{-3}		9
42	39	92	130	1.4×10^{-6}	1.4×10^{-7}	2.0×10^{-3}	1.3×10^{-3}	17
33	56	238	232	-7.5×10^{-4}	-7.2×10^{-4}	4.3×10^{-1}	3.6×10^{-1}	18
30	32	144	175	5.7×10^{-3}	-5.5×10^{-8}	1.2×10^{-1}	4.7×10^{-4}	6
30	38	101	507	3.2×10^{-7}	-2.1×10^{-1}	9.5×10^{-4}	5.8×10^{-1}	7
30	27	58	83	5.7×10^{-7}	-5.1×10^{-8}	1.0×10^{-3}	3.1×10^{-4}	8
30	67	*err*	*false*					11
24	24	59	85	-4.7×10^{-8}	-1.6×10^{-7}	5.5×10^{-4}	6.6×10^{-4}	5
24	23	169	151	-1.5×10^{-3}	-1.2×10^{-1}	1.0×10^{-1}	2.3×10^{-1}	10
24	17	56	68	4.7×10^{-7}	7.2×10^{-9}	1.3×10^{-3}	1.6×10^{-4}	12
24	30	92	73	1.6×10^{-7}	-1.2×10^{-9}	5.3×10^{-4}	5.3×10^{-5}	13
24	89	*err*	55		-2.8×10^{-7}		4.5×10^{-3}	21
21	35	72	80	3.4×10^{-7}	1.0×10^{-8}	4.2×10^{-1}	2.5×10^{-4}	14
21	28	70	79	-3.9×10^{-7}	-4.5×10^{-7}	1.0×10^{-3}	1.4×10^{-3}	16
21	23	*nc*	142		-1.1×10^{-7}		3.0×10^{-1}	20
21	19	43	174	5.6×10^{-7}	-1.6×10^{-3}	5.2×10^{-4}	2.7×10^{-1}	22
15	33	48	66	-9.7×10^{-7}	-1.1×10^{-9}	2.6×10^{-3}	4.4×10^{-5}	4
15	21	75	44	-2.0×10^{-10}	3.4×10^{-6}	3.6×10^{-5}	7.7×10^{-3}	19
15	16	40	56	2.9×10^{-7}	-2.2×10^{-8}	1.0×10^{-3}	2.8×10^{-4}	23
15	18	56	114	-1.0×10^{-8}	-1.8×10^{-8}	4.2×10^{-4}	3.9×10^{-4}	24
15	22	42	59	4.1×10^{-7}	1.5×10^{-7}	1.3×10^{-3}	1.1×10^{-3}	25
12	14	34	35	4.4×10^{-9}	6.4×10^{-9}	8.7×10^{-5}	7.3×10^{-5}	2
12	19	31	127	2.1×10^{-7}	-9.4×10^{-2}	4.0×10^{-4}	1.2	3
12	*err*	59	*err*					15
9	19	39	30	0.0	-1.8×10^{-9}	4.0×10^{-6}	4.0×10^{-5}	1



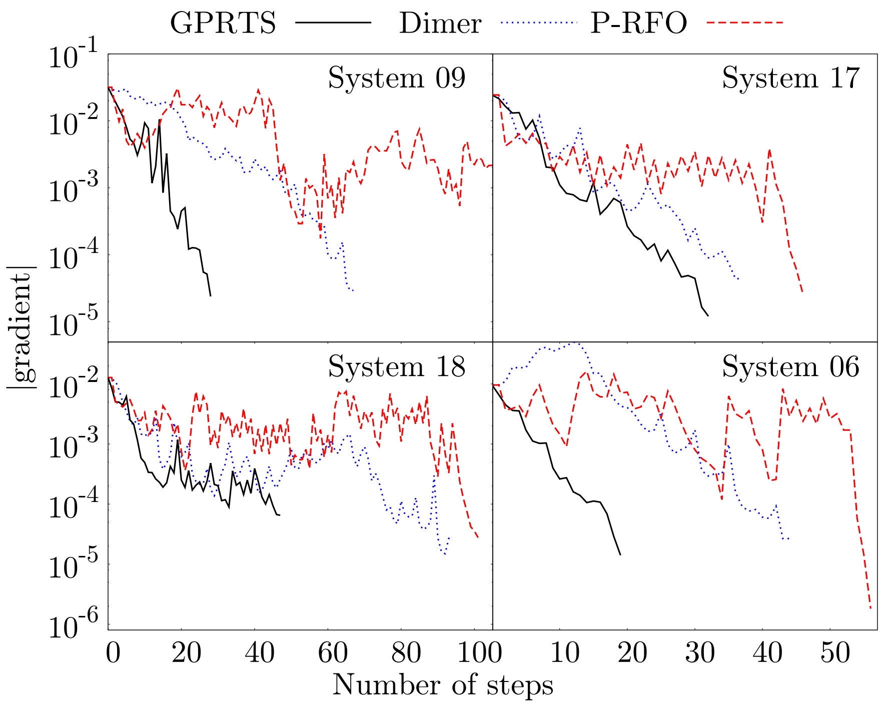


Figure 8.3: The Euclidean norm of the gradient with respect to the number of steps taken on the PES for the three optimizers in the four biggest systems in the Baker test set. Steps that are needed to optimize the minimum mode or calculate the Hessian (in the case of P-RFO) are not included. The plot for System 09 is truncated since P-RFO does not converge.

One can clearly see a better performance of the GPRTS method compared to the other two methods. The number of required energy calculations is more than halved on average. Overall the algorithm also seems to be more stable. It has the fewest convergence problems and converges to the correct TS in more cases than the other methods. To emphasize the fast convergence of the GPRTS method, the absolute value of the gradient is plotted against the number of steps taken on the PES in figure 8.3 for four of the largest test systems in the Baker test set and in figure 8.4 for the DFT test systems. Thereby, only the translational steps on the PES are shown. This means that all energy evaluations needed for converging the minimum mode or calculating a Hessian are not shown.

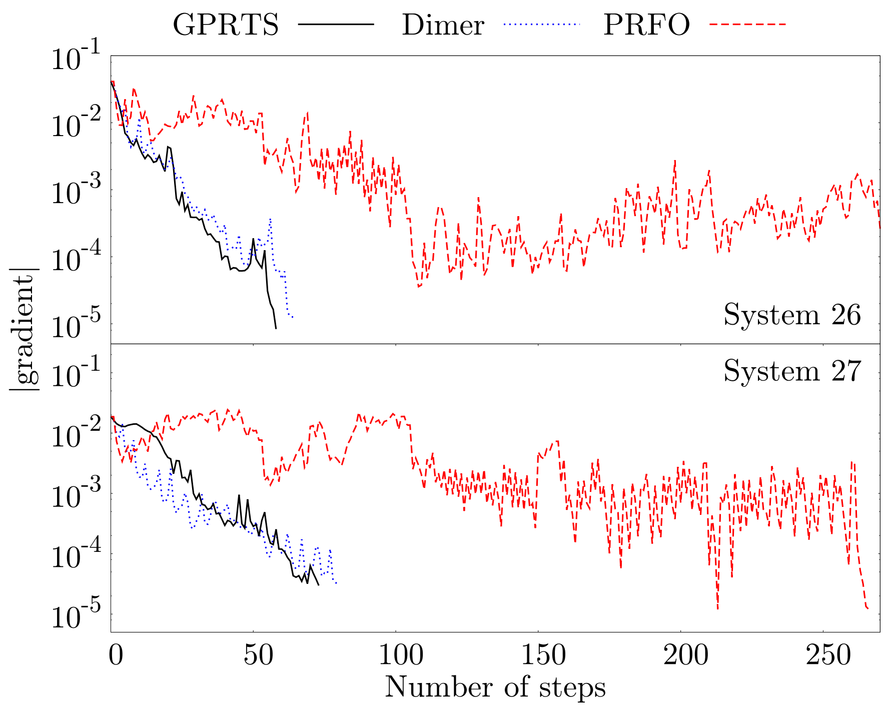
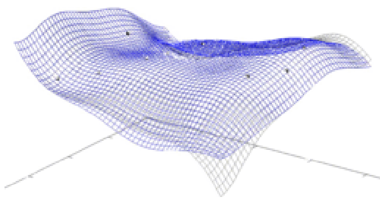


Figure 8.4: The Euclidean norm of the gradient with respect to the number of steps taken on the PES for the three optimizers in the test systems 26 and 27. Steps that are needed to optimize the minimum mode or calculate the Hessian (in the case of P-RFO) are not included. The plot for System 26 is truncated since P-RFO does not converge correctly.



8.2.2 Benchmarking Methods for the Initial Point

In this section, the NEB approximation in the IDPP [76] and the geodesic approach [85] are compared with respect to their capability to find an initial TS for GPRTS from the respective minima. For this benchmark, the correctly optimized TS for all the test systems, see Fig. 8.1, 8.2a, and 8.2b, are used to start *intrinsic reaction path* (IRC) searches to find the two minimum structures that are connected by the TS. An IRC is the steepest-descent MEP in mass-weighted Cartesians. For more information on the IRC algorithm, be referred to the publication on the respective implementation, see Ref. 54. The resulting minimum structures are used to optimize a NEB in the IDPP and to optimize a geodesic, each with 10 images. The resulting images are used in the procedure described in section 8.1.5 to find a TS. The results of this procedure are compared to the correctly converged TSs for the test systems found in section 8.2.1, i.e. all structures are obtained with GPRTS, except for the structures in system 10 and 15 which are obtained by the dimer method. Table 8.2 shows the number of energy evaluations that the GPRTS method requires using the different approaches to find the starting point.

In some cases the results of GPRTS showed wrong results/different structures than desired when started from the geodesic approach. This is the case for system 3, 12, 13, and 16. When starting from the NEB approximation in the IDPP, system 21 showed a wrong structure. Some high values of the RMSD need additional explanation.

- System 10: The angles in the structures are a little off, but the structures look very similar to the reference structures. Tighter convergence criteria eliminate this difference.
- System 11: The obtained results are just the mirror image of the reference structure.
- System 18: The dihedral angles in the obtained structure is different to the reference structure. This was already the case when comparing the results of the dimer method and P-RFO to the GPRTS method.
- System 22: The obtained structures are the same as the one found by P-RFO in the previous benchmark: The structures have a different angle of the attached OH group so that the structure is not planar as depicted in figure 8.1.
- System 25: Both methods find a planar structure in contrast to the reference structure.

Compared to the previous benchmark (GPRTS) the runs also converge in system 15. One can conclude that stability might be increased by including additional training points, e.g. the ones from the initial path. The estimated initial guess for GPRTS/IDPP and GPRTS/geodesic does not seem to be as adequate as the ones given in the previous benchmark and the optimization takes a bit longer. But note that only the minimum structures are needed to converge the system to a TS. Comparing GPRTS/IDPP to GPRTS/geodesic it seems that the geodesic approach does not work equally well. This could happen by chance but also as a result on fairly large steps that the used geodesic implementation yields along the geodesic path in some cases. There is however, one system (ID 21) in which it works better than the GPRTS/IDPP approach.

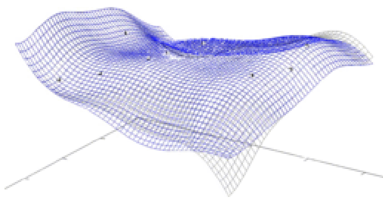


Table 8.2: A comparison of GPRTS to the dimer method and P-RFO. The number of steps until convergence is shown as well as the energy differences (Hartree) and RMSD values (Ång) comparing the respective structures to the structure obtained by GPRTS. Convergence problems of the AM1 method are marked with *err*, non-convergence after 1000 energy evaluations is marked with *nc* for *not converged*, and if a run produces a unrealistic TS, it is marked with *false*.

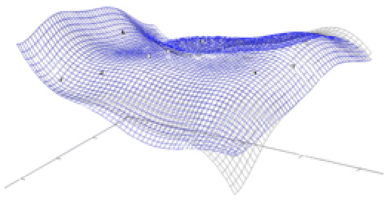
	Energy evaluations			Energy differences		RMSD values		
d	GPRTS	GPRTS/ IDPP	GPRTS/ geodesic	GPRTS/ IDPP	GPRTS/ geodesic	GPRTS/ IDPP	GPRTS/ geodesic	ID
123	70	57	56	-2.0×10^{-7}	-6.0×10^{-7}	2.0×10^{-3}	3.1×10^{-4}	26
123	88	131	57	-1.2×10^{-5}	-3.2×10^{-5}	3.9×10^{-3}	2.5×10^{-3}	27
48	42	45	41	-7.0×10^{-9}	-2.5×10^{-8}	4.3×10^{-4}	7.9×10^{-4}	9
42	39	90	54	4.8×10^{-7}	-3.4×10^{-8}	1.2×10^{-3}	3.5×10^{-4}	17
33	56	42	39	-4.0×10^{-4}	-5.9×10^{-4}	2.6×10^{-1}	1.7×10^{-1}	18
30	32	53	47	-4.7×10^{-8}	-2.5×10^{-8}	2.4×10^{-4}	5.4×10^{-4}	6
30	38	42	37	8.7×10^{-9}	1.3×10^{-8}	2.6×10^{-4}	1.6×10^{-4}	7
30	27	37	34	-1.9×10^{-8}	-3.5×10^{-8}	2.0×10^{-4}	3.0×10^{-4}	8
30	67	44	44	-5.1×10^{-9}	1.3×10^{-9}	1.4	1.4	11
24	24	26	24	-1.7×10^{-7}	-4.6×10^{-8}	1.5×10^{-4}	4.8×10^{-4}	5
24	23	35	35	-1.5×10^{-3}	-1.4×10^{-3}	2.0×10^{-1}	2.0×10^{-1}	10
24	17	90	*false*	-1.4×10^{-8}	1.7×10^{-2}	2.0×10^{-5}	5.9×10^{-1}	12
24	30	33	*false*	7.1×10^{-9}	-1.3×10^{-1}	1.0×10^{-4}	1.3	13
24	89	*false*	84	-7.8×10^{-9}	-8.0×10^{-9}	1.0	1.0×10^{-4}	21
21	35	21	21	3.0×10^{-7}	1.9×10^{-8}	8.8×10^{-4}	1.8×10^{-4}	14
21	28	28	*false*	-4.5×10^{-7}	-1.4×10^{-2}	6.5×10^{-5}	4.7×10^{-1}	16
21	23	45	37	4.0×10^{-10}	6.8×10^{-9}	9.2×10^{-5}	4.6×10^{-2}	20
21	19	32	30	-1.5×10^{-3}	-1.5×10^{-3}	2.7×10^{-1}	2.7×10^{-1}	22
15	33	73	64	8.0×10^{-2}	8.0×10^{-2}	1.4×10^{-4}	3.9×10^{-5}	4
15	21	46	32	0.0	1.8×10^{-7}	4.1×10^{-5}	2.1×10^{-3}	19
15	16	31	55	-4.4×10^{-8}	-1.2×10^{-8}	8.3×10^{-5}	2.1×10^{-4}	23
15	18	15	17	-1.8×10^{-8}	-1.8×10^{-8}	3.7×10^{-4}	3.6×10^{-4}	24
15	22	33	50	-2.4×10^{-2}	-2.4×10^{-2}	4.2×10^{-1}	4.2×10^{-1}	25
12	14	19	28	7.1×10^{-9}	1.0×10^{-8}	8.4×10^{-5}	9.3×10^{-5}	2
12	19	43	*false*	1.1×10^{-10}	-9.5×10^{-2}	4.9×10^{-6}	6.2×10^{-1}	3
12	*err*	29	33	8.3×10^{-10}	1.3×10^{-10}	4.0×10^{-5}	3.9×10^{-5}	15
9	19	13	24	0.0	0.0	5.0×10^{-6}	6.3×10^{-6}	1

8.2.2.1 Timing

With the drastic decrease of required energy evaluations the overall time that the algorithm needs to converge is reduced as well. To show that it is best to consider the DFT test cases. P-RFO converged only for system 27 to the correct TS and needed ~ 36 minutes. For the systems with ID 26/27, the dimer optimizations presented above took a little over 15/18 minutes while GPRTS took less than 8/9 minutes. The GPRTS algorithm itself took 3%/16% of that time. The optimizations of GPRTS/IDPP and GPRTS/geodesic were very similar in speed. They all took around 5 minutes with the optimizer itself taking about $2-6\%$ of that time. For system 27 however, GPRTS/geodesic took nearly 23 minutes with the optimizer itself taking around 56% of that time. The reason for that behavior is that the P-RFO method on the GPR-PES did not converge all the time. As explained above, a dimer translation on the GPR-PES is performed instead. The large number of unnecessary P-RFO steps on the GPR-PES is the main reason for the bad timings in this run. Probably the starting guess suggested by using the geodesic approach is not suitable.

8.3 Discussion

The scaling of the computational requirement of the algorithm is quadratic in the number of training points and the number of dimensions for constructing the GPR-PES. The evaluation of the Hessians on the GPR-PES, however, scales linearly with the number of training points and cubically with the number of dimensions in the system. The scaling of the memory requirements of the algorithm is quadratic with respect to the number of dimensions and the number of steps. Therefore, the algorithm is only advisable for smaller systems: From personal experience the author suggests to use the algorithm for systems up to 300 dimensions. It is possible to handle larger systems in combination with the multi-level approach (which is done automatically) but the performance might decrease. See section 6.2 for details on the multi-level approach. With the multi-level approach systems with 1000 dimensions might be possible to handle. For high-precision optimizations the algorithm is very likely to outperform traditional TS optimizers.



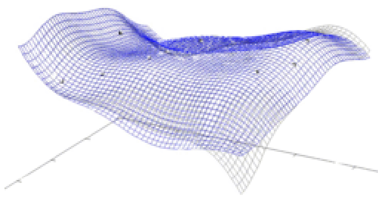
In this algorithm one can see that different floating-point models of the compiler can lead to slightly varying results. In the presented test cases different floating-point models lead to variations in the number of required energy evaluations, mostly less than 5, always less than 20, for better or worse. Note that this can also be obtained with P-RFO. Therefore, the high sensitivity of GPRTS might not only be attributed to the GPR part of the algorithm but also to the P-RFO optimization performed on the GPR-PES. Further numerically sensitive steps in GPRTS are the solution of the linear system, the evaluation of the GPR-PES, and the diagonalization of the Hessian.

Overall, GPRTS seems to be clearly superior to the traditional optimizers in the presented test cases. The ability to overshoot TSs without compromising the performance of the optimizer is probably the clearest advantage. Furthermore, the Hessian information on the GPR-PES seem to be accurate enough and probably better than traditional update mechanisms. In fact this is tested in a later chapter in the thesis at hand. The usage of MEP estimates to start the optimizations might be a good idea to get a converging TS in some cases where traditional methods fail. On the other hand, if good estimates of the TS are present, e.g. from calculations with lower level of electronic structure theory, the standard GPRTS method is probably faster.

9 Minimum Energy Path Optimization Algorithm

In this chapter, a new method to optimize minimum energy paths (MEPs) is presented. The described algorithm is based on the author's previously published work [24]. If one wants to describe the movement of atoms in a chemical reaction, the most common way to do that is by a MEP. It provides the means to understanding reaction mechanisms but often it is only optimized to find the TS connecting two known minima. The most prominent method to optimize a MEP is the nudged elastic band (NEB) method [39, 55]. Further approaches are the string methods [28, 29] which are very similar to NEB [75]. There exist other methods for MEP optimizations [6, 49, 63]. But since the NEB method is by far the most popular method the focus lies on a comparison to NEB.

In this part of the thesis an alternative approach is presented that uses a GPR-PES like the two optimizers presented before in this thesis. It is noteworthy that the NEB method itself was significantly improved by a similar approach with GPR surrogates [44,45]. Similar to their approach, the algorithm presented in this thesis uses a GPR surrogate and makes use of the statistical properties of GPR. But instead of performing a NEB optimization on the GPR-PES an alternative approach is used that does not introduce artificial forces like the spring forces in NEB. The presented approach minimizes energies and forces perpendicular to the path. The curve parameters are thereby directly optimized. This is similar to an approach by Vaucher and Reiher [79]. In contrast to their work the presented algorithm also uses the perpendicular forces on the path for optimization, exploits the GPR-PES, and uses GPR itself to interpolate a curve not B-splines. All these methods, including the one presented here, require the construction of an initial path from which



the optimization can be started. Choosing a path in which internuclear distances vary linearly is often referred to as Linear Synchronous Transit (LST) [35]. In the previous chapter the image dependent pair potential (IDPP) was mentioned in which a NEB optimization can be performed [76] and the geodesic approach [85]. All three methods are used to benchmark the presented algorithm. Some of the explanations are similar to the ones given in the author's previous work on MEP optimization [24].

A few words on the structure of this section. First two concepts are introduced that are needed for the optimization procedure. These are the regression of a curve with GPR in section 9.1 and a possibility to make the control points of this path equidistant in section 9.2. Then the principle of the optimizer are explained in detail. Some benchmarks are presented at the end.

9.1 Curve Regression Using GPR

In this section it is explained how one can interpolate a curve $\mathcal{C}(t)$ with a parametrization $t = [0, 1]$ mapping to $\mathbb{R}^d$. The N points that are used to regress the curve are called *control points* of the curve and denoted with $\mathbf{x}_n$, $n = 1, ..., N$. They are not necessarily equidistant in $\mathbb{R}^d$. For the presented algorithm the endpoints of the curve, $\mathcal{C}(t = 0)$ and $\mathcal{C}(t = 1)$, are assumed to be minima on the PES representing reactants and products, respectively. They shall stay constant when optimizing the MEP.
To regress a d dimensional curve one can build d separate GPs, one for each dimension. Each of these one-dimensional GPs is constructed with N training points $(t_{\tilde{l},n}, x_n^{\tilde{l}})$ where n enumerates the control points and $\tilde{l}$ the dimensions, respectively. As before, superscripts refer to the components of a vector in the respective dimension and subscripts are indices to enumerate points. Since $\tilde{l}$ runs over all dimensions of the system, it is also an index for the one-dimensional GPs that are used for curve regression. In this paragraph, the samples $t_{\tilde{l},n}$ are assumed to be equidistant and the same for every dimension $t_{\tilde{l},n} = n\Delta t$. In the next paragraph this will be changed to make the points equidistant in Euclidean space. Therefore, the index $\tilde{l}$ is not dropped here. To regress a curve with N training points one

builds a GP for every dimension $\tilde{l} = 1, ..., d$.

$$x^{\tilde{l}}(t) = \sum_{\tilde{k}=1}^{N} w_{\tilde{l},\tilde{k}} k(t, t_{\tilde{l},\tilde{k}}) + x^{\tilde{l}}_{\text{prior}}(t) \tag{9.1}$$

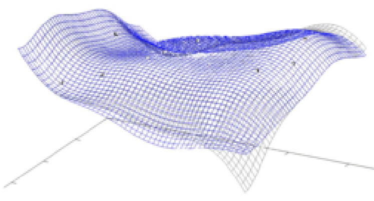
The prior $x^{\tilde{l}}_{\text{prior}}(t)$ shall be a linear interpolation of the first and the last training point.

$$x^{\tilde{l}}_{\text{prior}}(t) = \frac{x^{\tilde{l}}_N - x^{\tilde{l}}_1}{t_{\tilde{l},N} - t_{\tilde{l},1}} t = (x^{\tilde{l}}_N - x^{\tilde{l}}_1)t \tag{9.2}$$

with $t_{\tilde{l},N} - t_{\tilde{l},1} = 1$ by definition of the curve. This allows the GPR to learn only the error of the linear interpolation which is an easier task than directly learning the values. Non-intersection and non-degeneracy of the resulting parametrization is not a problem in practice. To regress curves with GPR the Matérn covariance function is used with a length scale parameter of $l = 0.5$. This means, since the path is only parameterized from 0 to 1, that every point has a strong influence on a large part of the path. A high value of l leads to a smooth regression of the path. This is what one usually wants to guarantee. Lower values of l can result in wiggly curves. On the other hand, a higher value of l requires to choose a higher value of the noise parameter, σ_{e}, for the regression of the path. This is necessary to guarantee numerical stability when solving the linear equations in GPR. A value of $\sigma_{\text{e,c}} = 2 \times 10^{-4}$ a_0 for the noise is chosen in the curve regression. Note that no gradient information is used for curve regression here. This facilitates the GPR equations since the terms considering the gradients at the training points can be ignored, see for example in Ref. [25,67]. The linear system is reduced to N dimensions which is usually very cheap to solve.

9.2 Securing the Equidistancy of Points on a Regressed Curve

To get a good representation of the MEP at every position it is often necessary to have the control points equidistantly distributed in Cartesian coordinate space. To achieve this, the problem is formulated as an optimization problem. Let there be N



points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \in \mathbb{R}^d$ at positions $t_1, t_2, ..., t_N$, respectively, while $t_{i+1} - t_i = \Delta t$ is the same for all i.

The Euclidean distances between the points is abbreviated with

$$\delta_i := \|\mathbf{x}_{i+1} - \mathbf{x}_i\| \tag{9.3}$$

for all $i = 1, ..., N-1$.

The loss function L for the optimization problem shall be the following.

$$L = \sum_{i=1}^{N-2} |\delta_{i+1} - \delta_i|^2 \tag{9.4}$$

The endpoints $\mathbf{x}_1$ and $\mathbf{x}_N$ shall be fixed. Therefore, L is minimized with respect to all t_j with $j = 2, ..., N-1$. On that account, one needs the derivative of L with respect to an arbitrary t_j with $j = 2, ..., N-1$. Changing t_j also implies a change of x_j, hence implying a change of δ_{j-1} and δ_j.

$$\begin{aligned}\frac{\partial L}{\partial t_j} = & -2(\delta_{j+1} - \delta_j)\frac{\partial \delta_j}{\partial t_j} + \\ & 2(\delta_j - \delta_{j-1})\left[\frac{\partial \delta_j}{\partial t_j} - \frac{\partial \delta_{j-1}}{\partial t_j}\right] + \\ & 2(\delta_{j-1} - \delta_{j-2})\frac{\partial \delta_{j-1}}{\partial t_j}\end{aligned} \tag{9.5}$$

For the derivative with respect to t_2 the last term (orange) is not present. For the derivative with respect to t_{N-1} the first term (blue) is not present. The derivatives $\frac{\partial \delta_j}{\partial t_j}$ and $\frac{\partial \delta_{j-1}}{\partial t_j}$ are still to be calculated. Let $x_j^1, x_j^2, ..., x_j^d$ be the components of the vector $\mathbf{x}_j$.

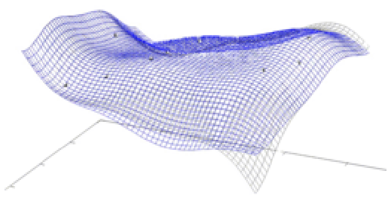
$$\frac{\partial \delta_i}{\partial t_j} = \sum_{l=1}^{d} \frac{\partial \delta_i}{\partial x_j^l}\frac{\partial x_j^l}{\partial t_j} \tag{9.6}$$

The expression $\frac{\partial x_j^l}{\partial t_j}$ is simply the derivative along the curve in dimension l at the point where the parametrization variable takes on the value t_j. This term can be obtained by evaluating the derivative of the regressed curve which is fairly easy when using GPR. One obtains the following derivatives.

$$\frac{\partial \delta_j}{\partial t_j} = \sum_{l=1}^{d} \left[- \frac{x_{j+1}^l - x_j^l}{\|\mathbf{x}_{j+1} - \mathbf{x}_j\|} \right] \frac{\partial x_j^l}{\partial t_j} \tag{9.7}$$

$$\frac{\partial \delta_{j-1}}{\partial t_j} = \sum_{l=1}^{d} \left[\frac{x_j^l - x_{j-1}^l}{\|\mathbf{x}_j - \mathbf{x}_{j-1}\|} \right] \frac{\partial x_j^l}{\partial t_j} \tag{9.8}$$

With equation (9.5) one can build a $N-2$ dimensional gradient $\mathbf{dL}$ of the loss function. Its entries are the derivatives of L with respect to all t_j with $j = 2, ..., N-1$. With this a $N-2$-dimensional, gradient-based optimization problem is formulated that is equivalent to making the control points on a curve equidistant. It can be solved by standard gradient-based optimizers like L-BFGS [18,32–34,48, 57,73]. However, some small adaptations are necessary to stabilize the procedure. To avoid crossing of points on the curve, the step size of the optimization for t_i is limited to half the distance $t_{i+1}-t_i$ if t_i is increasing and to half the distance t_i-t_{i-1} if t_i is decreasing. Usually, one does not require a perfectly equidistant distribution of the points. Therefore, a rather soft convergence criteria for the optimization can be chosen. Good results can be obtained by choosing $\max_i(dL^i) < 5 \times 10^{-4}$ as a convergence criterion with dL^i being the entries in dimension i of $\mathbf{dL}$.



9.3 The Optimization Algorithm

The theoretical basis of the MEP search presented in this thesis is the minimization of two loss functions by optimizing a path that is constructed via GPR. To optimize this path one uses a GPR-based surrogate for the PES. First, the two loss functions are introduced that can be used to optimize a MEP before explaining details on the optimization itself.

9.3.1 Loss Functions to Optimize a MEP

In order to do derive the loss functions some necessary, mathematical expressions are presented first. These expressions are colored in red. The norm $\|\cdot\|$ is implicitly assumed to be the 2-norm (Euclidean norm). The derivative of the norm can be written as follows.

$$\frac{\partial}{\partial x}\|\mathbf{f}(x)\| = \frac{\sum_{j=1}^{d} f_j(x)\frac{\partial}{\partial x}f_j(x)}{\|\mathbf{f}(x)\|} \tag{9.9}$$

One can write the GPR scheme used to regress a curve at arbitrary points $\mathbf{x}_i = \mathbf{x}(t_i)$ on the curve as a function of the weights.

$$x_i^{\tilde{l}}(w_\alpha) = \sum_{\tilde{k}=1}^{N} w_{\tilde{l},\tilde{k}}\,k(t_i, t_{\tilde{l},\tilde{k}}) \tag{9.10}$$

One can obtain the following equations.

$$\frac{\partial x_i^{\tilde{l}}}{\partial w_{l,k}} = k(t_i, t_{\tilde{l},k})\delta_{l,\tilde{l}} \tag{9.11}$$

$$\frac{\partial^2 x_i^{\tilde{l}}}{\partial w_{l,k}\partial t_i} = \frac{\partial k(t_i, t_{\tilde{l},k})}{\partial t_i}\delta_{l,\tilde{l}} \tag{9.12}$$

$$\frac{\partial}{\partial w_{l,k}}\left\|\frac{\partial \mathbf{x}_i}{\partial t_i}\right\| = \sum_{\tilde{l}=1}^{d}\frac{\frac{\partial x_i^{\tilde{l}}}{\partial t_i}\frac{\partial^2 x_i^{\tilde{l}}}{\partial w_{l,k}\partial t_i}}{\left\|\frac{\partial \mathbf{x}_i}{\partial t_i}\right\|} = \sum_{\tilde{l}=1}^{d}\frac{\frac{\partial x_i^{\tilde{l}}}{\partial t_i}\frac{\partial k(t_i,t_{\tilde{l},k})}{\partial t_i}\delta_{l,\tilde{l}}}{\left\|\frac{\partial \mathbf{x}_i}{\partial t_i}\right\|} \tag{9.13}$$

And for the derivative of the point $\mathbf{x}_i = \mathbf{x}(t_i)$ with respect to t_i one obtains the following.

$$\frac{\partial x_i^{\tilde{l}}}{\partial t_i} = \sum_{\tilde{k}=1}^{N} w_{\tilde{l},\tilde{k}} \frac{\partial k(t_i, t_{\tilde{l},\tilde{k}})}{\partial t_i} \tag{9.14}$$

Note that the points in the parameter space $t_{\tilde{l},k}$ that correspond to the control points $\mathbf{x}_k$ of the curve are denoted with two indices, with $\tilde{l} = 1, ..., d$ for the dimension and $k = 1, ..., N$ for the control point.
The t_i with $i = 0, ..., M + 1$ in the following are another possible discretization of the curve and do not have to be the same as the $t_{\tilde{l},k}$. The same applies to the respective points on the path, $\mathbf{x}_i := \mathbf{x}(t_i)$. They do not have to be the same as the control points.

Let $\mathbf{w}_\alpha \in \mathbb{R}^{d \cdot N}$ be the vector that contains all weights of the GPs used for the curve regression. Further, let $\mathbf{x}_i = \mathbf{x}(t_i)$ be an arbitrary point on the curve with its components $x_i^{\tilde{l}}$ as a function of the weights.

$$x_i^{\tilde{l}}(\mathbf{w}_\alpha) = \sum_{\tilde{k}=1}^{N} w_{\tilde{l},\tilde{k}} k(t_i, t_{\tilde{l},\tilde{k}}) \tag{9.15}$$

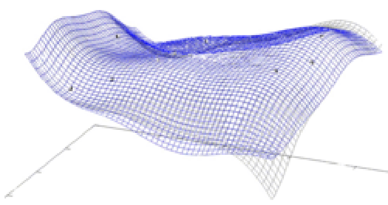
9.3.1.1 Loss Function Using an Energy Criterion

One possible loss function, $\mathcal{L}_\mathrm{E}$, based on energy values along the curve, is the following.

$$\begin{aligned} \mathcal{L}_\mathrm{E}^\mathrm{exact} &= \int_\mathcal{C} E^2(\mathcal{C}(t)) dL \\ &= \int_{t=0}^{1} E^2(\mathcal{C}(t)) \left\| \frac{\partial \mathcal{C}(t)}{\partial t} \right\| dt \end{aligned} \tag{9.16}$$

The symbol dL can intuitively be interpreted as an elementary arc length of the curve $\mathcal{C}$. Using a discretization of $M + 2$ points on the path, $\mathbf{x}_i = \mathbf{x}(t_i)$, $i = 0, ..., M + 1$, this can be approximated by

$$\mathcal{L}_\mathrm{E} = \sum_{i=0}^{M+1} E^2(\mathbf{x}_i) \left\| \frac{\partial \mathbf{x}_i}{\partial t_i} \right\| \Delta t \tag{9.17}$$



where $\frac{\partial \mathbf{x}_i}{\partial t_i}$ is short for $\frac{\partial \mathbf{x}(t)}{\partial t}|_{t=t_i}$, the derivative along the curve at the discretization point $\mathbf{x}_i$. The values $E(\mathbf{x}_i)$, $\frac{\partial \mathbf{x}_i}{\partial t_i}$, and Δt can easily be calculated. The gradient of the loss functions in curve-parameter space (the space of the weights for the GPs) is composed of the following elements. Note that $x_0 \equiv x(t_0)$ and $x_{M+1} \equiv x(t_{M+1})$ are assumed to be constant.

$$\begin{aligned}\frac{\partial \mathcal{L}_\mathrm{E}}{\partial w_{l,k}} &= \sum_{i=1}^{M} \left[2E(\mathbf{x}_i) \sum_{\tilde{l}=1}^{d} \frac{\partial E(\mathbf{x}_i)}{\partial x_i^{\tilde{l}}} \frac{\partial x_i^{\tilde{l}}}{\partial w_{l,k}} \left\| \frac{\partial \mathbf{x}_i}{\partial t_i} \right\| + E^2(\mathbf{x}_i) \frac{\partial}{\partial w_{l,k}} \left\| \frac{\partial \mathbf{x}_i}{\partial t_i} \right\| \right] \Delta t \\ &= \sum_{i=1}^{M} \left[2E(\mathbf{x}_i) \frac{\partial E(\mathbf{x}_i)}{\partial x_i^l} k(t_i, t_{l,k}) \left\| \frac{\partial \mathbf{x}_i}{\partial t_i} \right\| + E^2(\mathbf{x}_i) \frac{\frac{\partial x_i^l}{\partial t_i} \frac{\partial k(t_i, t_{l,k})}{\partial t_i}}{\left\| \frac{\partial \mathbf{x}_i}{\partial t_i} \right\|} \right] \Delta t\end{aligned} \tag{9.18}$$

9.3.1.2 Loss Function Using a Force Criterion

A second possible loss function, $\mathcal{L}_\mathrm{F}$, can be defined via the forces $\mathbf{F}_\perp$ perpendicular to the curve. The forces $\mathbf{F}_\perp$ are defined as the negative gradients of E, the PES or its surrogate, from which the components along the path are projected out. At a point $\mathbf{x}_i$ on the curve, given the gradient of the PES at that point, $\nabla E|_{\mathbf{x}_i}$, this is

$$\mathbf{F}_\perp = -\left(1 - \mathbf{v} \otimes \mathbf{v}\right) \nabla E|_{\mathbf{x}_i} \tag{9.19}$$

with $\mathbf{v} = \frac{\partial \mathbf{x}_i}{\partial t_i} / |\frac{\partial \mathbf{x}_i}{\partial t_i}|$ being the normalized vector that is tangential to the curve. Minimizing these forces along the curve will drive the curve to the MEP. This gives rise to the following loss function.

$$\begin{aligned}\mathcal{L}_\mathrm{F}^\mathrm{exact} &= \int_\mathcal{C} \left\| \mathbf{F}_\perp(\mathcal{C}(t)) \right\|^2 dL \\ &= \int_{t=0}^{1} \left\| \mathbf{F}_\perp(\mathcal{C}(t)) \right\|^2 \left\| \frac{\partial \mathcal{C}(t)}{\partial t} \right\| dt\end{aligned} \tag{9.20}$$

Using a discretization of $M+2$ points on the path, $\mathbf{x}_i = \mathbf{x}(t_i)$, $i = 0, ..., M+1$, this can be approximated by

$$\mathcal{L}_\mathrm{F} = \sum_{i=0}^{M+1} \left\| \mathbf{F}_\perp(\mathbf{x}_i) \right\|^2 \left\| \frac{\partial \mathbf{x}_i}{\partial t_i} \right\| \Delta t \tag{9.21}$$

where $\mathbf{F}_\perp$, $\frac{\partial \mathbf{x}_i}{\partial t_i}$, and Δt are known. The gradient of the loss functions in curve-parameter space is composed of the following elements.

$$\begin{aligned}\frac{\partial \mathcal{L}_\mathrm{F}}{\partial w_{l,k}} &= \sum_{i=1}^{M}\left[2\sum_{\tilde{l}=1}^{d}\sum_{j=1}^{d} F_\perp^j(\mathbf{x}_i)\frac{\partial}{\partial x_i^{\tilde{l}}}F_\perp^j(\mathbf{x}_i)\frac{\partial x_i^{\tilde{l}}}{\partial w_{l,k}}\left\|\frac{\partial \mathbf{x}_i}{\partial t_i}\right\| + \|\mathbf{F}_\perp(\mathbf{x}_i)\|^2\sum_{\tilde{l}=1}^{d}\frac{\frac{\partial x_i^{\tilde{l}}}{\partial t_i}\frac{\partial^2 x_i^{\tilde{l}}}{\partial w_{l,k}\partial t_i}}{\left\|\frac{\partial \mathbf{x}_i}{\partial t_i}\right\|}\right]\Delta t \\ &= \sum_{i=1}^{M}\left[2\sum_{j=1}^{d} F_\perp^j(\mathbf{x}_i)\frac{\partial}{\partial x_i^{l}}F_\perp^j(\mathbf{x}_i)k(t_i, t_{l,k})\left\|\frac{\partial \mathbf{x}_i}{\partial t_i}\right\| + \|\mathbf{F}_\perp(\mathbf{x}_i)\|^2\frac{\frac{\partial x_i^{l}}{\partial t_i}\frac{\partial k(t_i,t_{l,k})}{\partial t_i}}{\left\|\frac{\partial \mathbf{x}_i}{\partial t_i}\right\|}\right]\Delta t\end{aligned} \tag{9.22}$$

The elements $\frac{\partial}{\partial x_i^l}F_\perp^j(\mathbf{x}_i) = -H_\perp^{lj}(\mathbf{x}_i)$ equal the elements of the negative of the projected Hessian matrix

$$H_\perp(\mathbf{x}_i) = (1 - \mathbf{v}\otimes\mathbf{v})\, H(\mathbf{x}_i)\, (1 - \mathbf{v}\otimes\mathbf{v}) \tag{9.23}$$

with $\mathbf{v}$ being the normalized vector that is tangential to the curve and $H(\mathbf{x}_i)$ being the Hessian matrix of the PES or its surrogate at the point $\mathbf{x}_i$.

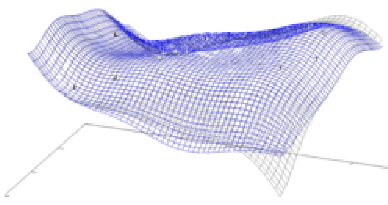
9.3.1.3 Reformulation in the Space of the Control Points

It is more intuitive to formulate the problem so that the control points $\mathbf{x}_n$ with $n = 1, ..., N$ are optimized and not the rather abstract parameters of the regression scheme. The derivatives of the loss functions from above must be formulated with respect to the coordinates of the control points.

$$\frac{\partial \mathcal{L}}{\partial x_m^l} = \sum_{k=1}^{M}\left(K_l^{-1}\right)^{km}\frac{\partial \mathcal{L}}{\partial w_l^k} \tag{9.24}$$

The $\left(K_l^{-1}\right)^{km}$ are the entries of the inverse of K_l, the covariance matrix for the GP used to interpolate the curve in dimension l

$$\sum_{m=1}^{d}(K_l)^{km}\, w_l^m = x_l^k \tag{9.25}$$



for all $k = 1, ..., d$. The inverse of this covariance matrix can be explicitly calculated without being inefficient. This is because the problem is only one-dimensional, does not include many training points, and since the discretization of t is chosen to be uniform $(K_l)^{km} = k(t_k, t_m)$ stays constant as long as the number of training points does not change.

This formulation results in a method with which one can optimize the control points x^l_m along the curve. After the control points are converged and they are equidistant, the curve is interpolated anew with the new control points and equidistant parametrization variables $t_{\tilde{l},k}$. Applying the described method allows to find equally distributed control points for the regression of the curve in Cartesian coordinates that are also equidistant in the parametrization t of the curve. How this optimization is performed in detail is explained in the following.

9.3.2 Optimizing the MEP

To illustrate the details of the path optimization algorithm the explanation is accompanied by a visualization of the optimization in the Müller-Brown PES [56] in figure 9.1, directly taken from Ref. 24. To find the initial guess of the path, it is called *initpath* in the following, three methods are compared.

- Using linear interpolation between the reactant and the product, see figure 9.1a.
- Using the optimized NEB in the image-dependent pair potential (IDPP) [76].
- Using the optimized geodesic [85].

All configurations on the initpath are aligned to the reactant, i.e. rotational and translational changes of the system are eliminated. The presented GPR-based optimizer for MEPs, called GPRMEP in the following, is started from an initial discretization of the initpath. In the presented test cases these are $N = 10$ points, $\mathbf{x}_i$, with $i = 1, ..., 10$. These points are then used to regress a curve using the regression method for curves presented in section 9.1. The user can specify how many control points shall be used to regress the curve. In the presented test cases simply 10 control points are used as well which is also the default value. If one chooses a number that is different from the number of points on the initpath, the curve is first regressed using the discretization of the initpath. Then the desired number of points is selected on the regressed curve so that the points are equidistant.

The optimization is started by calculating the energies and gradients at all of the initial points of the curve. This information is used to build up a GPR-based surrogate for the PES, the GPR-PES. Then the control points of the curve are optimized by minimizing a loss function $\mathcal{L}$ that is yet to be constructed from the two possible loss functions presented in section 9.3.1. These optimizations of the control points on the GPR-PES are performed using the gradient-based L-BFGS optimizer [57]. The L-BFGS optimization is stopped when certain convergence criteria are fulfilled. Let $\mathbf{g}$ be the gradient of the loss function. Further, let $\mathbf{s}$ be the vector suggested by the last L-BFGS step, i.e. the aggregation of the changes of all control points in all dimensions. The standard convergence criteria of DL-FIND are chosen.

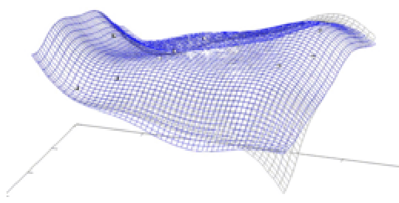
$$\max_i(g_i) < \delta_{\max(g)} := \delta \tag{9.26}$$

$$\frac{\mathbf{g}}{d} < \delta_g := \frac{2}{3}\delta \tag{9.27}$$

$$\max_i(s_i) < \delta_{\max(s)} := 4\,\delta \tag{9.28}$$

$$\frac{\mathbf{s}}{d} < \delta_s := \frac{8}{3}\delta \tag{9.29}$$

The parameter δ is user-defined and g_i and s_i are the components of $\mathbf{g}$ and $\mathbf{s}$. The optimization of the control points is also stopped if their coordinates change too much compared to the previous guess of the curve. Because when the control points deviate strongly from the initial ones, the confidence in the GPR-PES model is low. The coordinates of the endpoints (the first and the last control point) are never changed since they are assumed to be in the optimized minimum of the real PES. After L-BFGS suggested a new step on the GPR-PES the algorithm proceeds as follows.

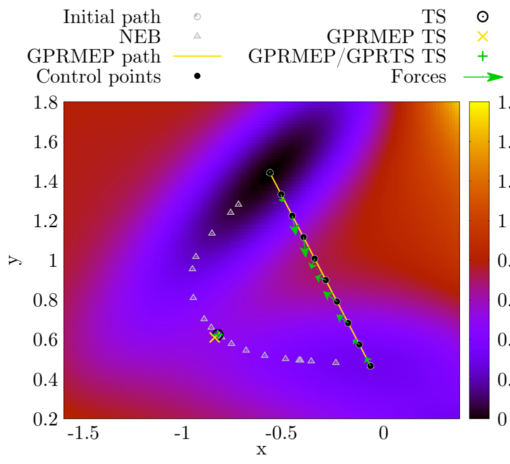


- Project out the components parallel to the path from the step vector resulting from L-BFGS. The same projection is done with the gradient of the loss function to check its convergence. Note that these are only small corrections that yield slightly better results in the end.
- Limit the step size to 2×10^{-2}.
- Move the control points.
- Build the new path with the new control points.
- Avoid small distances: If two points on the path got too close together, all points on the curve are made equidistant in the coordinate space (Cartesian) as described in section 9.2.

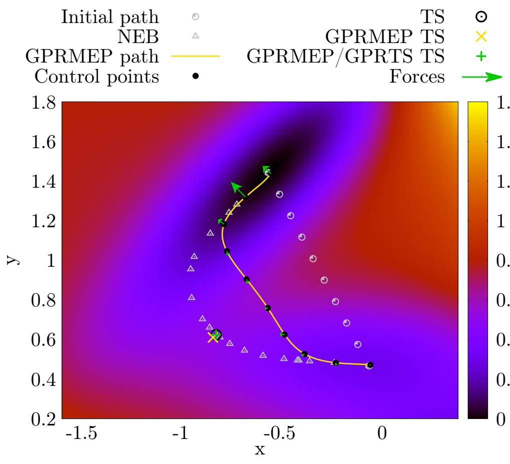
The complete procedure (until the optimization on the GPR-PES is converged) is called a *GPRMEP step.*

The loss function $\mathcal{L}$ drives the L-BFGS optimizations. Its construction is explained in the following. Since the GPR-PES is built solely by energies and gradients it has no meaningful second derivative information in the direction perpendicular to the path in the first GPRMEP step. Only energies and gradients on the initial guess of the path are calculated. Therefore, the energy-based loss function, $\mathcal{L}_{\mathrm{E}}$, see section 9.3.1.1, is used in the first GPRMEP step. This loss function does only require first-order derivatives of the GPR-PES. The forces drawn in figure 9.1a are the ones from the energy-based loss function. In the next optimizations on the GPR-PES one can also make use of the force-based loss function, see section 9.3.1.2, since more gradient information is added to train the GPR-PES and the second-derivative information is expected to improve steadily. Therefore, the force-based loss function, $\mathcal{L}_F$, becomes more useful every step: The loss function $\mathcal{L} \equiv \mathcal{L}_i$ in step $i = 1, 2, ..., 6$ is chosen as a mixture of both loss functions with increasing influence of $\mathcal{L}_F$. From step 7 onwards only $\mathcal{L}_F$ is used.

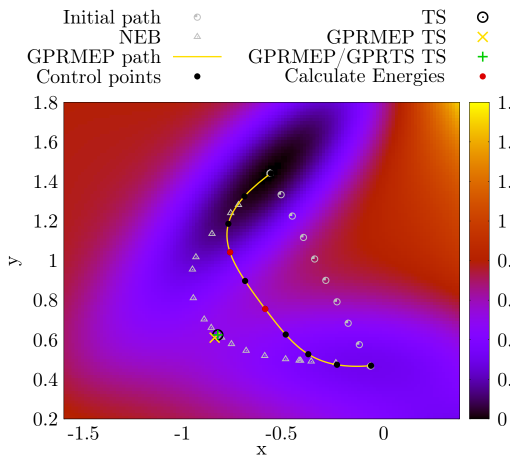
$$\mathcal{L}_i = \begin{cases} \frac{i-1}{5}\mathcal{L}_{\mathrm{F}} + \frac{6-i}{5}\mathcal{L}_{\mathrm{E}} & i = 1, ..., 6 \\ \mathcal{L}_{\mathrm{F}} & i = 7, ... \end{cases} \tag{9.30}$$



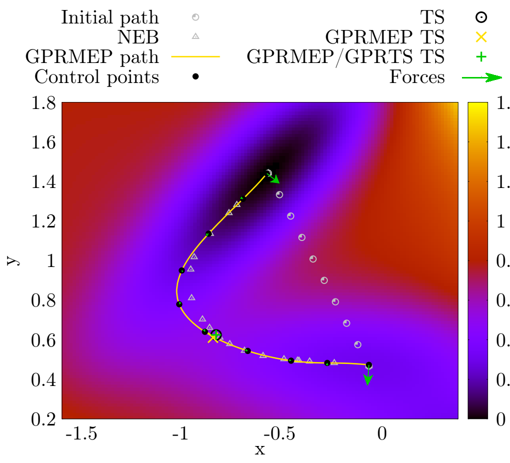
(a) Choose a linear interpolation between the two minima as the initial guess for the path.



(b) The forces on the control points drive the optimization.

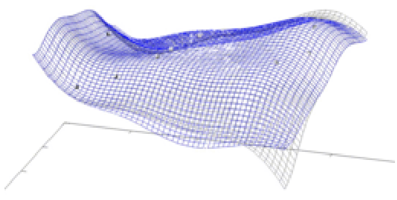


(c) After making the points equidistant, decide where to calculate energies. Include calculated energies and gradients in the GPR-PES.



(d) The path is considered converged when the forces on the path are small enough and all points have a low variance.

Figure 9.1: Visualization of some steps along the optimization in the Müller-Brown potential. Note that the forces depicted in the diagrams are normalized so that the largest force in every picture is always portrayed with an arrow of the same length. As a result, the graphical depiction of the forces cannot be compared between different images.



The advantage of the force-based loss function is that it does not drive the control points along the curve in the direction of the minima. Using only the energy-based loss function results in an agglomeration of control points in the minima. This can potentially lead to coiling of the curve and strong movement in that area. For the Müller-Brown PES the forces in step 5 can be seen in figure 9.1b.

When one GPRMEP step is completed, i.e. the optimization on the GPR-PES is converged, the points on the curve are made equidistant in real coordinate space (Cartesians) as described in section 9.2. Such a redistribution process of the control points only results in small changes of the curve since it is performed after every GPRMEP step. But it can be seen in the Müller-Brown example when comparing figure 9.1b to figure 9.1c.

To determine at which points one should calculate the next energies/gradients to improve the GPR-PES one makes use of the statistical properties of GPR. The variance of the GPR-PES is calculated (parallelized with OpenMP) at the control points of the curve. Only the local maxima of these calculated variances are considered. In this case a local maximum simply means that the control points to either side have a lower variance. If the calculated variance of such a local maximum is above a certain threshold (10^{-11}), the coordinates of this maximum are added to a list that is called `ToCalcList`. In figure 9.1c the points that are added to the `ToCalcList` are marked in red. The `ToCalcList` is sorted according to descending variance. Now one successively calculates energies and gradients on the real PES at the points on the `ToCalcList` starting from the point with highest variance. Every time when energy/gradient were calculated at the point of highest variance it is checked whether all other points in the `ToCalcList` still have a variance above the threshold (10^{-11}). If not, they are eliminated from the list. As soon as the energy/gradient for every point on the `ToCalcList` are calculated another GPRMEP step is started from the last guessed curve.

The variance criterion avoids the possibility that two training points are very close to each other. Having two very close training points can lead to numerical problems for the covariance matrix, i.e. it might not be clearly positive-definite.

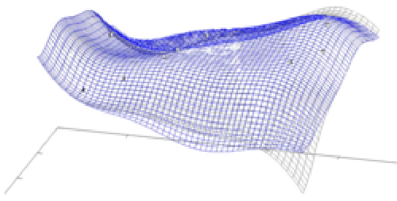
But the variance criterion also gives a good estimate of how confident one can be that the converged MEP on the GPR-PES is also a valid MEP on the real PES. Therefore, two convergence criteria are used which both have to be fulfilled.

- The curve did not change much with a GPRMEP step. This is determined by choosing 60 equidistant (in the curve parametrization t) points on the old and on the new curve. The mean distance between the respective points as well as the maximum distance between them must both be smaller than 5×10^{-3}.
- The `ToCalcList` is empty.

The resulting curve (the converged MEP) is only a good approximation of the path, see figure 9.1d, and does not necessarily go through the exact transition state. However, the energetically highest point on this path is a good guess for the TS that can be optimized afterwards. This is usually true for NEB as well. One can start the GPRTS algorithm, presented in the previous chapter, from that guess for the TS. Fortunately, one can reuse the GPR-PES that was created in the GPRMEP procedures described above in the GPRTS algorithm. This will speed up its convergence significantly and GPRTS will only need few additional energy evaluations to get a properly converged TS.

9.4 Applications/Benchmarks

As a benchmark for GPRMEP the same 27 test cases as for GPRTS in the previous chapter are chosen. The minima that were used to benchmark the GPRTS/IDPP and GPRTS/geodesic method in the previous chapter are the same that are used to start GPRMEP in this chapter. The minima and TSs for all test systems are shown in figure 9.2 and figure 9.3.



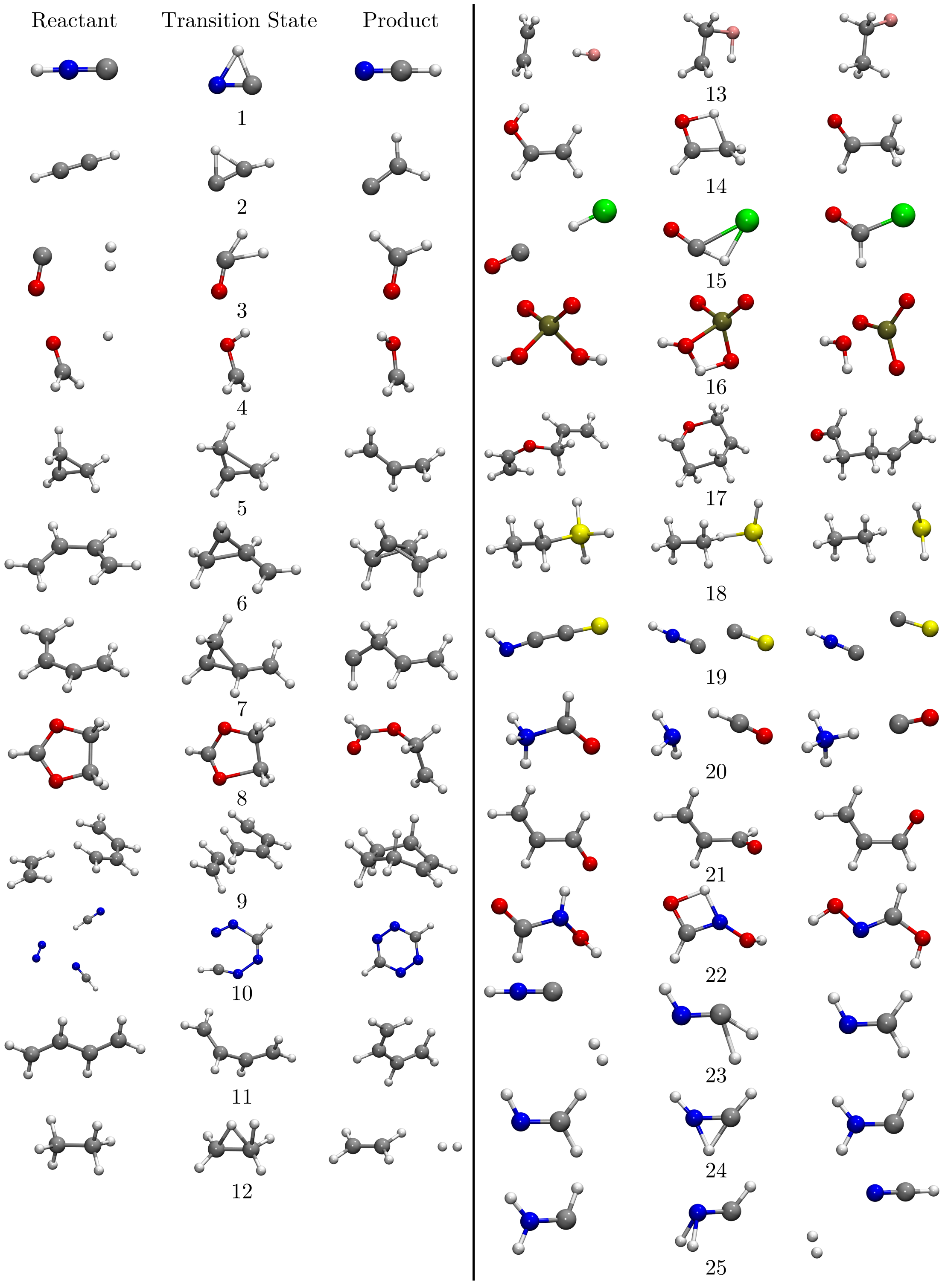


Figure 9.2: Reactants, TSs and products of the reactions implied by the Baker test set. The MEP searches are started from the reactant and product structures only. The TSs are the reference structures for the benchmark that were optimized beforehand.

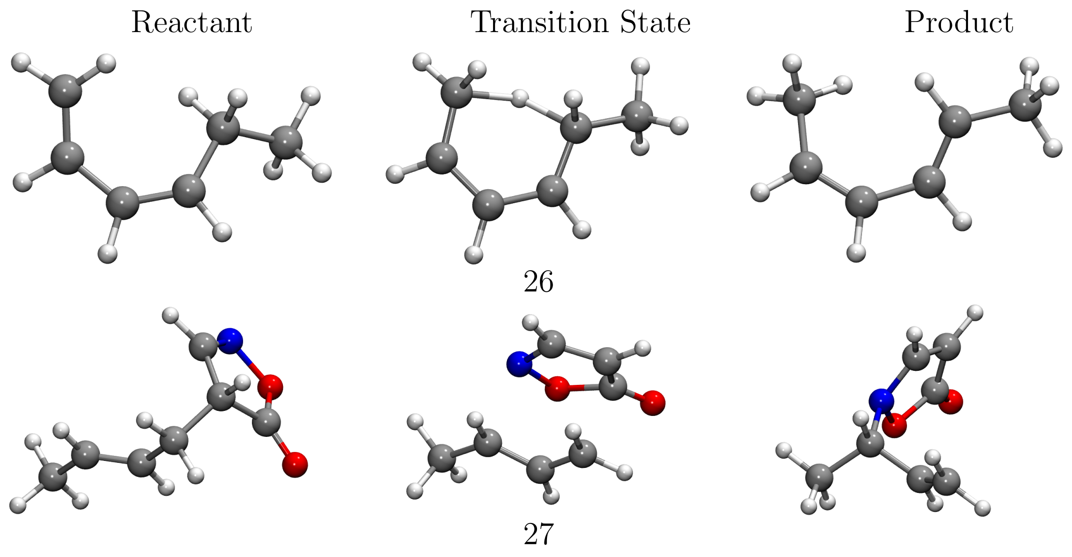


Figure 9.3: Reactants, TSs and products of the reactions of system 26 [53], the [1,5] H shift of 1,3(Z)-hexadiene to 2(E),4(Z)-hexadiene and System 27 [68], an asymmetric allylation of a simple isoxazolinone.

All optimizations are carried out in Cartesian coordinates. The results of these optimizations are shown in table 9.1. Results are marked with *false* if the found TS does not correspond to the reference TS, *nc* if the algorithm did not converge after 1000 energy evaluations, and *err* if an error of the AM1 convergence occured. Note that *false* optimizations might also be occurring because of flaws in the electronic structure method. They are not necessarily a consequence of a bad optimizer.

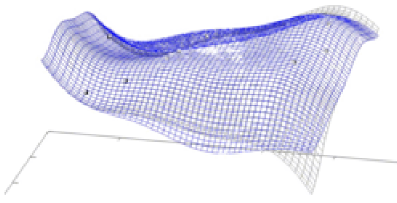
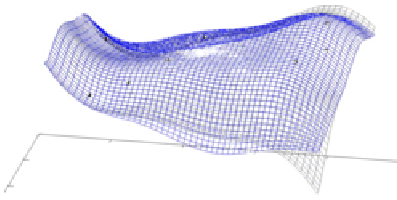


Table 9.1: The number of energy evaluations that the different optimizers need until convergence.

		Linear interpolation			IDPP			Geodesic		
d	System ID	NEB	GPRMEP	+GPRTS	NEB	GPRMEP	+GPRTS	NEB	GPRMEP	+GPRTS
48	26	179	10	57	158	40	87	124	43	77
57	27	303	16	198	306	56	269	123	59	119
48	9	271	35	69	*err*	36	85	*err*	31	72
42	17	*nc*	410	465	*false*	68	161	*false*	56	115
33	18	90	25	44	114	26	53	*false*	23	54
30	6	247	65	103	241	40	88	285	38	92
30	7	*nc*	45	75	194	45	87	145	39	68
30	8	*err*	76	104	*err*	*false*	*false*	*false*	*false*	*false*
30	11	94	40	64	46	23	53	25	22	46
24	5	148	30	48	*err*	23	43	207	25	44
24	10	265	54	77	213	52	86	*false*	51	87
24	12	*err*	77	155	*err*	*nc*	*nc*	*err*	74	168
24	13	130	46	70	169	64	102	*nc*	76	107
24	21	*err*	29	*err*	59	33	138	77	33	156
21	14	139	36	53	138	37	57	101	34	53
21	16	88	42	52	105	60	94	715	34	53
21	20	157	41	59	163	36	66	151	41	80
21	22	121	33	51	126	32	55	117	34	55
15	4	*false*	34	53	*false*	33	70	*false*	*false*	*false*
15	19	81	20	30	95	22	40	95	20	37
15	23	154	33	45	*false*	39	87	*err*	48	119
15	24	83	22	28	114	23	36	91	24	37
15	25	196	36	50	174	38	69	*err*	69	105
12	2	145	36	41	130	33	50	318	29	45
12	3	89	33	41	241	51	96	1911	26	48
12	15	132	30	44	124	33	58	289	31	54
9	1	168	50	55	168	29	43	257	64	84

Especially since there are no RMSD values presented in the table, some problems should be addressed for runs in which the RMSD values of the obtained TSs to the reference structure are not small.

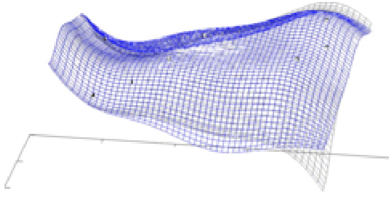
- System 4
  - Starting from either of the three initial paths NEB yields a larger distance of the H atom in the OH group and the angles are not correct. The eigenvalue of the Hessian corresponding to the transition mode is almost zero or even positive.
  - Starting from the linear interpolation and the IDPP approximation GPRMEP + GPRTS finds a TS with a slightly larger distance of the H atom in the OH group. The negative eigenvalue of the Hessian is larger than in the reference structure but the structure looks reasonable.
  - Starting from the geodesic solution GPRMEP + GPRTS finds an unrealistically large distance of the H atom as in the NEB calculations.
  - This system has a flat PES near the TS which leads to problems with both optimizers. Setting tighter convergence criteria can overcome this problem.
- System 8
  - GPRMEP + GPRTS starting from the IDPP solution and the geodesic solution yield a TS in which the ring is opened further.
  - NEB starting from the geodesic solution shows the same problem.
- System 10
  - NEB starting from linear interpolation yields a structure that looks very similar to the reference structure but has two negative eigenvalues. The reason are small angular deviations in the ring structure.
  - NEB starting from the geodesic solution yields a closed, deformed ring structure.



- System 17
  - NEB starting from the IDPP solution yields an opened ring structure that corresponds to the rotation of the aldehyde group.
  - NEB starting from the geodesic solution shows a separation of the molecule in several smaller structures and has no similarity to the desired TS anymore.
- System 18
  - NEB starting from the geodesic solution yields a high distance between SiH_2 and the rest of the molecule.
- System 21
  - The GPRMEP + GPRTS results starting from the geodesic solution yield a TS with a slight rotation of the carboxyl group. The angular deviation is approximately 10°. The Hessian's eigenvalues are similar to the reference structure.
- System 23
  - The NEB run starting from the IDPP solution yields a large distance of the H_2 molecule from the rest of the system. There is no negative eigenvalue.
- System 25
  - All converging optimization runs yield the same TS that has a different angle to the abstracted H_2 molecule than in the reference structure. The H_2 molecule is rotated by approximately 90°.

GPRMEP yields plausibly looking MEPs but the obtained TSs from GPRMEP alone are not fully converged, much like it is often the case in NEB. Usually the point of maximum energy on the MEP obtained by GPRMEP is a worse estimate of the TS than the one obtained by NEB. This is to be expected since no TS optimization in the sense of the climbing image procedure is performed. But the estimate is intended to work as a starting guess for a GPRTS optimization. Using the already built GPR-PES it is very advantageous to combine GPRMEP with a succeeding GPRTS run to optimize the TS. GPRMEP combined with GPRTS needs much fewer energy evaluations than NEB and is clearly more stable in the presented test systems. Furthermore, in contrast to NEB it already yields fully converged TSs. The TSs from NEB often must be refined with traditional TS optimizers. The traditional optimizers are expected to perform worse than GPRTS. Not only because of the general performance improvement, see section 8, but because GPRTS can use the GPR-PES that was built during the GPRMEP run.

Comparing the different possibilities for a starting guess the results indicate no clear advantage of either method. Visually only the paths from the IDPP method show some unrealistic wiggling of the abstracted H_2 molecule in system 12 and 23. All other paths look acceptable when inspecting them visually.



9.5 Discussion

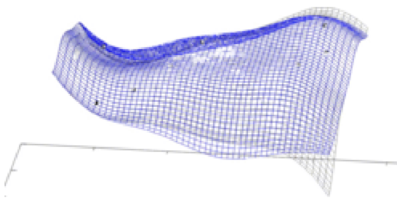
The GPRMEP optimizations have a computational overhead. This is mainly because of the MEP optimization on the GPR-PES and the construction of the GPR-PES. In the largest test systems 26/27 the average time consumption of GPRMEP + GPRTS is 21%/35%. The remaining time is used for the DFT calculations. For system 26 the overall time consumption of the optimizations is 44% lower, for system 27 it is 17% higher. Note that most of this time is spent in the GPRTS optimizer, i.e. the refinement of the TS. The TSs obtained by NEB must be refined with a further optimization which is not necessary for the GPRMEP + GPRTS combination. The timings were performed on two 8-core Intel Xeon E5-2670 with parallelized DFT calculations.
The scaling of the overall algorithm is dominated by the scaling of the evaluation of the Hessians on the GPR-PES: It scales linearly in the number of training points but cubically with the number of dimensions in the system. The multi-level approach that must be employed for systems with hundreds of atoms to avoid excessive memory usage will most likely limit the performance of the optimizer. The GPRMEP optimizer is not recommended for systems that are much larger than 500 degrees of freedom, although systems up to 1000 degrees of freedom might be possible to handle. For larger systems one must freeze atoms that are not relevant to the reaction.
The optimizer is sensitive to numerical errors. Different floating point models for the compiler lead to different but not consistently worse or better results. The reason lies mainly in the GPRTS method. Also the creation of the GPR-PES itself is dependent on the floating point model, compare section 8.3.

10 Updating Hessians

In this rather short chapter it is shown that it is possible to update Hessians in a very reliable way using GPR. *Updating a Hessian* in this case means that a Hessian that was determined by electronic structure theory at a point $\mathbf{x}_0$ is changed by using gradient information at another point $\mathbf{x}_1$ so that it gives a reasonable approximation of the Hessian at the point $\mathbf{x}_1$. Commonly used methods for that are the update mechanisms by Powell [64] and Bofill [17] when performing a TS search. Also BFGS, explained in section 2.1, is in principle a similar update mechanism, but it is only usable for minimizations.

10.1 The Update Mechanism

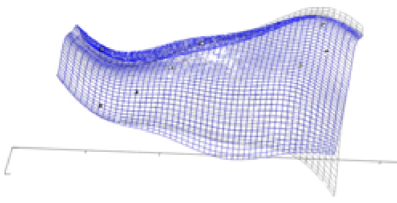
It is quite straightforward to implement an update mechanism for Hessians using GPR. One simply evaluates the Hessian of the PES surrogate, called GPR-PES, at the point $\mathbf{x}_1$ while the GPR-PES was constructed using the original Hessian information at point $\mathbf{x}_0$ and the gradient information at $\mathbf{x}_1$. However, the algorithm must be able to combine training data of different types, i.e. including only gradients or including gradients and Hessians. This is possible with the GPR software that was implemented during the work for this thesis. Furthermore, the prior of the energy in equation (5.34) is chosen to be a Taylor expansion of order 2 at the last point at which the Hessian was calculated. This significantly improves the quality of the update mechanism.
To demonstrate the potential benefits this update mechanism was implemented in DL-FIND so that it can be used for various optimizations. A short benchmark is presented, on the Baker test set and two larger DFT-based test systems, the same systems used in section 8 and section 9. See these sections for further information on the test systems. In this benchmark P-RFO optimizations are per-



formed with different Hessian update mechanisms. New Hessians are calculated after 50 updates in these optimization runs. Note that analytic Hessians are not available for the AM1 method used for systems 1-25. The gradient evaluations needed for the central difference scheme to obtain numerical Hessians are counted as energy evaluations in the table. Analytic Hessians in systems 26 and 27 are not counted as energy evaluations. The tolerance parameter is set to $\delta = 3 \times 10^{-4}$, the maximum step size to $s_{\max} = 0.3$. The results are presented in table 10.1. The number of required energy calculations until convergence is shown, as well as the RMSD values (Å) of the resulting structures compared to the carefully optimized reference structures that were also used for the GPRTS benchmarks. Convergence problems of the electronic structure method are marked with *err*, non-convergence after 1000 energy evaluations is marked with *nc* for *not converged*.

	Energy evaluations			RMSD values			
d	GPR	Bofill	Powell	GPR	Bofill	Powell	ID
123	32	419	120	2.4×10^{-2}	5.9×10^{-1}	1.1×10^{-2}	26
123	18	731	386	6.3×10^{-2}	1.2	2.1	27
48	*nc*	*nc*	*nc*				9
42	101	130	*nc*	5.0×10^{-3}	9.1×10^{-4}		17
33	80	421	*nc*	6.2×10^{-1}	7.9×10^{-3}		18
30	79	175	174	1.2×10^{-1}	1.2×10^{-1}	1.2×10^{-1}	6
30	174	433	214	1.2×10^{-4}	5.8×10^{-1}	1.7×10^{-4}	7
30	67	83	112	1.9×10^{-4}	1.8×10^{-4}	7.6×10^{-4}	8
30	185	71	71	1.0×10^{-3}	7.4×10^{-1}	7.4×10^{-1}	11
24	67	85	68	2.7×10^{-4}	2.8×10^{-4}	2.8×10^{-4}	5
24	89	151	259	2.6×10^{-1}	2.6×10^{-1}	2.6×10^{-1}	10
24	86	68	80	1.1×10^{-4}	1.3×10^{-4}	1.8×10^{-4}	12
24	71	73	80	1.1×10^{-4}	6.5×10^{-5}	2.1×10^{-4}	13
24	65	55	55	1.0×10^{-4}	4.5×10^{-3}	4.4×10^{-3}	21
21	68	80	73	4.2×10^{-1}	2.3×10^{-4}	4.2×10^{-1}	14
21	59	79	82	7.4×10^{-5}	4.4×10^{-5}	4.6×10^{-4}	16
21	79	142	912	5.7×10^{-2}	3.0×10^{-1}	1.3	20
21	146	174	233	2.7×10^{-1}	2.7×10^{-1}	2.7×10^{-1}	22
15	118	66	115	4.9×10^{-1}	2.3×10^{-1}	4.9×10^{-1}	4
15	52	44	60	1.1×10^{-4}	7.7×10^{-3}	3.6×10^{-4}	19
15	47	56	69	7.8×10^{-5}	2.2×10^{-4}	9.0×10^{-5}	23
15	49	114	58	4.7×10^{-5}	5.6×10^{-5}	2.9×10^{-4}	24
15	47	59	67	5.3×10^{-4}	1.1×10^{-3}	3.3×10^{-4}	25
12	34	35	41	1.0×10^{-4}	7.4×10^{-5}	4.9×10^{-5}	2
12	36	190	101	5.6×10^{-6}	1.2	3.3×10^{-5}	3
12	*err*	*err*	*err*				15
9	30	30	31	1.3×10^{-5}	4.6×10^{-5}	8.3×10^{-6}	1

Table 10.1: A comparison of P-RFO optimizations using different update mechanisms for the Hessian.



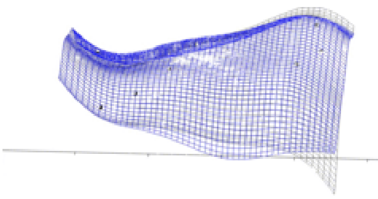
The benchmark suggests that the GPR-based updates are more reliable and allow faster P-RFO convergence than Bofill- and Powell-based updates. Some comments on the higher RMSD values in the chart are following:

- System 3: The Bofill-based run finds a different structure with a larger H_2 distance in which the Hessian has two negative eigenvalues.
- System 4: Powell- and GPR-based runs show structures that have no negative eigenvalue in their Hessian and are planar. In the Bofill-based run a structure with a different distance of the H atom is obtained.
- System 6: The angles are only slightly different from the reference. If the optimizations are done with a tighter convergence criterion, they converge to the correct TS.
- System 7: The Bofill-based run finds a different structure in which the ring structure opens. The negative eigenvalue of its Hessian has a very small absolute value.
- System 10: All runs yield a closed ring structure with no negative eigenvalue.
- System 11: The Bofill- and Powell-based runs yield an almost plain structure with no negative eigenvalue.
- System 14: GPR- and Powell-based runs yield the mirror image of the reference structure.
- System 18: The GPR-based run shows a twist in the molecule.
- System 20: In the GPR- and Bofill-based runs the distance of the two molecules in the structure vary slightly. In the Powell-based run all H atoms stay at the N and the structure's Hessian has no negative eigenvalue.
- System 22: All obtained structures show the same TSs in which the OH group is attached with a different angle, not in a planar arrangement as in the reference structure.

- System 26: The H atom is not transferred but the angles in the molecule are changed. This is the same phenomenon observed in the P-RFO optimization in section 8.2.1.
- System 27: Bofill- and Powell-based runs show structures that are close to either of the two minima. The Powell-based run does not have a negative eigenvalue. The GPR structure looks close to the reference.

10.2 Discussion

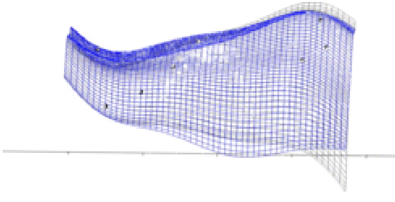
The presented benchmark gives only an indication towards the possibilities of GPR-based Hessian updates. But it suggests that GPR-based updating can outperform other, widely used update schemes for P-RFO optimization. As towards the limitations of this update scheme one must clearly state that the memory requirements might become quite large for high-dimensional systems if many Hessians are included. Also the computational scaling is worse when explicitly using Hessians. But on the other hand one can possibly drop later training data much like other update-schemes do as well, i.e. only one single Hessian is used which is updated with following gradients. Then the scaling of the method becomes manageable. The dimension of the linear system in GPR is then limited to $d \times (d+1+m)+m+1$ if one performs updates after m steps and d is the number of dimensions in the system.



11 Final Discussion/Conclusion

In the present study three different optimizers and a method to update Hessians were presented, all based on a newly developed GPR implementation, see section 6. The results of the benchmarks indicate that the optimizers can outperform classical optimizers drastically in terms of number of required electronic structure calculations. It was also shown that the optimizers can be expected to be more stable in many cases and that they can be a reasonable choice when other optimizers fail to converge.

The largest problem for GPR-based algorithms is the high scaling of classical GPR in terms of computational and memory demand. The computational demand is significantly reduced (from cubic scaling to quadratic scaling) by the iterative implementation of the Cholesky decomposition, see section 6.1. Only the TS optimizer and the MEP optimizer scale cubically in the number of dimensions in the system. This is because they require Hessians on the GPR-PES that is constructed using gradients. The memory demand is reduced by employing a multi-level scheme, see section 6.2. It can make the algorithm's memory demand independent of the length of the optimization history, i.e. only the scaling with the system size can be a problem. However, for very large test systems with many hundreds of atoms the presented optimizers might not be applicable in a meaningful way because of the memory restrictions. From theoretical considerations and some practical experience it is recommended to apply the optimizers only for a maximum of 500 atoms. In larger systems one might consider freezing atoms that do not participate in the respective reaction. This would allow the algorithms to be employed on larger systems as well. The higher the level of theory used for the electronic structure calculations the more beneficial the use of the presented optimizers can be. The reduced number of energy evaluations often outweighs the

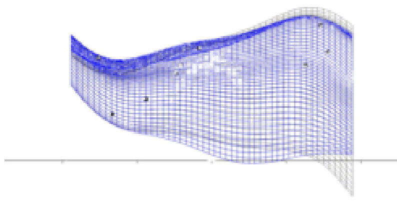


computational overhead of the GPR-based optimizers.

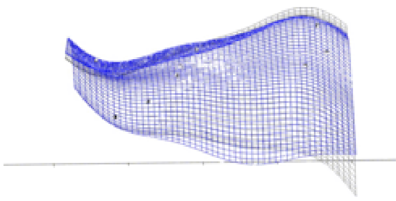
The presented GPR implementation and optimizers are only capable of handling Cartesian coordinates. Other coordinate systems are able to intrinsically incorporate translational and rotational invariance. Many coordinate systems can also incorporate invariance to permutations of identical atoms. These can in many cases improve the performance of machine learning methods [11, 13, 36, 66].

However, some of these more modern descriptors like the Coulomb matrix representation of the Bag of Bonds approach increase the number of dimensions in the system. That would lead to a larger overhead of the optimizers. In other works it was shown that the Z-Matrix representation can be used for geometry optimization with GPR [70] in which new hyperparameters are introduced for the length scales. The algorithms implemented in this thesis are not capable of handling those. Cartesian coordinates are preferred because of their simplicity. Furthermore, it is not clear that the benefits for geometry optimization are large when using other coordinate systems. The reason is that geometry optimization is a comparably local task which might make the said invariances irrelevant. It was shown that for traditional TS optimizers the usage of internal coordinate systems like the Z-matrix is not generally advisable [8, 9].
The hyperparameters that were chosen in the presented optimizers are mainly determined heuristically: The interpolation quality on several test systems was measured using different hyperparameters to find the best performing ones for a large amount of systems. In principle it is possible to optimize these hyperparameters via the *maximum likelihood* method. The underlying reasons are manifold and summarized in section 5.6.
Statistical training methods like GPR usually perform better in interpolation when smaller quantities are interpolated [65]. This inspires the idea of *delta* machine learning in which one uses low-level approximations to the PES and only interpolates the error to a higher-level approximation. This approach has the potential to further increase the performance of the optimizers.

The developed black box optimizers can be expected, especially in smaller systems, to outperform classical optimizers significantly in terms of performance and stability. They find minimum structures, transition states, and minimum energy paths in a reliable and fast way.



Appendices



A.1 Mathematical Appendix

Lemma A.1.1 (Schur Complement)**.** *The inverse of a matrix of the form*

$$\begin{bmatrix} K & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \tag{A.1}$$

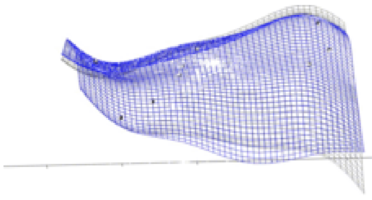
can be written as

$$\begin{bmatrix} K & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix}^{-1} = \begin{bmatrix} \mathbb{1} & -K^{-1}\mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} K^{-1} & \mathbf{0} \\ \mathbf{0}^T & \left(b - \mathbf{a}^T K^{-1}\mathbf{a}\right)^{-1} \end{bmatrix} \begin{bmatrix} \mathbb{1} & \mathbf{0} \\ -\mathbf{a}^T K^{-1} & 1 \end{bmatrix} \tag{A.2}$$

Proof.

$$\begin{aligned} & \begin{bmatrix} \mathbb{1} & -K^{-1}\mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} K^{-1} & \mathbf{0} \\ \mathbf{0}^T & \left(b - \mathbf{a}^T K^{-1}\mathbf{a}\right)^{-1} \end{bmatrix} \begin{bmatrix} \mathbb{1} & \mathbf{0} \\ -\mathbf{a}^T K^{-1} & 1 \end{bmatrix} \begin{bmatrix} K & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \\ = & \begin{bmatrix} \mathbb{1} & -K^{-1}\mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} K^{-1} & \mathbf{0} \\ \mathbf{0}^T & \left(b - \mathbf{a}^T K^{-1}\mathbf{a}\right)^{-1} \end{bmatrix} \begin{bmatrix} K & \mathbf{a} \\ \mathbf{0}^T & -\mathbf{a}^T K^{-1}\mathbf{a} + b \end{bmatrix} \\ = & \begin{bmatrix} \mathbb{1} & -K^{-1}\mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbb{1} & K^{-1}\mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix} \\ = & \begin{bmatrix} \mathbb{1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \end{aligned} \tag{A.3}$$

□



Lemma A.1.2 (Woodbury Matrix Identity)**.** *Let the necessary (implicitly stated) invertibility requirements in the following formula hold for given Matrices A, U, C, V. The dimension of these matrices shall be such that usual matrix multiplication in the equation shall be defined properly.*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{A.4}$$

Proof.

$$\begin{aligned}
&[A + UCV]\left[A^{-1} - A^{-1}U\left[C^{-1} + VA^{-1}U\right]^{-1}VA^{-1}\right] \\
&= \mathbb{1} - U\left[C^{-1} + VA^{-1}U\right]^{-1}VA^{-1} \\
&\quad + UCVA^{-1} - UCVA^{-1}U\left[C^{-1} + VA^{-1}U\right]^{-1}VA^{-1} \\
&= \mathbb{1} + UCVA^{-1} - \left(U + UCVA^{-1}U\right)\left[C^{-1} + VA^{-1}U\right]^{-1}VA^{-1} \\
&= \mathbb{1} + UCVA^{-1} - UC\underbrace{\left[C^{-1} + VA^{-1}U\right]\left[C^{-1} + VA^{-1}U\right]^{-1}}_{=\mathbb{1}}VA^{-1} \\
&= \mathbb{1}
\end{aligned} \tag{A.5}$$

□

A.2 Including Second-Order Derivatives

This section was borrowed from the appendix of the author's work on the GPR-based minimizer [25]. If one wants to use derivative information up to second order, the energy is given by the following equation.

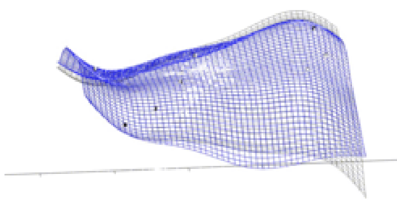
$$\begin{aligned} E(\mathbf{x}) = \sum_{n=1}^{N} w_n k(\mathbf{x}, \mathbf{x}_n) + \sum_{n=1}^{N}\sum_{i=1}^{d} v_n^i \frac{dk(\mathbf{x}, \mathbf{x}_n)}{dx_n^i} \\ + \sum_{n=1}^{N}\sum_{i=1}^{d}\sum_{j=1}^{d} u_n^{ij} \frac{d^2k(\mathbf{x}, \mathbf{x}_n)}{dx_n^i dx_n^j} + E_{\text{prior}}(\mathbf{x}) \end{aligned} \tag{A.6}$$

The number of dimensions in the system is given by d and the number of training points is given by N. The weights w_n, v_n^i, u_n^{ij} will be obtained through a larger covariance matrix. Introducing σ_{h}, the assumed noise on the Hessian entries, the covariance matrix has the following form.

$$K = \begin{bmatrix} k(\mathbf{x}_m, \mathbf{x}_n) + \sigma_{\text{e}}^2\delta_{mn} & \frac{dk(\mathbf{x}_m,\mathbf{x}_n)}{dx_n^i} & \frac{d^2k(\mathbf{x}_m,\mathbf{x}_n)}{dx_n^i dx_n^j} \\ \frac{dk(\mathbf{x}_m,\mathbf{x}_n)}{dx_m^i} & \frac{d^2k(\mathbf{x}_m,\mathbf{x}_n)}{dx_m^i dx_n^j} + \sigma_{\text{g}}^2\delta_{mn}\delta_{ij} & \frac{d^3k(\mathbf{x}_m,\mathbf{x}_n)}{dx_n^i dx_n^j dx_m^k} \\ \frac{d^2k(\mathbf{x}_m,\mathbf{x}_n)}{dx_m^i dx_m^j} & \frac{d^3k(\mathbf{x}_m,\mathbf{x}_n)}{dx_m^i dx_m^j dx_n^k} & \frac{d^4k(\mathbf{x}_m,\mathbf{x}_n)}{dx_n^i dx_n^j dx_m^k dx_m^l} + \sigma_{\text{h}}^2\delta_{mn}\delta_{ik}\delta_{jl} \end{bmatrix} \tag{A.7}$$

The linear system will become

$$K \begin{bmatrix} w_1 \\ \vdots \\ w_N \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} E_1 \\ \vdots \\ E_N \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_N \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_N \end{bmatrix} - \begin{bmatrix} E_{\text{prior}}(\mathbf{x}_1) \\ \vdots \\ E_{\text{prior}}(\mathbf{x}_N) \\ \nabla E_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} \\ \vdots \\ \nabla E_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_N} \\ \mathbf{h}_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} \\ \vdots \\ \mathbf{h}_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_N} \end{bmatrix} \tag{A.8}$$



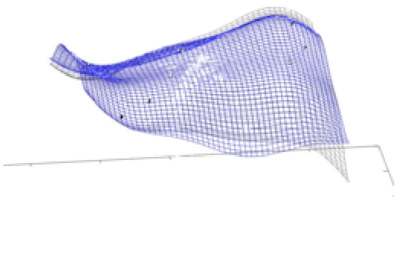
in which the entry $(i \cdot d + j)$ of vector $\mathbf{h}_m$ is $h_m^{i \cdot d+j} = H_m^{ij}$, the entry of the Hessian matrix corresponding to the coordinates i and j at the training point m and the entry $(i \cdot d + j)$ of $\mathbf{h}_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_m}$ the second derivative of the prior mean function $\frac{d^2}{dx^i dx^j} E_{\text{prior}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_m}$, respectively. The vectors $\mathbf{u}_m$ are of the same shape. In the GPR implementation of this thesis only half of the Hessian entries are used by exploiting the Hessian's symmetry. This will limit the number of entries in the vectors $\mathbf{h}_m$ and $\mathbf{u}_m$ to $(d[d+1])/2$ which increases the efficiency quite a bit. Calculations using second order derivative information scale with $\mathcal{O}([Nd[d+1]^2/2 + Nd + N]^3)$. Therefore, they should only be used for regression tasks with only a view data points for which the Hessian information is available.

A.3 Derivatives of Covariance Functions

In the following the derivatives necessary for implementing GPR up to second for two different kernels are presented. The derivatives of the squared exponential and the Matérn covariance function with $\nu = 5/2$ are shown. For the optimization of the likelihood additional derivatives with respect to the parameters are needed. They are not shown here but can be extracted from the DL-FIND code.

A.3.0.1 Squared Exponential Kernel

$$
\begin{aligned}
k(x_m, x_n) &= \sigma_f^2 \exp\left(-\frac{\gamma}{2}|\mathbf{x}_m - \mathbf{x}_n|^2\right) \\
\frac{dk(x_m, x_n)}{dx_n^i} &= \sigma_f^2 \gamma (x_m^i - x_n^i) \exp\left(-\frac{\gamma}{2}|\mathbf{x}_m - \mathbf{x}_n|^2\right) \\
\frac{dk(x_m, x_n)}{dx_m^i} &= -\frac{dk(x_m, x_n)}{dx_n^i} \\
\frac{d^2k(x_m, x_n)}{dx_m^i dx_n^j} &= \sigma_f^2 \gamma \left[\delta_{ij} + \gamma(x_n^i - x_m^i)(x_m^j - x_n^j)\right] \exp(...) \\
\frac{d^2k(x_m, x_n)}{dx_n^i dx_n^j} &= -\frac{d^2k(x_m, x_n)}{dx_m^i dx_n^j} \\
\frac{d^2k(x_m, x_n)}{dx_m^i dx_m^j} &= \frac{d^2k(x_m, x_n)}{dx_n^i dx_n^j} \\
\frac{d^3k(x_m, x_n)}{dx_n^i dx_n^j dx_m^k} &= \sigma_f^2 \gamma^2 \Big[-\delta_{ij}(x_n^k - x_m^k) + \delta_{jk}(x_m^i - x_n^i) + \delta_{ik}(x_m^j - x_n^j) \\
&\qquad + \gamma(x_m^j - x_n^j)(x_m^i - x_n^i)(x_n^k - x_m^k)\Big] \exp(...) \\
\frac{d^3k(x_m, x_n)}{dx_m^i dx_m^j dx_n^k} &= -\frac{d^3k(x_m, x_n)}{dx_n^i dx_n^j dx_m^k} \\
\frac{d^4k(x_m, x_n)}{dx_n^i dx_n^j dx_m^k dx_m^l} &= \sigma_f^2 \gamma^2 \Big[\delta_{ij}\left[\delta_{kl} - \gamma(x_n^k - x_m^k)(x_n^l - x_m^l)\right] + \\
&\qquad \delta_{jk}\left[\delta_{il} - \gamma(x_n^i - x_m^i)(x_n^l - x_m^l)\right] + \\
&\qquad \delta_{ik}\left[\delta_{jl} - \gamma(x_n^j - x_m^j)(x_n^l - x_m^l)\right] + \\
&\qquad \gamma\big[\delta_{jl}(x_m^i - x_n^i)(x_n^k - x_m^k) + \\
&\qquad\quad \delta_{il}(x_m^j - x_n^j)(x_n^k - x_m^k) + \\
&\qquad\quad \delta_{kl}(x_m^j - x_n^j)(x_n^i - x_m^i) + \\
&\qquad\quad \gamma(x_m^j - x_n^j)(x_m^i - x_n^i)(x_n^k - x_m^k)(x_n^l - x_m^l)\big]\Big] \\
&\quad \exp(...)
\end{aligned}
\tag{A.9}
$$



A.3.0.2 Matérn Kernel for $\nu = 5/2$

Let $r := |\mathbf{x}_m - \mathbf{x}_n|$.

$$\begin{aligned}
k(x_m, x_n) &= \sigma_f^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left[-\frac{\sqrt{5}r}{l}\right] \\
\frac{dk(x_m, x_n)}{dx_n^i} &= \sigma_f^2 \, \frac{5}{3l^3} \left[(x_m^i - x_n^i)\left(l + \sqrt{5}r\right)\right] \exp\left[-\frac{\sqrt{5}r}{l}\right] \\
\frac{dk(x_m, x_n)}{dx_m^i} &= -\frac{dk(x_m, x_n)}{dx_n^i} \\
\frac{d^2k(x_m, x_n)}{dx_m^i dx_n^j} &= \sigma_f^2 \, \frac{5}{3l^4} \left[(\delta_{ij} l(l + \sqrt{5}r) - 5(x_m^i - x_n^i)(x_m^j - x_n^j))\right] \exp\left[-\frac{\sqrt{5}r}{l}\right] \\
\frac{d^2k(x_m, x_n)}{dx_n^i dx_n^j} &= -\frac{d^2k(x_m, x_n)}{dx_m^i dx_n^j} \\
\frac{d^2k(x_m, x_n)}{dx_m^i dx_m^j} &= \frac{d^2k(x_m, x_n)}{dx_n^i dx_n^j} \\
\frac{d^3k(x_m, x_n)}{dx_n^i dx_n^j dx_m^k} &= \sigma_f^2 \, \frac{-25}{3rl^5} \Big[- rl \left[\delta_{ij}(x_m^k - x_n^k) + \delta_{ik}(x_m^j - x_n^j) + \delta_{jk}(x_m^i - x_n^i)\right] \\
&\qquad + \sqrt{5}(x_m^i - x_n^i)(x_m^j - x_n^j)(x_m^k - x_n^k)\Big] \exp\left[-\frac{\sqrt{5}r}{l}\right] \\
\frac{d^3k(x_m, x_n)}{dx_m^i dx_m^j dx_n^k} &= -\frac{d^3k(x_m, x_n)}{dx_n^i dx_n^j dx_m^k} \\
\frac{d^4k(x_m, x_n)}{dx_n^i dx_n^j dx_m^k dx_m^l} &= \sigma_f^2 \, \frac{25}{3r^3l^6} \Big[r^3 l^2 (\delta_{ij}\delta_{kl} + \delta_{ij}\delta_{jl} + \delta_{il}\delta_{jk}) + \\
&\qquad (\sqrt{5}l + 5r)(x_m^i - x_n^i)(x_m^j - x_n^j)(x_m^k - x_n^k)(x_m^l - x_n^l) - \\
&\qquad \sqrt{5}lr^2 \Big(\delta_{ij}(x_m^k - x_n^k)(x_m^l - x_n^l) + \delta_{ik}(x_m^j - x_n^j)(x_m^l - x_n^l) + \\
&\qquad\quad \delta_{il}(x_m^j - x_n^j)(x_m^k - x_n^k) + \delta_{jk}(x_m^i - x_n^i)(x_m^l - x_n^l) + \\
&\qquad\quad \delta_{jl}(x_m^i - x_n^i)(x_m^k - x_n^k) + \delta_{kl}(x_m^i - x_n^i)(x_m^j - x_n^j) \Big) \Big] \\
&\qquad \exp\left[-\frac{\sqrt{5}r}{l}\right]
\end{aligned} \tag{A.10}$$

A.4 Maximum Likelihood Estimation

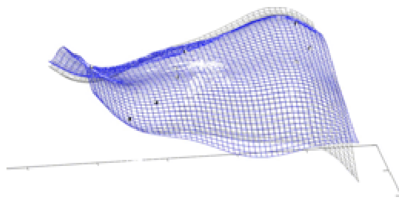
In this section the maximum likelihood estimation is shortly reviewed as a method to optimize the hyperparameters in GPR. The likelihood describes the probability of a set of observations given the parameters and hyperparameters of the model. Therefore, it seems plausible to make use of this property to optimize the hyperparameters of the model. Specifically, one can use $p(\mathbf{y}|X)$. It is the likelihood of the prediction marginalized over the function values (not the observed function values but the "real" values of the underlying function). Marginalization just means that the probability distribution is averaged over the information about the function values. This is done by integrating over the likelihood of the prediction times the prior.

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X) \tag{A.11}$$

The prior, $\mathbf{f}|X \sim \mathcal{N}(\bar{y}_{\text{pr}}, K)$, as well as the likelihood, $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathbb{1})$, are described by Gaussian probability distributions. The logarithm of the marginal likelihood is given by [67]

$$\log\left[p(\mathbf{y}|X)\right] = -\frac{1}{2}\left(\mathbf{y}-\bar{\mathbf{y}}\right)^T (K)^{-1}\left(\mathbf{y}-\bar{\mathbf{y}}\right) - \frac{1}{2}\log|K| - \frac{n}{2}\log\left[2\pi\right] \tag{A.12}$$

where $K = k(\mathbf{x}_i, \mathbf{x}_j) + \delta_{ij}\sigma_n^2$ and $|K|$ is the determinant of K. Note that the marginal likelihood is also a function of the hyperparameters, the parameters of the covariance function, i.e. the length scale l and the noise parameter σ. The first term of equation (A.12), $-\left(\mathbf{y}-\bar{\mathbf{y}}\right)^T (K)^{-1}\left(\mathbf{y}-\bar{\mathbf{y}}\right)/2$ describes the fitting quality to the training data. The larger this term is (the closer to zero since it is negative) the better the training data is fitted. One could think about optimizing the hyperparameters by considering only this term. And in fact the training data will be fitted better and better. But this will lead to strong overfitting, no generalization to further test points is possible. The second term of equation (A.12), $-\log|K|/2$, counters this behavior: The expression $\log|K|/2$ is a complexity penalty that punishes complex models. A complex model would be for example one with a very low length scale that fits training points perfectly but only very close to them. The most simplest model would be a linear or even constant model. This term will become larger for more complex models. Including the minus sign it pushes simpler



models to a higher marginal likelihood. The last term, $\log[2\pi]$, is a normalization constant.

12 Bibliography

[1] Abramowitz, M., and Stegun, I. A. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, vol. 55. Dover Publications, Inc., New York, NY, USA, 1974.

[2] Adler, R. J. *The Geometry of Random Fields.* John Wiley & Sons, Ltd, Hoboken, New Jersey, 1981.

[3] Ahmad, A., Hashim, U. K. M., Mohd, O., Abdullah, M. M., Sakidin, H., Rasib, A. W., and Sufahani, S. F. Comparative analysis of support vector machine, maximum likelihood and neural network classification on multispectral remote sensing data. *Int. J. Adv. Comput. Sci. Appl. 9*, 9 (2018), 529–537.

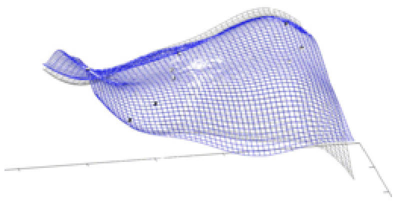
[4] Anjyo, K., and Lewis, J. RBF interpolation and Gaussian process regression through an RKHS formulation. *J. Math. Ind 3*, 6 (2011), 63–71.

[5] Aronszajn, N. Theory of reproducing kernels. *Trans. Amer. Math. Soc. 68*, 3 (1950), 337–404.

[6] Ayala, P. Y., and Schlegel, H. B. A combined method for determining reaction paths, minima, and transition state geometries. *J. Chem. Phys. 107*, 2 (1997), 375–384.

[7] Baker, J. An algorithm for the location of transition states. *J. Comput. Chem. 7*, 4 (1986), 385–395.

[8] Baker, J., and Chan, F. The location of transition states: A comparison of cartesian, z-matrix, and natural internal coordinates. *J. Comput. Chem. 17*, 7 (1996), 888–904.



[9] Baker, J., and Hehre, W. J. Geometry optimization in cartesian coordinates: The end of the z-matrix? *J. Comput. Chem. 12*, 5 (1990), 606–610.

[10] Banerjee, A., Adams, N., Simons, J., and Shepard, R. Search for stationary points on surfaces. *J. Phys. Chem. 89*, 1 (1985), 52–57.

[11] Bartók, A. P., Kondor, R., and Csányi, G. On representing chemical environments. *Phys. Rev. B 87*, 18 (2013), 184115.

[12] Becke, A. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A 38*, 6 (1988), 3098–3100.

[13] Behler, J. Perspective: Machine learning potentials for atomistic simulations. *J. Chem. Phys. 145*, 17 (2016), 170901.

[14] Behn, A., Zimmerman, P. M., Bell, A. T., and Head-Gordon, M. Efficient exploration of reaction paths via a freezing string method. *J. Chem. Phys. 135*, 22 (2011), 224108.

[15] Berlinet, A., and Thomas-Agnan, C. *Reproducing Kernel Hilbert Spaces in Probability and Statistics.* Springer Science, New York, NY, USA, 2011.

[16] Billingsley, P., Ed. *Probability and Measure.* John Wiley & Sons, Ltd, Hoboken, New Jersey, 1979.

[17] Bofill, J. M. Updated Hessian matrix and the restricted step method for locating transition structures. *J. Comput. Chem. 15*, 1 (1994), 1–11.

[18] Broyden, C. G. The convergence of a class of double-rank minimization algorithms 1. General considerations. *J. Inst. Maths. Appl. 6*, 1 (1970), 76–90.

[19] Burger, S. K., and Yang, W. Quadratic string method for determining the minimum-energy path based on multiobjective optimization. *J. Chem. Phys. 124*, 5 (2006), 054109.

[20] Cawley, G. C., and Talbot, N. L. C. Reduced rank kernel ridge regression. *Neural Process. Lett. 16*, 3 (2002), 293–302.

[21] Chaffey-Millar, H., Nikodem, A., Matveev, A. V., Krüger, S., and Rösch, N. Improving upon string methods for transition state discovery. *J. Chem. Theory Comput. 8*, 2 (2012), 777–786.

[22] Christakos, G. *Random Field Models in Earth Sciences.* Academic Pr., San Diego, 1992.

[23] Cortes, C., and Vapnik, V. Support vector machine. *Mach. Learn. 20*, 3 (1995), 273–297.

[24] Denzel, A., Haasdonk, B., and Kästner, J. Gaussian process regression for minimum energy path optimization and transition state search. *J. Phys. Chem. A 123*, 44 (2019), 9600–9611.

[25] Denzel, A., and Kästner, J. Gaussian process regression for geometry optimization. *J. Chem. Phys. 148*, 9 (2018), 094114.

[26] Denzel, A., and Kästner, J. Gaussian process regression for transition state search. *J. Chem. Theory Comput. 14* (2018), 5777–5786.

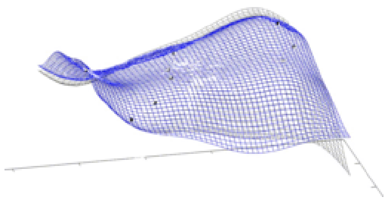
[27] Dewar, M. J., Zoebisch, E. G., Healy, E. F., and Stewart, J. J. Development and use of quantum mechanical molecular models. 76. AM1: a new general purpose quantum mechanical molecular model. *J. Am. Chem. Soc. 107*, 13 (1985), 3902–3909.

[28] E, W., Ren, W., and Vanden-Eijnden, E. String method for the study of rare events. *Phys. Rev. B 66*, 5 (2002), 052301.

[29] E, W., Ren, W., and Vanden-Eijnden, E. Simplified and improved string method for computing the minimum energy paths in barrier-crossing events. *J. Chem. Phys. 126*, 16 (2007), 164103.

[30] Elstrodt, J. *Maß- und Integrationstheorie.* Springer, Berlin, Germany, 2018.

[31] Fernandez-Ramos, A., Ellingson, B. A., Garrett, B. C., and Truhlar, D. G. In *Reviews in Computational Chemistry*, vol. 23. John Wiley & Sons, Ltd, Hoboken, New Jersey, 2007, pp. 125–232.



[32] Fletcher, R. A new approach to variable metric algorithms. *Comp. J. 13*, 3 (1970), 317–322.

[33] Fletcher, R. *Practical Methods of Optimization*. John Wiley & Sons, Ltd, Hoboken, New Jersey, 2003.

[34] Goldfarb, D. A family of variable-metric methods derived by variational means. *Math. Comp. 24*, 109 (1970), 23–26.

[35] Halgren, T. A., and Lipscomb, W. N. The synchronous-transit method for determining reaction pathways and locating molecular transition states. *Chem. Phys. Lett. 49*, 2 (1977), 225–232.

[36] Hansen, K., Biegler, F., Ramakrishnan, R., Pronobis, W., von Lilienfeld, O. A., Müller, K.-R., and Tkatchenko, A. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *J. Phys. Chem. Lett. 6*, 12 (2015), 2326–2331.

[37] Hariharan, P. C., and Pople, J. A. The influence of polarization functions on molecular orbital hydrogenation energies. *Theor. Chem. Acc. 28*, 3 (1973), 213–222.

[38] Henkelman, G., and Jónsson, H. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. *J. Chem. Phys. 111*, 15 (1999), 7010–7022.

[39] Henkelman, G., Uberuaga, B. P., and Jónsson, H. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *J. Chem. Phys. 113*, 22 (2000), 9901–9904.

[40] Heyden, A., Bell, A. T., and Keil, F. J. Efficient methods for finding transition states in chemical reactions: Comparison of improved dimer method and partitioned rational function optimization method. *J. Chem. Phys. 123*, 22 (2005), 224101.

[41] Jónsson, H., Mills, G., and Jacobsen, K. W. In *Classical and Quantum Dynamics in Condensed Phase Simulations*. World Scientific, Singapore, 1998, pp. 385–404.

[42] Kästner, J., Carr, J. M., Keal, T. W., Thiel, W., Wander, A., and Sherwood, P. DL-FIND: an open-source geometry optimizer for atomistic simulations. *J. Phys. Chem. A 113*, 43 (2009), 11856–11865.

[43] Kästner, J., and Sherwood, P. Superlinearly converging dimer method for transition state search. *J. Chem. Phys. 128*, 1 (2008), 014106.

[44] Koistinen, O.-P., Dagbjartsdóttir, F. B., Ásgeirsson, V., Vehtari, A., and Jónsson, H. Nudged elastic band calculations accelerated with gaussian process regression. *J. Chem. Phys. 147*, 15 (2017), 152720.

[45] Koistinen, O.-P., Maras, E., Vehtari, A., and Jónsson, H. Minimum energy path calculations with gaussian process regression. *Nanosystems: Phys. Chem. Math. 7*, 6 (2016), 925–935.

[46] Krige, D. A statistical approach to some basic mine valuation problems on the Witwatersrand. *J. South. Afr. Inst. Min. Metall. 52*, 6 (1951), 119–139.

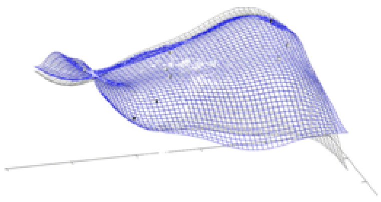
[47] Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand. B 45* (1950), 255–282.

[48] Liu, D. C., and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program. 45*, 1 (1989), 503–528.

[49] Liu, H., Lu, Z., Cisneros, G. A., and Yang, W. Parallel iterative reaction path optimization in ab initio quantum mechanical/molecular mechanical modeling of enzyme reactions. *J. Chem. Phys. 121*, 2 (2004), 697–706.

[50] Loève, M. *Probability theory*, 3. ed. Van Nostrand, Princeton, NJ, 1963.

[51] Lu, D., Truong, T. N., Melissas, V. S., Lynch, G. C., Liu, Y., Garrett, B. C., Steckler, R., Isaacson, A. D., Rai, S. N., Hancock, G. C., Lauderdale, J. G., Joseph, T., and Truhlar, D. G.



POLYRATE 4: A new version of a computer program for the calculation of chemical reaction rates for polyatomics. *Comput. Phys. Commun. 71*, 3 (1992), 235–262.

[52] Matérn, B. *Spatial Variation.* Springer, Berlin, Germany, 1986.

[53] Meisner, J., and Kästner, J. Dual-level approach to instanton theory. *J. Chem. Theory Comput. 14*, 4 (2018), 1865–1872.

[54] Meisner, J., Markmeyer, M. N., Bohner, M. U., and Kästner, J. Comparison of classical reaction paths and tunneling paths studied with the semiclassical instanton theory. *Phys. Chem. Chem. Phys. 19*, 34 (2017), 23085–23094.

[55] Mills, G., and Jónsson, H. Quantum and thermal effects in h_2 dissociative adsorption: Evaluation of free energy barriers in multidimensional quantum systems. *Phys. Rev. Lett. 72* (1994), 1124–1127.

[56] Müller, K., and Brown, L. D. Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theor. Chem. Acc. 53*, 1 (1979), 75–93.

[57] Nocedal, J. Updating quasi-Newton matrices with limited storage. *Math. Comput. 35*, 151 (1980), 773–773.

[58] Ogorodnikov, V., and Prigarin, S. *Numerical Modelling of Random Processes and Fields: Algorithms and Applications.* De Gruyter, Berlin, Germany, 2018.

[59] Pedersen, A., Hafstein, S. F., and Jónsson, H. Efficient sampling of saddle points with the minimum-mode following method. *SIAM J. Sci. Comput. 33*, 2 (2011), 633–652.

[60] Perdew, J. P. Density-functional approximation for the correlation energy of the inhomogeneous electron gas. *Phys. Rev. B 33*, 12 (1986), 8822–8824.

[61] Peters, B., Heyden, A., Bell, A. T., and Chakraborty, A. A growing string method for determining transition states: Comparison to the

nudged elastic band and string methods. *J. Chem. Phys. 120*, 17 (2004), 7877–7886.

[62] Plasencia Gutiérrez, M., Argáez, C., and Jónsson, H. Improved minimum mode following method for finding first order saddle points. *J. Chem. Theory Comput. 13*, 1 (2017), 125–134.

[63] Plessow, P. Reaction path optimization without NEB springs or interpolation algorithms. *J. Chem. Theory Comput. 9*, 3 (2013), 1305–1310.

[64] Powell, M. J. D. Recent advances in unconstrained optimization. *Math. Program. 1*, 1 (1971), 26–57.

[65] Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Big data meets quantum chemistry approximations: The Δ-machine learning approach. *J. Chem. Theory Comput. 11*, 5 (2015), 2087–2096.

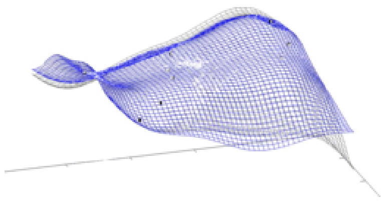
[66] Ramakrishnan, R., and von Lilienfeld, O. A. In *Reviews in Computational Chemistry*. John Wiley & Sons, Ltd, Hoboken, New Jersey, 2017, pp. 225–256.

[67] Rasmussen, C. E., and Williams, C. K. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA, 2006.

[68] Rieckhoff, S., Meisner, J., Kästner, J., Frey, W., and Peters, R. Double regioselective asymmetric C-allylation of isoxazolinones: Iridium-catalyzed N-allylation followed by an Aza-Cope rearrangement. *Angew. Chem. Int. Ed. 57*, 5 (2018), 1404–1408.

[69] Saitoh, S. Quadratic inequalities associated with integrals of reproducing kernels. *Linear Algebra Its Appl. 101* (1988), 269–280.

[70] Schmitz, G., and Christiansen, O. Gaussian process regression to accelerate geometry optimizations relying on numerical differentiation. *J. Chem. Phys. 148*, 24 (2018), 241704.



[71] Schölkopf, B., Smola, A. J., Bach, F., et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[72] Sen, S., Frey, W., Meisner, J., Kästner, J., and Buchmeiser, M. R. An anionic molybdenum amidato bisalkyl alkylidyne complex. *J. Organomet. Chem. 799-800* (2015), 223–225.

[73] Shanno, D. F. Conditioning of quasi-Newton methods for function minimization. *Math. Comp. 24*, 111 (1970), 647–656.

[74] Shepard, D. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference* (New York, NY, USA, 1968), ACM, pp. 517–524.

[75] Sheppard, D., Terrell, R., and Henkelman, G. Optimization methods for finding minimum energy paths. *J. Chem. Phys. 128*, 13 (2008), 134106.

[76] Smidstrup, S., Pedersen, A., Stokbro, K., and Jónsson, H. Improved initial guess for minimum energy path calculations. *J. Chem. Phys. 140*, 21 (2014), 214106.

[77] Stein, M. L. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, Luxembourg City, Luxembourg, 2012.

[78] Truhlar, D. G., Isaacson, A. D., Skodje, R. T., and Garrett, B. C. Incorporation of quantum effects in generalized-transition-state theory. *J. Phys. Chem. A 86*, 12 (1982), 2252–2261.

[79] Vaucher, A. C., and Reiher, M. Minimum energy paths and transition states by curve optimization. *J. Chem. Theory Comput. 14*, 6 (2018), 3091–3099.

[80] Weigend, F., and 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* (2005), 3297–3305.

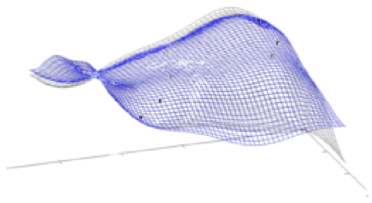
[81] Wolfe, P. Convergence conditions for ascent methods. *SIAM Review 11*, 2 (1969), 226–235.

[82] Wolfe, P. Convergence conditions for ascent methods. II: Some corrections. *SIAM Review 13*, 2 (1971), 185–188.

[83] Zeng, Y., Xiao, P., and Henkelman, G. Unification of algorithms for minimum mode optimization. *J. Chem. Phys. 140*, 4 (2014), 044115.

[84] Zheng, J., and Frisch, M. J. Efficient geometry minimization and transition structure optimization using interpolated potential energy surfaces and iteratively updated hessians. *J. Chem. Theory Comput. 13*, 12 (2017), 6424–6432.

[85] Zhu, X., Thompson, K. C., and Martínez, T. J. Geodesic interpolation for reaction pathways. *J. Chem. Phys. 150*, 16 (2019), 164103.



Erklärung über die Eigenständigkeit der Dissertation

Ich versichere, dass ich die vorliegende Arbeit mit dem Titel *Chemical structure prediction based on machine learning* selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe; aus fremden Quellen entnommene Passagen und Gedanken sind als solche kenntlich gemacht.

Declaration of Authorship

I hereby certify that the dissertation entitled *Chemical structure prediction based on machine learning* is entirely my own work except where otherwise indicated. Passages and ideas from other sources have been clearly indicated.

Name/Name:	Alexander Denzel
Unterschrift/Signed:	
Datum/Date:	January 29, 2020