# Machine-Learning Techniques for Geometry Optimization

Von der Fakultät Chemie der Universität Stuttgart
und dem Stuttgart Center for Simulation Science
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Daniel Born

aus Stuttgart

| | |
|---|---|
| **Hauptberichter:** | Prof. Dr. Johannes Kästner |
| **Mitberichter:** | Prof. Dr. Reinhold Fink |
| **Prüfungsvorsitzender:** | Prof. Dr. Blazej Grabowski |
| **Tag der mündlichen Prüfung:** | 15. Mai 2023 |

Institut für Theoretische Chemie
Universität Stuttgart

2023

# Abstract

The Born–Oppenheimer (BO) approximation defines the concept of the potential energy surface (PES) and molecular geometries. The PES, a high-dimensional function of all atomic coordinates, is obtained from the electronic Schrödinger equation (SE). Calculating single points on the PES from ab initio methods is computationally expensive. Therefore, stationary points, which have a vanishing gradient, are usually of interest. Minima and first-order saddle points are the most interesting points. A minimum on the PES corresponds to a stable configuration or geometry of a molecule, while a first-order saddle point corresponds to a transition state (TS), which is the maximum of the minimum energy path connecting two minima.

The search for minima, or geometry optimization, and the search for first-order saddle points, TS search, require quite an amount of ab initio calculations to converge to those points, starting from an initial guess. For the most part, Cartesian coordinates are used, because they are easily constructed. However, they are usually highly coupled. For example consider a $CH_3$ end group, which is bound to a rest R. Rotating around the C-R bond requires all 9 Cartesian coordinates of the $CH_3$ group to move, instead of one coordinate, the angle between R-C-H. This is inefficient and thus internal coordinates are introduced. They consist of bond lengths, bond angles, and dihedrals. Once created, they reduce the coupling between the coordinates. In addition, they are invariant under translation and rotation. This reduces the total amount of ab intio calculations.

Another way to reduce the amount of ab initio calculations is the idea of a surrogate surface for the PES. The key idea of a surrogate surface is to use a few points of the PES and fit the surrogate surface to these points. On the surrogate surface, geometry optimization or TS search can be performed. These surrogate surfaces are usually linear combinations of basis functions, for example, polynomials. To get additionally a measure of uncertainty, Gaussian process regression (GPR) can be used. Here, a linear combination of covariance functions is calculated to build a surrogate surface. An advantage of GPR is that derivative information, which is easily obtained from ab initio calculations, can be included to get a better fit to the PES. This GPR surface is then used for geometry optimization or TS search. Starting with the initial geometry, the GPR surface is iteratively improved until the convergence criteria are fulfilled, which reduces the total amount of ab initio calculations.

Combining GPR and internal coordinates reduces the amount of ab initio calculations further. For geometry optimization, there are two ways to combine them. The first approach is to use the internal coordinates to evaluate the covariance function and to build the GPR surrogate. The optimization on the GPR surrogate is then performed in internal coordinates and therefore, the predicted new geometry is in internal coordinates. Thus, they have to be transformed back to Cartesian coordinates. This procedure is repeated until the minimum on the PES is found. The second approach uses the chain rule, where the covariance function is evaluated in internal coordinates and then "transformed" to Cartesian coordinates to build the GPR surrogate. Thus, the GPR-surrogate surface is in Cartesian coordinates. With this approach, no back-transformation is required, because the predicted geometry is in Cartesian coordinates. Both approaches reduce the amount of ab initio calculations compared to standard algorithms.

Finally, GPR and internal coordinates are combined for the TS search. In contrast to geometry optimization, the second approach drops out due to the additional computational overhead. Thus, only the first approach remains. To perform a TS search on the GPR surrogate a good GPR-Hessian is required. Additional to a dimer rotation to sample points for the GPR-Hessian, an alternative approach is presented. This approach uses an information-based acquisition function to sample the points for the GPR-Hessian. However, the back transformation from internal coordinates to Cartesian coordinates is more prone to error for TS search compared to geometry optimization. To overcome this problem an alternative way to construct the internal coordinates, namely only bond length between all atoms, is used.

Overall the thesis shows, that combining GPR and internal coordinates improves the overall performance of geometry optimization and TS search based on GPR. The obtained results outperform standard algorithms.

# Zusammenfassung

Mit Hilfe der Born-Oppenheim-Näherung können die Begriffe der Energiehyperfläche und der Molekülgeometrie definiert werden. Diese Begriffe sind von zentraler Bedeutung in der rechnergestützten Chemie. Eine Energiehyperfläche is eine mehrdimensionale Funktion aller atomaren Koordinaten. Sie wird durch das Lösen der elektronischen Schrödingergleichung bestimmt. Da die Berechnung von einzelnen Punkten auf der Energiehyperfläche teuer ist, sind nur Punkte interessant, deren Ableitungen nach den Koordinaten verschwinden. Die interessantesten Punkte sind Minima und Sattelpunkte erster Ordnung. Ein Minimum auf der Energiehyperfläche entspricht einer stabilen Molekülgeometrie und ein Sattelpunkt erster Ordung einem Übergangszustand.

Beginnend mit einer Anfangsgeometrie braucht eine Geometrieoptimierung oder eine Übergangszustandssuche viele Berechunung von ab-initio-Rechnungen um zu stationären Punkten zu konvergieren. Meistens werden dabei kartesische Koordinaten verwendet, da sie einfach zu erzeugen sind. Allerdings sind Kartesische Koordinaten untereinander stark gekoppelt. Das kann man zum Beispiel an der Rotation einer $CH_3$-Gruppe um eine Bindung erkennen. Anstatt nur die Bindung zu rotieren, müssen alle 9 Koordinaten der $CH_3$-Gruppe bewegt werden. Das ist ineffizient und wird durch Verwendung von internen Koordinaten behoben. Die internen Koordinaten bestehen aus Bindungslängen, Bindungswinkel und Diederwinkeln, welche die Kopplung zwischen den Koordinaten verringert. Zusätzlich sind diese unveränderlich bezüglich einer Parallelverschiebung und einer Drehung des Moleküls.

Weiterhin kann man Ersatzflächen der Energiehyperfläche verwenden. Diese verwenden wenige Punkte der Energiehyperfläche um diese dann durch Ausgleichsrechungen als Linearkombination von Basisfunktionen darzustellen um dann auf diesen Ersatzflächen eine Geometrieoptimierung oder Übergangszustandssuche durchzuführen. Um zusätzlich ein MaSS für die Unsicherheit zu bekommen, wird Gaußprozessregression verwendet. Dabei werden Linearkombinationen von Kovarianzfunktionen verwendet. Ein weiterer Vorteil der Gaußprozessregression ist, dass Informationen über die Ableitung mit eingebaut werden können. Das liefert eine bessere Ersatzfläche für die Energiehyperfläche. Somit kann diese Ersatzfläche, die mit Hilfe von Gaußprozessregression bestimmt wurde, für die Geometrieoptimierung und Übergangszustandssuche verwendet werden. Dafür braucht man eine Anfangsgeometrie. Itereativ wird dann die Ersatzfläche verbessert bis ein stationärer Punkt auf der Energiehyperfläche gefunden wurde.

Interne Koordinaten und Gaußprozessregression können dann kombiniert werden. Für Geometrieoptimierung gibt es zwei Möglichkeiten. Die erste berechnet die Kovarianz in internenen Koordinaten und bekommt dann eine Gaußprozessregressionersatzfläche in internen Koordinaten. Eine Geometrieoptimierung auf der Ersatzfläche liefert eine Vorhersage in internen Koordinaten, die zurück in kartesiche Koordinaten transformiert werden muss. Dies wird solange wiederhohlt, bis ein Minimum auf der Energiehyperfläche gefunden wurde. Der zweite Weg wertet die Kovarianzfunktion ebenfalls in internen Koordianten aus, verwendet die Kettenregel um eine "Transformation "in kartesiche Koordinaten zu bekommen. Daher ist die Ersatzfläche in kartesischen Koordinaten. Eine Geometrieoptimierung auf dieser Ersatzfläche liefert dann eine Vohersage in kartesischen Koordinaten. Hierbei wird die Rücktransformation umgangen. Beide Methoden verringern die Gesammtzahl an ab-intio Rechnungen im Vergleich zu Standardoptimierern.

Schließlich wird Gaußprozessregression mit internen Koordinaten für die Übergangszustandssuche verknüpft. Im Gegensatz zur Geometrieoptimierung, kann der zweite Ansatz nicht verwendet werden, da zusätzliche Rechenzeit benötigt wird. Daher bleibt nur der erste Ansatz übrig. Die Hessematrix der Gaußprozessersatzfläche muss gut genug sein um auf der Ersatzfläche eine Übergangszustandssuche auszuführen. Zusätzlich zur Dimerroation um gute Punkte auszuwählen, wird eine alternativer Ansatz verwendet. Dieser Ansatz nutzt eine Erwerbsfunktion auf Basis der Informationstheorie um gute Punkte auszuwählen. Nachdem genug Punkte gesammelt wurden, kann eine Übergangszustandssuche auf der Gaußprozessersatzfläche statt finden. Die vorhergesagten Geometrien sind in internen Koordinaten und müssen zurücktransformiert werden. Diese Rücktransformation ist allerdings im Gegensatz zur Geometrieoptimierung fehleranfällig. Um das zu beheben kann man die internen Koordinaten nur aus Bindungslängen bestimmen.

Am Ende zeigt in dieser Arbeit, dass die Kombination von Gaußprozessregresion und internen Koordinate die Ergebnisse der Geometrieoptimierung und Übergangszustandssuche auf der Basis der Gaußprozessregression deutlich verbessert. Die erhaltenen Ergebnisse sind besser als Standardalgorithmen.

# Peer-reviewed publications

[1]: D. Born and J. Kästner: *Geometry Optimization in Internal Coordinates Based on Gaussian Process Regression: Comparison of Two Approaches*. Journal of Chemical Theory and Computation **17** (9), 5955–5967 (2021)

Other publications by the author, not included in this thesis

- C. A. Dietrich, R. Schuldt, D. Born, H. Solodenko, G. Schmitz and J. Kästner: *Evaporation and Fragmentation of Organic Molecules in Strong Electric Fields Simulated with DFT*. The Journal of Physical Chemistry A **124**, 3131–3145 (2020)

# Contents

# Danksagung

Es gibt viele Personen, denen ich mein Dank ausdrücken möchte und die mich während meiner Promotion unterstüzt haben.

Als Erstes möchte ich Johannes Kästner für die Möglichkeit danken, mit allen Freiheiten an meinem Thema arbeiten zu dürfen. Bei den regelmäßigen Treffen konnte ich nicht nur wissenschaftliche Themen beprechen, sondern auch private Probleme ansprechen. Seine Erfahrung und wissenschaftliche Expertise waren von essentieller Bedeutung während meiner Promotion. Ich möchte mich auch bei Reinhold Fink bedanken mein Zweitprüfer zu sein. Desweiteren möchte ich Blazej Grabowski danken, dass er den Prüfungsvorsitz übernimmt. Für die finazielle Unterstüzung möchte ich mich bei SimTech bedanken: „Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech).“

Weiterhin möchte ich meinen ehemaligen Kollegen danken, insbesondere meinem Büropartner Robin Schuldt. Ohne ihn wäre die Coronaphase kaum durchzustehen gewesen. Bakloni war halt was besonderes, wo nicht nur wissenschaftliche Themen, sondern auch private Themen diskutiert und analysiert wurden. Auch möchte ich mich bei Viktor Zaverkin danken, dafür das Gan-Sao, Pak-Sao und Fok-Sao (und bald auch Tan-Sao) immer besser werden ;). Stuttfahrt-Rohr steht natürlich auch wieder an. Auch möchte ich ihm für wertvolle tips und Anregungen bedanken, insbesondere den Hinweis mit der Woddbury-Idendität. Ich möchte auch Maria Fyta danken, die Zweitprüferin bei meinem Meilenstein gewesen zu sein.

Ich möchte meinen Eltern Wolfram Herbert und Margarete sowie meinem Bruder Patrick für die Unterstüzung während des Studiums und der Promotion danken. Weiteren Dank geht raus an Marvin Poul für die lustigen Abende während des Studiums. Du weißt, Eric Weikum, dass du nicht fehlen darfst. Vielen Dank für die ergiebigen Diskussionen im politisch nicht korrekten Bereich ;) Danke auch für den schwierigen Winter 2021/2022, den ich ohne dich fast nicht überstanden hätte.

Ich möchte MVP Soeren danken für das legendäre Flunkyball Tunier und dafür, dass du meine Arbeit gelesen hast. Desweiteren möchte ich Alexander Waigum dafür danken, dass er immer ein offenes Ohr hatte und die jüngsten sportlichen Aktivitäten. Und auch danke für das Lesen der Arbeit. Weiteren Dank gehen an Katrin Gugeler und Juliane Heitkämper für das Lesen einzelner Kapitel meiner Arbeit.

Außeruniversitär möchte ich Denis Möller und Eric Tänzer für die vielen Auswärtsfahrten danken, die mich auf den Boden der Tatsachen zurückgeholt haben.

Am Ende möchte ich Sina Klostermann danken für ihre Ausdauer mein ständiges Genörgel und Gemecker während dem Schreiben dieser Arbeit auszuhalten. Mau :*

# List of Abbreviations

| | |
|---|---|
| **BFGS** | Broyden–Fletcher–Goldfarb–Shanno |
| **BO** | Born–Oppenheimer |
| **DLC** | Delocalized internal coordinates |
| **EI** | Expected improvement |
| **GP** | Gaussian process |
| **GPR** | Gaussian process regression |
| **GPRTS** | Gaussian process regression for transition state search |
| **GPRTS-PES** | Gaussian process regression for transition state search based on predictive entropy search |
| **KL** | Kullback-Leibler |
| **L-BFGS** | Limited memory Broyden–Fletcher–Goldfarb–Shanno |
| **P-RFO** | Partitioned rational function optimization |
| **PES** | Potential energy surface |
| **POI** | Probability of improvement |
| **SCF** | Self-consistent field |
| **SE** | Schrödinger equation |
| **SVD** | Singular value decomposition |
| **tc** | Total connection scheme |
| **TS** | Transition state |
| **TST** | Transition state theory |

# 1 Introduction

In computational chemistry, one of the fundamental tasks is the explanation and prediction of chemical phenomena, like the investigation of a reaction mechanism. On the fundamental level, the solution of the Schrödinger equation (SE) [2] explains all phenomena. However, the SE is a partial differential equation and thus is difficult to solve, if more than two particles are present. For example, there is an analytic solution to the hydrogen atom and hydrogen-like atoms, e.g. $He^{+}$. Another example is the dihydrogen cation, $H_2^{+}$, that can be solved analytically because there is no electron-electron interaction present [3]. Thus, approximations have to be introduced. Thus with the Born–Oppenheimer (BO) approximation [4] the time-independent SE is simplified. There, only the electronic part of the SE is solved, because the motion of nuclei is decoupled from the electron movement. The mass difference between nuclei and electrons by over three orders of magnitude justifies this approximation. Within the BO approximation, the solution of the electronic SE for a specific position of the nuclei leads to the potential energy surface (PES) and thus the concept of molecular geometry. A detailed explanation of BO approximation and the PES is given in chapter 2.

However, even with the BO approximation, the solution of the electronic SE is still expensive. Thus, the focus lies on stationary points of the PES. Stationary points have a vanishing gradient, i.e. the derivatives of the energy with respect to all nuclear coordinates are zero. The two most interesting stationary points on the PES are minima and first-order saddle points. A minimum of the PES corresponds to a stable configuration or geometry of a molecule, which can be found in the real physical world. First-order saddle points correspond to transition states (TS). A TS is the highest point of the minimum energy path or reaction path, which connects two minima on the PES. According to transition state theory (TST), the TS can be used to calculate reaction rates [5].

There are three factors, that can influence the performance of finding stationary points. The choice of the optimization algorithm, either for geometry optimization or TS search, is the first factor. Basic algorithms, how to find stationary points on the PES, are presented in chapter 3. There, minimization and saddle point search algorithms are explained briefly.

The next important factor is the choice of coordinates. Up to now geometry optimization and TS search are performed in Cartesian coordinates. In Cartesian coordinates, a molecule is uniquely defined. However, a molecule can be rotated or translated, then the Cartesian coordinates will change, but the energy of the molecule will be the same. To fix this problem internal coordinates to describe the molecule are introduced in chapter 4. The $3N_{\text{at}}$ Cartesian coordinates, where $N_{\text{at}}$ is the number of atoms, are mapped into a $3N_{\text{at}} - 6$ dimensional space. The molecules can be described by bond length, bond angles, and dihedrals [6, 7]. This set of internal coordinates leads to a non-redundant set of coordinates Since the gradient and eventually, the Hessian has to be transformed into internal coordinates, the chain rule has to be applied [8]. To reduce the redundancy, the delocalized internal coordinates are introduced [9]. However, the transformation is not linear and the matrix containing the derivatives of the Cartesian coordinates with respect to internal coordinates is rectangular. Therefore, no direct matrix inversion can be used and a pseudo-inverse has to be used. Having transformed the gradient and eventually, the Hessian transformed properly, a reduction of the total amount of energy evaluations from ab initio programs can be achieved. In chapter 4 also difficulties for the back-transformation from internal coordinates to Cartesian coordinates are explained.

The third factor, that influences the optimization procedure is the quality of the obtained energies and gradients from the ab initio program. The computational cost for one gradient and one energy evaluation is comparable. However, the Hessian calculation is much more expensive. These calculations are the bottleneck of optimization, and thus the goal is to reduce the total amount of these calculations.

Machine learning (ML) techniques, like Gaussian process regression (GPR) [10] are widely used in computational chemistry [11–14]. They have been applied for example to geometry optimization [15–18] or TS search [19–25], where GPR as ML technique is used. However, reinforcement learning with Markow decision processes is used, too [26]. To get a better understanding GPR is explained in chapter 5. GPR is used to fit the data obtained from the ab initio calculations to get a surrogate surface of the PES. Then the minimum or saddle point is located there, followed by a next ab initio calculation. Since GPR can approximate the function with few training points. Therefore, GPR is useful for geometry optimization and TS search, where only one training point, in the beginning, is available.

The main part of the thesis was the development of methods to combine GPR and internal coordinates, as described in part II, "Algorithms Based on Gaussian Process Regression in Internal Coordinates". In chapter 6 the results of combining GPR and internal coordinates for geometry optimization are presented. First, a general introduction to how geometry optimization based on GPR works, based on earlier works by Denzel et al. [15]. The inputs for the

GPR are the coordinates of a molecule, while the labels or outputs are energies and gradients. This is followed by two approaches, how to combine GPR with internal coordinates. For the first approach, the GPR-surrogate is built in internal coordinates and thus predictions are in internal coordinates. The second approach transforms the internal coordinates via chain rule to Cartesian coordinates. Thus a GPR surrogate in Cartesian is obtained, where predictions are then in Cartesian coordinates. In this chapter, ways to improve geometry optimization based on GPR are presented. However, they do not give better results than using the minimum on the GPR surrogate as a prediction.

The two approaches for combining GPR and internal coordinates for geometry optimization might lead to the conclusion to extend both methods for TS search. However, the additional computational overhead for the second approach is too high. Therefore, the focus relies on the first approach. This is presented in chapter 7. In contrast to geometry optimization, TS search is more complicated and requires usually the Hessian or at least a good approximation. Using only energies and gradients, a way to sample training points to describe the Hessian is required. In addition to a dimer rotation on the GPR surrogate, an information-based selection scheme is developed.

# Part I

# Theory

# 2 Basics of Quantum Chemistry

In this chapter, the basics of quantum chemistry are reviewed. Beginning with the molecular Hamiltonian, the approximation of Born–Oppenheimer (BO) [4] is introduced. This is done in section 2.1. The derivation follows Jensen [27].

With this approximation, the potential energy surface (PES) can be constructed. Exploring the PES leads to stationary points, which have a physical meaning. The PES is introduced in section 2.2.

## 2.1 The Born–Oppenheimer Approximation

The time-independent Schrödinger equation is

$$\hat{H}\Psi(\mathbf{x},t) = E\Psi(\mathbf{x},t) \tag{2.1}$$

or in Bra-Ket notation:

$$\hat{H}\,|\Psi\rangle = E\,|\Psi\rangle \tag{2.2}$$

It is described by the Hamilton operator, which has for point particles the form

$$\hat{H} = \hat{T} + \hat{V}. \tag{2.3}$$

which contains the kinetic energy operator $\hat{T}$ and the potential energy operator $\hat{V}$. The kinetic operator is $\frac{\hat{p}_k^2}{2m_k}$ for particle k with mass $m_k$, where $\hat{p}_{k,l} = \frac{1}{\mathrm{i}}\frac{\partial}{\partial x_{k,l}}$, in atomic units, is the momentum operator acting on the $l$-th dimension of particle $k$. A molecular system contains $N_{\text{at}}$ nuclei with position $\mathbf{R}_K,\ K = 1, \cdots, N_{\text{at}}$ and $n$ electrons with position $\mathbf{r}_i,\ i = 1, \cdots, n$. The interaction between all particles is the Coulomb interaction, in atomic units, $V(r) = \frac{q_1 q_2}{r}$, where $r$ is the distance between particle 1 and 2, and $q_1$ is the charge of particle 1 and $q_2$ of particle 2.

Then the total molecular Hamiltonian $\hat{H}$ consists of the kinetic energy of the nuclei $\hat{T}_N$ and electrons $\hat{T}_e$, as well as the interaction between the nuclei $\hat{V}_{KK}$, the electrons $\hat{V}_{ee}$ and between nuclei and electrons $\hat{V}_{eK}$ and can be written in atomic units:

$$\hat{H} = \underbrace{-\sum_{K=1}^{N_{\text{at}}} \frac{\nabla_K^2}{2m_K}}_{\hat{T}_N} \underbrace{-\sum_{i=1}^{n} \frac{\nabla_{i,e}^2}{2}}_{\hat{T}_e} + \underbrace{\sum_{K\neq M} \frac{Z_K Z_M}{|\mathbf{R}_K - \mathbf{R}_M|}}_{\hat{V}_{KK}} + \underbrace{\sum_{i\neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}}_{\hat{V}_{ee}} \underbrace{-\sum_{i,K} \frac{Z_K}{|\mathbf{R}_K - \mathbf{r}_i|}}_{\hat{V}_{Ke}} . \tag{2.4}$$

The coordinates of the nuclei $K$ are $\mathbf{R}_K$ with mass $m_K$ and the position of the electron $i$ is $\mathbf{r}_i$. The charge of the nuclei is given as $Z_K$. Due to the mass difference between electrons and protons, $m_e/m_p \sim 10^{-4}$, here $m_p$ is the mass of a single proton and $m_e$ the mass of an electron , the coupling between the motion of nuclei and electrons can be neglected.

Mass polarization should be added because the motion of the center of mass and the internal motion cannot be separated if more than two particles are present. However, this term is neglected here.

The electronic Hamiltonian only depends parametric on the position of the nuclei (through $\hat{V}_{Ke}$ and $\hat{V}_{KK}$), but not on the momenta, i.e. derivatives with respect to Cartesian coordinates.

$$\hat{H}_{el}(\mathbf{R}, \mathbf{r}, \partial_{\mathbf{r}}) = \hat{H} - \hat{T}_N. \tag{2.5}$$

where $\mathbf{R} = (\mathbf{R}_1, \cdots, \mathbf{R}_{N_{\text{at}}})$ is the position of the nuclei and $\mathbf{r} = (\mathbf{r}_1, \cdots, \mathbf{r}_n)$ are the coordinates of the electrons. Thus, the time-independent electronic SE is

$$\hat{H}_{el}(\mathbf{R})\Psi_i(\mathbf{R}, \mathbf{r}) = E_i(\mathbf{R})\Psi_i(\mathbf{R}, \mathbf{r}) \tag{2.6}$$

The total wave function can be constructed as linear combinations of $\phi_i(\mathbf{R})$ and solutions of the electronic SE $\Psi_i(\mathbf{r}, \mathbf{R})$

$$\Psi_{\text{tot}}(\mathbf{r}, \mathbf{R}) = \sum_{i=1}^{\infty} \phi_i(\mathbf{R})\Psi_i(\mathbf{r}, \mathbf{R}) \tag{2.7}$$

This can be inserted in the complete SE and the following equation is obtained:

$$\hat{H}\Psi_{\text{tot}} = \sum_{i=1}^{\infty} (\hat{T}_N + \hat{H}_e)\phi_i(\mathbf{R})\Psi_i(\mathbf{r}, \mathbf{R}) = E_{\text{tot}} \sum_{i=1}^{\infty} \phi_i(\mathbf{R})\Psi_i(\mathbf{r}, \mathbf{R}) \tag{2.8}$$

Since $\hat{H}_e$ only acts on the electronic wave function, using the definition of the kinetic energy operator for the nuclei $\hat{T}_N = \nabla_N^2 = -\sum\limits_{K=1}^{N_{\text{at}}} \nabla_K^2/(2m_K)$ leads to

$$\sum_{i=1}^{\infty} \nabla_N^2 \left[\Psi_i(\mathbf{r},\mathbf{R})\phi_i(\mathbf{R})\right] + \phi_i(\mathbf{R}) \left[\hat{H}_e \Psi_i(\mathbf{r},\mathbf{R})\right] = E_{\text{tot}} \sum_{i=1}^{\infty} \phi_i(\mathbf{R})\Psi_i(\mathbf{r},\mathbf{R}) \tag{2.9}$$

If the product rule is applied twice and using eq. (2.6), the following equation is obtained

$$\begin{aligned}\sum_{i=1}^{\infty} \Psi_i(\mathbf{r},\mathbf{R})\nabla_N^2\phi_i(\mathbf{R}) &+ 2(\nabla_K\Psi_i(\mathbf{r},\mathbf{R}))(\nabla_K\phi_i(\mathbf{R})) \\ &+\phi_i(\mathbf{R})\nabla_N^2\Psi_i(\mathbf{r},\mathbf{R}) + \phi_i(\mathbf{R})E_i(\mathbf{R})\Psi_i(\mathbf{r},\mathbf{R}) \\ &= E_{\text{tot}}\sum_{i=1}^{\infty}\phi_i(\mathbf{R})\Psi_i(\mathbf{r},\mathbf{R})\end{aligned} \tag{2.10}$$

Since $\Psi_i(\mathbf{r},\mathbf{R})$ are orthonormal, multiplying by $\Psi_j(\mathbf{r},\mathbf{R})^\star$ and integrating over the electronic coordinates leads to

$$\nabla_N^2\phi_j(\mathbf{R}) + E_j(\mathbf{R}) + \sum_{j=1}^{\infty} 2\left\langle\Psi_j|\nabla_K|\Psi_i\right\rangle \nabla_K\phi_i(\mathbf{R}) + \left\langle\Psi_i|\nabla_N^2|\Psi_i\right\rangle\phi_i(\mathbf{R}) = E_{\text{tot}}\phi_j(\mathbf{R}) \tag{2.11}$$

Here, the bra-ket notation is used. This includes two non-adiabatic correction terms. In the adiabatic approximation all coupling terms are neglected (only terms with $i = j$ remain). Then, the following equation is obtained:

$$\left(\nabla_N^2 + E_j(\mathbf{R}) + \left\langle\Psi_j|\nabla_N^2|\Psi_j\right\rangle\right)\phi_j(\mathbf{R}) = E_{\text{tot}}\phi_j(\mathbf{R}) \tag{2.12}$$

In the Born–Oppenheimer approximation the term $\langle\Psi_j|\nabla_N^2|\Psi_j\rangle$ is neglected and the electronic energy act as a potential energy

$$(\hat{T}_N + E_j(\mathbf{R}))\phi_j(\mathbf{R}) = E_{\text{tot}}\phi_j(\mathbf{R}) \tag{2.13}$$

and the nuclei move on the potential energy surface (PES), which are solutions of the electronic SE.

## 2.2 The Potential Energy Surface

The Energy expression in eq. 2.6 $E(\mathbf{R})$ is called the potential energy surface (PES) describing the energy of every atom over all possible positions. In Cartesian coordinates $\mathbf{R} \in \mathbb{R}^{3N_{\text{at}}}$ the PES is a $3N_{\text{at}}$-dimensional surface, where $N_{\text{at}}$ is the total number of atoms. Each structure is defined by this vector $\mathbf{R}$, where

$$\mathbf{R} = (X_1, Y_1, Z_1, X_2, Y_2, Z_2, \cdots, X_{N_{\text{at}}}, Y_{N_{\text{at}}}, Z_{N_{\text{at}}}) \tag{2.14}$$

Here, $(X_K, Y_K, Z_K)$ are the Cartesian coordinates of the atom $K$. Using Cartesian coordinates uniquely defines a molecule, i.e. a collection of atoms. However, the origin can be chosen arbitrarily. The dimensionality can be reduced by removing translation and rotation.

The energy $E(\mathbf{R})$ does not change upon a translation $\mathbf{u}$, i.e. $E(\mathbf{R} + \mathbf{u}) = E(\mathbf{R})$. This reduces the degree of freedom by three. Additionally a rotation about the $x$,$y$ or $z$ axes does not change $E(\mathbf{R})$. Let $U$ be a rotation, $U \in \mathrm{O}(3)$, then applying this rotation does not change the energy $E(\mathbf{R})$, i.e. $E(U\mathbf{R}) = E(\mathbf{R})$. Therefore, the total degrees of freedom for a molecule is $3N_{\text{at}} - 6$.

The coordinates of a molecule can be transformed into so-called internal coordinates. How this is done exactly, is presented in chapter 4. Then the PES can be written as $E(\mathbf{Q})$, where $\mathbf{Q}$ is a set of internal coordinates, that are rotational and translational invariant.

The visualization and calculation of a complete PES for most molecules are hard to achieve due to the high-dimensional surface. Therefore, only stationary points of the PES, i.e. points with $\nabla_{\mathbf{R}} E(\mathbf{R}) = 0$, are of special interest. This includes local minima, which correspond to stable geometries of the molecule. Additional saddle points are of interest, especially first-order saddle points because they correspond to a maximum of lowest energy paths connecting two minima. Thus they are relevant for the concept of a transition state, which will be briefly discussed in the next section. A complete calculation of the PES provides all information of all chemical structures and pathways connecting them.

# 3 Standard Optimization Algorithms

In this chapter different algorithms to find stationary points of the potential energy surface (PES) are discussed. Stationary points require a vanishing gradient, i.e. $\nabla f(\mathbf{x}) = \mathbf{0}$. The most interesting points are the minima of the PES, which correspond to stable geometries of molecules. This is often called geometry optimization or minimization, which will be dealt with in section 3.1. The other stationary points of interest are first-order saddle points, which correspond to transition states. Two algorithms are presented in section 3.2, how to find those points.

A function $f : G \subset \mathbb{R}^d \to \mathbb{R}$ is in $\xi = (x_1, \cdots, x_d) \in G$ differentiable. If $f$ has at the point $\xi$ the partial derivatives $\frac{\partial f}{\partial x_i}$, $i = 1, \cdots, d$, then

$$\mathbf{g} = \mathrm{grad}\, f(\xi) = \begin{pmatrix} \frac{\partial f(\xi)}{\partial x_1} \\ \vdots \\ \frac{\partial f(\xi)}{\partial x_d} \end{pmatrix} \tag{3.1}$$

is the gradient of $f$ at the point $\xi$.

The gradient indicates the direction of strongest increase of $f$. If the direction is turned around, the direction of the gradient is the strongest decent of $f$. Taylor's theorem can be applied around a point $\mathbf{x}_0$, if $f$ is two times differentiable:

$$f(\mathbf{x}_0 + \mathbf{h}) = f(\mathbf{x}_0) + f'(\mathbf{x}_0)\mathbf{h} + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 f(\mathbf{x}_0)}{\partial x_i \partial x_j} h_i h_j + \mathcal{O}(\mathbf{h}^3) \tag{3.2}$$

## 3.1 Minimization

In general, an optimizer tries to find $\min\limits_{x \in G} f(x)$, or $\max\limits_{x \in G} f(x)$ of a function $f : \mathbb{R}^d \to \mathbb{R}$. Usually $G$ is a subset of the real vector space, i.e. $G \subset \mathbb{R}^d$.

The function $f$ has a local minimum in $\xi \in G$ respectively a local maximum if there exists a neighborhood $U$ of $\xi$ with:

$$\forall \mathbf{x} \in U \cap G : f(\mathbf{x}) \geq f(\xi) \quad \text{respectively } f(\mathbf{x}) \leq f(\xi) \tag{3.3}$$

Note, that functions don't have to have minima and maxima. Now it's time to introduce a necessary criterion for extrema. If $f$ has a local extremum then grad $f(\xi) = 0$. If grad $f(\xi) = 0$, then $\xi$ is called a critical point of $f$. Note that edge points can be local extrema without being critical points and critical points are not necessarily local extrema. To get a sufficient criterion for local extrema the Hessian is introduced:

$$H(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1^2} & \cdots & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_d \partial x_1} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_2} & \cdots & \\ \vdots & & \ddots & \\ \frac{\partial f(\mathbf{x})}{\partial x_1 \partial x_d} & \cdots & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_d^2} \end{pmatrix} \tag{3.4}$$

If $H(\xi)$ is positive definite respectively negative definite then $\xi$ is a local minimum respectively maximum. If $H$ is indefinite, then $\xi$ is no local extremum, but rather a saddle point. With this definition eq. (3.2) can be rewritten

$$f(\mathbf{x}_0 + \mathbf{h}) = f(\mathbf{x}_0) + \mathbf{g}^\top \mathbf{h} + \frac{1}{2} \mathbf{h}^\top \mathbf{H} \mathbf{h} + \mathcal{O}(\mathbf{h}^3) \tag{3.5}$$

A numerical way to get to extrema is Newton's method. In general, Newton's method tries to find a root of a function $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, where $(f_1, \cdots, f_d)^\top : G \subset \mathbb{R}^d \to \mathbb{R}^d$ and leads to the system of equations

$$\begin{aligned} f_1(x_1, \cdots, x_d) &= 0 \\ &\vdots \\ f_d(x_1, \cdots, x_d) &= 0 \end{aligned} \tag{3.6}$$

The idea is to use the Newtons sequence

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{f}'(\mathbf{x}_n)^{-1}\mathbf{f}(\mathbf{x}) \tag{3.7}$$

If $\mathbf{f}$ is differentiable then $\mathbf{x}_n \to \xi$, and therefore $\mathbf{f}(\mathbf{x}_n) \to \mathbf{f}(\xi)$. Using the gradient $\mathbf{g}$ with the derivative, the Hessian $\mathbf{H}$, the Newton Step is defined as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{H}(\mathbf{x}_n)^{-1}\mathbf{g}(\mathbf{x}_n) \tag{3.8}$$

Since the Hessian is usually expensive the calculate, approximations the the scale of the gradient has to be done. In the following steepest decent, the conjugate gradient method and the limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [28–32] algorithms are shortly explained. They are called quasi-Newton methods, because they try to approximate the inverse Hessian.

### 3.1.1 Steepest Decent

Steepest decent is attributed to Augustin-Louis Cauchy in the year 1847 [33]. The iterative steepest decent or gradient descent calculates its step as

$$\mathbf{x}_{n+1} - \mathbf{x}_n = -\alpha_n \nabla f(\mathbf{x}_n) \tag{3.9}$$

where $\alpha_n$ is a scaling of the step length and $\nabla f(\mathbf{x}_n)$ is the gradient. The negative gradient direction is, where the function decreases fastest. The scaling $\alpha_n$ is, for example, obtained by line search, where the function $h(\alpha_n) := f(\mathbf{x}_n - \alpha_n \nabla f(\mathbf{x}_n))$ is minimized.

### 3.1.2 Conjugate Gradient

The conjugate gradient [34] improves the method mentioned above. It starts with a steepest decent step:

$$\mathbf{s}_0 = \Delta\mathbf{x}_0 = -\nabla f(\mathbf{x}_0) \tag{3.10}$$

After the first iteration the following steps for one iteration is performed where $\mathbf{s}_n$ is the conjugate direction:

1. Calculate the steepest decent direction: $\Delta\mathbf{x}_n = -\nabla f(\mathbf{x}_n)$

2. Compute $\gamma_n$, according to one of the following formulas.

- FletcherReeves [35]:
$$\gamma_n^{FR} = \frac{\Delta\mathbf{x}_n^\top \Delta\mathbf{x}_n}{\Delta\mathbf{x}_{n-1}^\top \Delta\mathbf{x}_{n-1}} \tag{3.11}$$
- PolakRibière: [36]
$$\gamma_n^{PR} = \frac{\Delta\mathbf{x}_n^\top (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})}{\Delta\mathbf{x}_{n-1}^\top \Delta\mathbf{x}_{n-1}} \tag{3.12}$$
- Hestenes-Stiefel [34]
$$\gamma_n^{HS} = \frac{\Delta\mathbf{x}_n^\top (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})}{-\mathbf{s}_{n-1}^\top (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})} \tag{3.13}$$
- DaiYuan [37]
$$\gamma_n^{DY} = \frac{\Delta\mathbf{x}_n^\top \Delta\mathbf{x}_n}{-\mathbf{s}_{n-1}^\top (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})} \tag{3.14}$$

3. Calculate the conjugate direction: $\mathbf{s}_n = \Delta\mathbf{x}_n + \gamma_n \mathbf{s}_{n-1}$
4. Perform a line search to get $\beta_n$
5. Calculate new coordinates $\mathbf{x}_{n+1} = \mathbf{x}_n + \beta_n \mathbf{s}_n$

A line search tries to find a satisfactory step along the direction $\mathbf{s}_n$. This is done by minimizing $h(\beta_n) = f(\mathbf{x}_n + \beta_n \mathbf{s}_n)$ with respect to $\beta_n$, i.e. by solving $h'(\beta_n) = 0$.

### 3.1.3 L-BFGS

To get a better quasi-Newton method, the L-BFGS algorithm was developed. It tries to solve the linear system of equation
$$\mathbf{B}_n \mathbf{s}_n = -\nabla f(\mathbf{x}_n) \tag{3.15}$$
But instead of using the Hessian $\mathbf{H}$, it is updated at each iteration step. The update of the Hessian is chosen to grantee that the Hessian remains a positive definite. The update equation from $\mathbf{B}_n$ to $\mathbf{B}_{n+1}$ is
$$\mathbf{B}_{n+1} = \mathbf{B}_n + \frac{\mathbf{g}_n \mathbf{g}_n^\top}{\mathbf{g}_n^\top \mathbf{s}_n} - \frac{\mathbf{B}_n \mathbf{s}_n \mathbf{s}_n^\top \mathbf{B}_n^\top}{\mathbf{s}_n^\top \mathbf{B}_n \mathbf{s}_n} \tag{3.16}$$
where $\mathbf{g}_n = \nabla f(\mathbf{x}_{n+1}) - \nabla f(\mathbf{x}_n)$ and $\mathbf{s}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$ Starting with the initial guess $\mathbf{x}_0$ and an approximate $\mathbf{B}_0$, which is usually the identity matrix, the algorithm is as follows:

1. obtain $\mathbf{s}_n$ from eq 3.15.

2. Optionally a line search to find a good step size can be performed
3. Get a new gradient $\nabla f(\mathbf{x}_{n+1})$.
4. Update the Hessian as in eq 3.16

This is repeated until the convergence criteria are full filled. Instead of solving the eq. 3.15, the inverse can be updated. This is done by

$$\mathbf{B}_{n+1}^{-1} = \mathbf{B}_n^{-1} + \frac{\left(\mathbf{s}_n^\top \mathbf{g}_n + \mathbf{g}_n^\top \mathbf{B}_n^{-1} \mathbf{g}_n\right)\left(\mathbf{s}_n \mathbf{s}_n^\top\right)}{\left(\mathbf{s}_n^\top \mathbf{g}_n\right)^2} - \frac{\mathbf{B}_n^\top \mathbf{g}_n \mathbf{s}_n^\top + \mathbf{s}_n \mathbf{g}_n^\top \mathbf{B}_n^{-1}}{\mathbf{s}_n \mathbf{g_n}} \tag{3.17}$$

Since the update provides a positive definite approximation of the Hessian, this can lead to problems if a point is to far away from the minimum.

The extension limited memory BFGS (L-BFGS) [32] stores $m$ number of Broyden–Fletcher–Goldfarb–Shanno (BFGS) corrections. In the first $m$ iterations it is identical to the regular BFGS. If $n > m$ the inverse $\mathbf{B}_n^{-1}$ is obtained via m updates of $\mathbf{B}_0$, where the information of the previous $m$ updates are used. The pairs $\{\mathbf{g}_i, \mathbf{s}_i\}_{i=n-l}^{n}$ are stored, with $l = \min(n, m-1)$. Note, that $\mathbf{B}_n^{-1}$ is not formed explicitly, but with previous, up to $m$, values of $\mathbf{g}_i$ and $\mathbf{s}_i$:

$$\begin{aligned}
\mathbf{B}_{n+1}^{-1} = &\left(\mathbf{V}_n^\top \cdots \mathbf{V}_{n-l}^\top\right) \mathbf{B}_0^{-1} \left(\mathbf{V}_{k-l} \cdots \mathbf{V}_n\right) \\
&+ \rho_{n-l} \left(\mathbf{V}_n^\top \cdots \mathbf{V}_{n-l+1}^\top\right) \mathbf{s}_{n-l} \mathbf{s}_{n-l}^\top \left(\mathbf{V}_{k-l+1} \cdots \mathbf{V}_n\right) \\
&+ \rho_{n-l+1} \left(\mathbf{V}_n^\top \cdots \mathbf{V}_{n-l+2}^\top\right) \mathbf{s}_{n-l+1} \mathbf{s}_{n-l+1}^\top \left(\mathbf{V}_{n-l+2} \cdots \mathbf{V}_n\right) \\
&\vdots \\
&+ \rho_n \mathbf{s}_n \mathbf{s}_n^\top
\end{aligned} \tag{3.18}$$

where $\rho = \frac{1}{\mathbf{g}_n^\top \mathbf{s_n}}$ and $\mathbf{V}_n = \mathbf{I} - \rho_n \mathbf{g}_n \mathbf{s}_n^\top$.

## 3.2 Saddle Point Search

Stationary points, which are more complicated to find, are saddle points of $i$th-order, $0 < i \leq d$. They are defined via the eigenvalues of their Hessian: A saddle point of $i$th-order has $i$ negative eigenvalues and $d - i$ positive eigenvalues. Usually, the search for saddle points requires the full Hessian to calculate, but for $i = 1$ an approximation can be used, which only requires the calculation of the gradient. This method is called the dimer method and is discussed in the next section, section 3.2.1. An algorithm to find a first-order saddle point with full Hessian available is the partitioned rational function optimization (P-RFO) algorithm and is discussed in section section 3.2.2.

### 3.2.1 Dimer Method

The dimer method was originally introduced by Henkelmann and Jónsson [38]. A dimer consists of two close lying endpoints $\mathbf{x}_1$ and $\mathbf{x}_2$, which have a common midpoint $\mathbf{R}$. The distance between both endpoints is $2R$. The endpoints can be defined by

$$\mathbf{x}_1 = \mathbf{R} + R\mathbf{N} \quad \text{and} \quad \mathbf{x}_2 = \mathbf{R} - R\mathbf{N} \tag{3.19}$$

Here, $\mathbf{N}$ is a unit vector in the direction of the dimer, i.e.

$$\mathbf{N} = \frac{\mathbf{x}_1 - \mathbf{R}}{R} = \frac{\mathbf{R} - \mathbf{x}_2}{R} \tag{3.20}$$

with $|\mathbf{N}| = 1$. An schematic representation of a dimer can be seen in fig. 3.1 The saddle point search via the dimer method involves two steps, first a rotation in the direction of the eigenvector corresponding to the lowest eigenvalue, i.e. in the direction of lowest curvature and a translation of the mid point towards a saddle point. In the first step the dimer is rotated around the midpoint. The curvature along the dimer axes can be calculated by the exact Hessian, or if not known or to expensive, via a finite difference approach

$$C_N = \mathbf{N}^\top \mathbf{H} \mathbf{N} \approx -\frac{(\mathbf{g}_1 - \mathbf{g}_2) \cdot \mathbf{N}}{2R} \approx \frac{E_1 + E_2 - 2E_0}{R^2} \tag{3.21}$$

where $\mathbf{g}_1$ is the gradient at $\mathbf{x}_1$ and $\mathbf{g}_2$ at $\mathbf{x}_2$. The total dimer energy is $E = E_1 + E_2$. The plane in which the rotation takes place is spanned by $\mathbf{N}$ and a unit vector $\Theta$, which has to be determined, that is orthogonal to the dimer axis $\mathbf{N}$, i.e. $\mathbf{N} \cdot \Theta = 0$. The optimal direction is the maximum overlap between $\Theta$ and the eigenvector corresponding to the lowest eigenvalue.

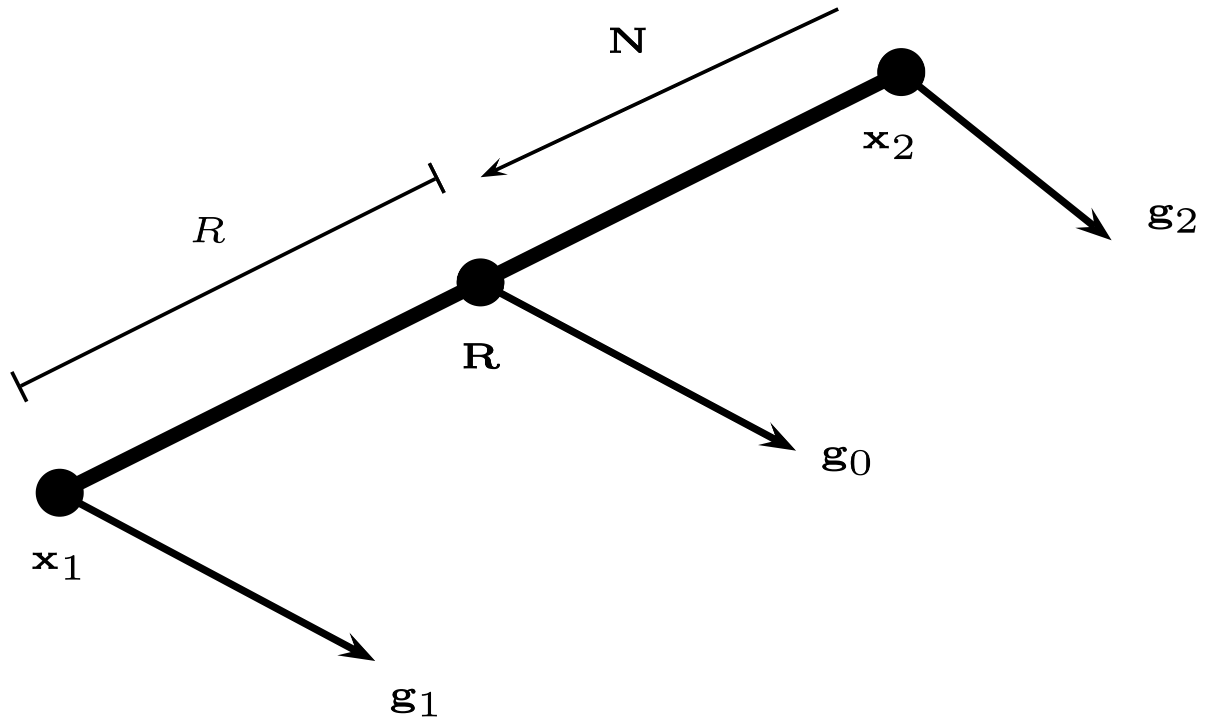


**Figure 3.1:** Dimer with endpoints $\mathbf{x}_i$ and the mid point $\mathbf{R}$. The distance between one endpoint and the midpoint is $R$, and $\mathbf{N}$ is a unit vector in the direction of the dimer. The gradients are labeled by $\mathbf{g}_i$.

This is done by minimizing

$$\mathbf{F}_{\mathrm{R}} = -(\mathbf{g}_1 - \mathbf{g}_2) + ((\mathbf{g}_1 - \mathbf{g}_2) \cdot \mathbf{N})\,\mathbf{N} \tag{3.22}$$

then the dimer aligns along the eigenmode corresponding to the lowest eigenvalue of the Hessian. Kästner et al [39] uses the gradient at the dimer midpoint and at one endpoint, rather the gradients for both endpoints, i.e.

$$\mathbf{F}_{\mathrm{R}} = -2\,(\mathbf{g}_1 - \mathbf{g}_0) + 2\,((\mathbf{g}_1 - \mathbf{g}_0) \cdot \mathbf{N})\,\mathbf{N} \tag{3.23}$$

Then the curvature becomes

$$C_N = \frac{(\mathbf{g_1} - \mathbf{g}_0) \cdot N}{R} \tag{3.24}$$

The optimal rotation angle is obtained by minimizing the curvature, i.e minimizing $C_N$. The angle can be estimated without gradient evaluation by [40]

$$\phi_1 = -\frac{1}{2} \arctan \frac{\frac{\partial C_N}{\partial \phi}}{2|C_N|} \tag{3.25}$$

Finally the optimal rotation angle is [40]

$$\phi_{\mathrm{min}} = \frac{1}{2} \arctan \frac{b_1}{a_1} \tag{3.26}$$

with

$$\begin{aligned} b_1 &= \frac{1}{2} \frac{\partial C_N}{\partial \phi}\bigg|_{\phi=0} \\ a_1 &= \frac{C_N|_{\phi=0} - C_n|_{\phi=\phi_1} + b_1 \sin(2\phi_1)}{1 - \cos(2\phi_1)} \end{aligned} \tag{3.27}$$

which are obtained by assuming a local quadratic PES [40].

After the direction is align with the minimal mode, a translation is done and the force

$$\mathbf{F}_{\mathrm{t}} = -\mathbf{g}_0 + 2(\mathbf{g}_0 \cdot \mathbf{N})\mathbf{N} \tag{3.28}$$

is followed. $\mathrm{g}_0$ is the gradient of the dimer midpoint. Rotation and translation are repeated until the convergence criteria are fulfilled. The main advantage here is, that no analytic hessian is required. This makes the dimer method applicable to large dimensional systems.

### 3.2.2 P-RFO

The P-RFO algorithm was introduced by Cerjan and Miller [41] and was improved Baker [42]. The Newton step from eq 3.8 is rewritten in terms of the eigenvectors $\mathbf{v}_i$ and there corresponding eigenvalues $\lambda_i$

$$\Delta\mathbf{x} = -\sum_{i=1}^{d} \frac{\mathbf{v}_i^\top \mathbf{g} \mathbf{v}_i}{\lambda_i} \tag{3.29}$$

Banerjee et. al [43] showed that this step is directed in the opposite direction of the gradient along eigenvectors with positive eigenvalue and along the gradient with negative eigenvalue. Being on the right position of the potential energy surface, meaning one negative eigenvalue and all others positive, this is a good step. However the position on the PES does not have this required eigenvalue structure. Cerjan and Miller [41] used a simple modification to search for stationary points with required eigenvalue structure. They introduced a shift parameter $\beta$ on the $i$-th eigenvalue

$$\Delta\mathbf{x} = -\sum_{i=1}^{d} \frac{\mathbf{v}_i^\top \mathbf{g} \mathbf{v}_i}{\lambda_i - \beta} \tag{3.30}$$

Based on Banerjee et. al [43] a rotational approximation is used

$$f(\mathbf{x}+\mathbf{h}) - f(\mathbf{x}) \approx \frac{\mathbf{g}^\top \mathbf{h} + \frac{1}{2}\mathbf{h}^\top \mathbf{H}\mathbf{h}}{1 + \mathbf{h}^\top \mathbf{S}\mathbf{h}} \tag{3.31}$$

The matrix $\mathbf{S} \in \mathbb{R}^{d\times d}$ is symmetric and is used for scaling. If the condition $\nabla_{\mathbf{h}} f = 0$ is used, an eigenvalue equation is obtained [42]

$$\begin{pmatrix} \mathbf{H} & \mathbf{g} \\ \mathbf{g}^\top & 0 \end{pmatrix} \begin{pmatrix} \mathbf{h} \\ 1 \end{pmatrix} = \beta \begin{pmatrix} \mathbf{S} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{h} \\ 1 \end{pmatrix} \tag{3.32}$$

This is an equation of $d+1$ dimension and can be expanded into [42]

$$\begin{aligned} (\mathbf{H} - \beta\mathbf{S})\,\mathbf{h} + \mathbf{g} &= 0 \\ \mathbf{g}^\top \mathbf{h} &= \beta \end{aligned} \tag{3.33}$$

$\beta$ can then be calculated from the following equation

$$\sum_{i=1}^{d} \frac{\left(\mathbf{v}_i^\top \mathbf{g}\right)^2}{\beta - \lambda_i} = \beta \tag{3.34}$$

Separating the Rational functional optimization matrix can be split in two parts, one for the eigenvector to be maximized and $d-1$ to be minimized leads to two smaller optimization matrices and can be treated separately [40]. To do so two separate shift parameter $\beta_p$ and $\beta_d$ are used. The step is therefore

$$\Delta\mathbf{x} = \sum_{i=1}^{d} \Delta\mathbf{x}_i, \quad \Delta\mathbf{x}_k = \frac{\mathbf{v}_k^\top \mathbf{g} \mathbf{v}_k}{\lambda_k - \beta_p}, \quad \Delta\mathbf{x}_{i,i\neq k} = \frac{\mathbf{v}_i^\top \mathbf{g} \mathbf{v}_i}{\lambda_i - \beta_d} \tag{3.35}$$

where eigenmode $k$ corresponds to the lowest eigenvalue.

# 4 Internal Coordinates

In this chapter, the description of internal coordinates is introduced. Cartesian coordinates are usually highly coupled. This means, that for example rotating a $CH_3$-group by a certain angle requires all four atoms, and therefore 9 Cartesian coordinates move, instead of one single coordinate. Another aspect is the translation and rotational invariance of molecules. Therefore, not the full set of $3N_{\text{at}}$ (Cartesian) coordinates is required. Here, $N_{\text{at}}$ is the number of atoms. In the following, internal coordinates to reduce the coupling between coordinates and reduction to $3N_{\text{at}} - 6$ coordinates are explained. For linear molecules, the internal coordinates consist of $3N_{\text{at}} - 5$ coordinates. However, in this chapter, non-linear molecules are considered.

In section 4.1 how a so-called connection scheme is set up to build the primitive internal coordinates. These primitives consist of bond length, bond angles, and dihedrals. An alternative set is the total connection scheme, where every atom is connected to every other one.

Next, in section 4.2, the relation between Cartesian coordinates and internal coordinates is discussed. This is followed by the delocalization of the primitives in section 4.3 to get the $3N_{\text{at}} - 6$ degrees of freedom. Then in section 4.4 the iterative back transformation is introduced, followed by a discussion about its failure. Finally, in section 4.5 a general scheme, how geometry optimization in internal coordinates is performed, is presented.

All the energies, forces (gradients), coordinates, etc. are in atomic units.

## 4.1 Primitve Internal Coordinates

First, the so-called primitive internal coordinates have to be constructed [6, 44]. But before they can be constructed a connection scheme has to be calculated. Therefore from each atom, the distance is calculated to each other atom and if the distance to a certain atom is below a threshold, they are considered connected [9]. The criteria for the connection is if the square of the inter-atomic distance is less than 1.25 times the square of the sum of the corresponding covalent atomic radii. The covalent radius describes the size of an atom taking part in a bond. In general, the sum of two covalent radii should be the bond length.

The covalent radius of atom A is $r_{\mathrm{A}}$ and the for atom B $r_{\mathrm{B}}$, then the sum of both covalent radii is $r_{\mathrm{AB}} = r_{\mathrm{A}} + r_{\mathrm{B}}$. the squared distance between A and B is $s_{\mathrm{AB}} = (x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2$. Then a connection is made if

$$s_{\mathrm{AB}} \leq 1.25 \times r_{\mathrm{AB}}^2 \tag{4.1}$$

The covalent radius of an atom depends on its type. The following covalent radii are used in DL-FIND [9, 45]

- $i \leq 2$: $r_{\mathrm{AB}} = 1$ (for atoms H and He)
- $i \leq 10$: $r_{\mathrm{AB}} = 1.6$ (for Li,$\cdots$, Ne)
- $i \leq 18$: $r_{\mathrm{AB}} = 2$ (for Na, $\cdots$, Ar)
- $i \leq 36$: $r_{\mathrm{AB}} = 3$ (for K, $\cdots$, Kr)
- $i \leq 54$: $r_{\mathrm{AB}} = 4$ (for Rb, $\cdots$, Xe)
- $i > 54$ : $r_{\mathrm{AB}} = 5$ ( Cs, $\cdots$ )

where $i$ is the nucleus charge, i.e. the atomic number.

Sometimes this procedure can lead to isolated fragments. To counter this problem, missing connections can be inserted by the following. Starting with the first atom, a path connecting the nearest neighbor atoms is generated. If a connection of this path is not yet in the connectivity list, it is added [9]. Once this is done, the actual primitive internal coordinates can be calculated. They consist of bond length, bond angles, and dihedrals. They can be seen in fig. 4.1.

The bond length is the distance between two connected atoms with coordinates $\mathbf{x}_n$ and $\mathbf{x}_m$ and is calculated as:

$$q_r = |\mathbf{x}_n - \mathbf{x}_m| = \sqrt{\sum_{i=1}^{3} (x_{n,i} - x_{m,i})^2} \tag{4.2}$$

The bond length is symmetric and therefore the bond length between atom 1 and atom 2 is the same as the bond length of atom 2 with atom 1.

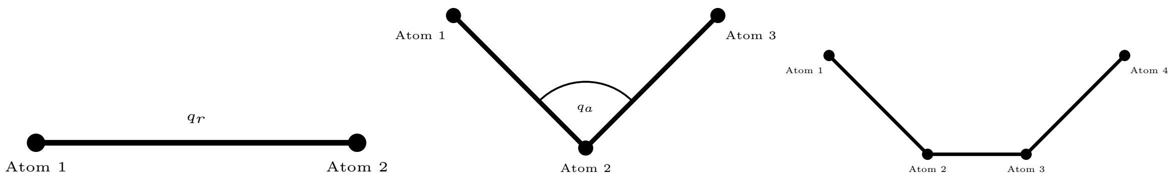


**Figure 4.1:** Primitive internal coordinates, on the left bond lengths, in the middle bond angles, and on the right dihedrals.

Between three connected atoms with coordinates $\mathbf{x}_n$, $\mathbf{x}_m$ and $\mathbf{x}_l$ a bond angle around atom $\mathbf{x}_m$ can be calculated

$$q_a = \arccos \frac{(\mathbf{x_n} - \mathbf{x}_m)(\mathbf{x}_l - \mathbf{x}_m)}{|\mathbf{x}_n - \mathbf{x}_m| \, |\mathbf{x}_l - \mathbf{x}_m|} \tag{4.3}$$

If the bonding angle is almost linear ($> 175°$ or $< 5°$) it drops out. There are two cases, where this angle is replaced. If the central atom of this angle is connected to two atoms, the bonding angle is replaced by two Cartesian coordinates of the central atom, which are the most orthogonal to the line of the other two atoms. The other case is if the central atom is connected to more than two atoms and is the center of a planar system. Then, one of these angles is replaced by an improper dihedral, which means the remaining atom of the other angle is taken as the final atom. Improper dihedrals are defined by a central atom 4, which is connected to atoms 1, 2, and 3. However, the four atoms are not connected in a row. Consider that atoms 1,4,3 are connected, as well as atoms 1,4,2 and atoms 2,4,3, i.e atom 2 is not connected to atom 3 or atom 1. They would rise to three angles. If this system is planar, one of these angles is dropped out, e.g. the angle between atoms 1,4,2. Then this angle is replaced by the dihedral, see below, formed for example by atoms 1,4,3,2.

If four atoms are connected the dihedral can be calculated. It is the angle between the plane spanned by atoms 1,2,3 and the plane spanned by atoms 2,3,4. The dihedral is then calculated as

$$q_d = \arccos \frac{(\mathbf{u} \times \mathbf{w})(\mathbf{v} \times \mathbf{w})}{|\mathbf{u} \times \mathbf{w}| \, |\mathbf{v} \times \mathbf{w}|} \tag{4.4}$$

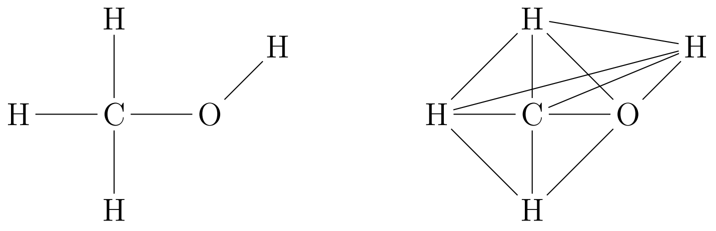


**Figure 4.2:** Methanol represented as normal molecule via the connectivity calculation on the left and in the total connection scheme on the right.

with the following vectors

$$\mathbf{u} = \frac{\mathbf{x}_n - \mathbf{x}_m}{|\mathbf{x}_n - \mathbf{x}_m|} \tag{4.5}$$

$$\mathbf{v} = \frac{\mathbf{x}_k - \mathbf{x}_l}{|\mathbf{x}_k - \mathbf{x}_l|} \tag{4.6}$$

$$\mathbf{w} = \frac{\mathbf{x}_l - \mathbf{x}_m}{|\mathbf{x}_l - \mathbf{x}_m|} \tag{4.7}$$

where $\mathbf{x}_n$ are the coordinates of atom 1, $\mathbf{x}_m$ of atom 2, $\mathbf{x}_l$ of atom 3 and $\mathbf{x}_k$ of atom 4. All calculated primitive internal coordinates calculated from bond lengths, bond angles, and dihedrals form the coordinates $\mathbf{q} \in \mathbb{R}^M$, with $M$ being the number of primitive internal coordinates.

An alternative way to construct primitive internal coordinates is the total connection scheme. Thereby, all atoms are connected and only the bond length from eq. (4.2) is used to calculate the primitives. An example can be seen in fig. 4.2, where on the left is the connection scheme for length, angles, and dihedral and on the right for the total connection scheme. This will in general generate more internal coordinates than the other set of primitive internal coordinates. If primitive internal coordinates are mentioned in the following, they refer to both, the connectivity list with bond length, bond angles, and dihedrals, as well as the total connection scheme.

## 4.2 Relation to Cartesian Coordinates

Small displacements of the internal coordinates $\Delta\mathbf{q}$ can be related to small displacements of Cartesian coordinates $\Delta\mathbf{x}$ via the so called $\mathbf{B}$-matrix [46]

$$\Delta\mathbf{q} = \mathbf{B}\Delta\mathbf{x} \tag{4.8}$$

Therefore, the $\mathbf{B}$-matrix is defined as (if the displacements are infinitesimal):

$$B_{ij} = \frac{\partial q_i}{\partial x_j} \tag{4.9}$$

The $\mathbf{B}$-matrix contains the partial derivative of all primitive internal coordinates with respect to the Cartesian coordinates. If $M$ is the number of primitive internal coordinates, then $\mathbf{B} \in \mathrm{R}^{M\times 3N}$. Usually, $M$ is much larger than $3N$.

Because internal coordinates are applied to optimization problems and transition state search, the gradient and the Hessian have to be transformed accordingly. Applying the chain rule the Cartesian gradient is related to the internal gradient by

$$\nabla_{\mathbf{x}}E(\mathbf{x}) = \mathbf{B}^{\top}\nabla_{\mathbf{q}}E(\mathbf{q}(\mathbf{x})) \tag{4.10}$$

and the Hessian

$$\mathbf{H}_{\mathbf{x}} = \mathbf{B}^{\top}\mathbf{H}_{\mathbf{q}}\mathbf{B} + \mathbf{C} \tag{4.11}$$

with

$$C_{kl} = \sum_{i=1}^{M}(\nabla_{\mathbf{q}}E(\mathbf{q}))_i\frac{\partial^2 q_i}{\partial x_k\partial x_l} \tag{4.12}$$

The transformation of the Hessian requires therefore the second derivatives of the primitive internal coordinates with respect to the Cartesian coordinates.

To transform the Cartesian gradient to primitive internal coordinates, the $\mathbf{B}$-matrix has to be inverted. However, this matrix is usually rectangular since, in general, $M \neq 3N_{\text{at}}$ the number of primitive internal coordinates is larger than the number of Cartesian coordinates. Hence, the generalized inverse of a rectangular matrix has to be used.

Generalized inverses are classified and defined upon following conditions [47]:

- $AA^gA = A$
- $A^gAA^g = A^g$

- $(AA^g)^\top = AA^g$
- $(A^gA)^\top = A^gA$

Here, $A \in \mathbb{R}^{n\times m}$ is a general rectangular matrix. If $A^g \in \mathbb{R}^{m\times n}$ fulfills the first condition it is a generalized inverse of $A$, if the first two are fulfilled, then it is called a reflexive generalized inverse. If all four conditions are fulfilled, it is called a pseudo-inverse and is denoted as $A^+$ and is known as Moore-Pensrose-inverse [48, 49]. Note, when $A$ is a regular matrix, i.e. a square matrix, that has an inverse, then $A^g = A^{-1}$ and is uniquely defined. A computationally efficient way to calculate the pseudo-inverse is the singular value decomposition (singular value decomposition (SVD)) of $A$: $A = U\Sigma V^\top$, then the pseudo inverse is $A^+ = V\Sigma^+U^\top$. The pseudo-inverse of the rectangular diagonal matrix $\Sigma$ is obtained by taking the reciprocal value of the non-zero elements on the diagonal, while the other entries remain zero. Then the matrix is transposed to obtain $\Sigma^+$.

Using this pseudo inverse $\mathbf{B}^+$ the gradient and Hessian can be transformed as the following:

$$\nabla_{\mathbf{q}} E(\mathbf{q}(\mathbf{x})) = \left(\mathbf{B}^\top\right)^+ \nabla_{\mathbf{x}} E(\mathbf{x}) \tag{4.13}$$

$$\mathbf{H_q} = \left(\mathbf{B}^\top\right)^+ \left(\mathbf{H_x} - \mathbf{C}\right) \mathbf{B}^+ \tag{4.14}$$

## 4.3 Delocalized Internal Coordinates

Up to now, the primitive internal coordinates are redundant. They have to be reduced to $3N-6$ coordinates. In order to do so the $\mathbf{G}$-matrix [46] can be calculated

$$\mathbf{G} = \mathbf{B}\mathbf{B}^\top \in \mathbb{R}^{M\times M} \tag{4.15}$$

The diagonalization of $\mathbf{G}$ leads to $m = 3N_{\text{at}} - 6$ eigenvectors with positive eigenvalues and $M - m$ eigenvectors with eigenvalues equal to zero. The non-zero eigenvectors can be written in the matrix $\mathbf{U} \in \mathbb{R}^{M\times m}$. This matrix can be used to calculate $3N_{\text{at}} - 6$ non-redundant coordinates

$$\mathbf{Q} = \mathbf{U}^\top \mathbf{q} \tag{4.16}$$

as well as transforming the $\mathbf{B}$-matrix in delocalized internal coordinates [9, 50]

$$\tilde{\mathbf{B}} = \mathbf{U}^\top \mathbf{B} \in \mathbb{R}^{m\times 3N_{\text{at}}} \tag{4.17}$$

With the matrix $\tilde{\mathbf{B}}$, the delocalized internal coordinates (DLC) can be defined

$$\mathbf{Q} = \tilde{\mathbf{B}}\mathbf{x} \tag{4.18}$$

This matrix is rectangular and therefore no inverse can be calculated. However, this is required to transform the Cartesian gradient into a delocalized non-redundant gradient. The definition of the pseudo inverse of the previous section can be used analogously. Here, $\tilde{\mathbf{B}}^\top \in \mathbb{R}^{3N_{\text{at}}\times m}$ has linear independent columns. Therefore $\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top$ is a regular matrix and thus invertible. Then, the pseudo inverse can be calculated by using the inverse of $\tilde{\mathbf{G}} = \tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top$ (which is a symmetric matrix).

$$\left[\tilde{\mathbf{B}}^\top\right]^+ = \left[\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top\right]^{-1} \tilde{\mathbf{B}} = \tilde{\mathbf{G}}^{-1}\tilde{\mathbf{B}} \tag{4.19}$$

This pseudo inverse is a left inverse, since

$$\left[\tilde{\mathbf{B}}^\top\right]^+ \tilde{\mathbf{B}}^\top = \left[\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top\right]^{-1} \tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top = \tilde{\mathbf{G}}^{-1}\tilde{\mathbf{G}} = \mathbf{I} \quad \in \mathbb{R}^{m\times m} \tag{4.20}$$

Using this pseudo inverse, the gradient can be transformed

$$\nabla_{\mathbf{Q}} E(\mathbf{Q}) = \left[\tilde{\mathbf{B}}^\top\right]^+ \nabla_{\mathbf{x}} E(\mathbf{x}) = \tilde{\mathbf{G}}^{-1}\tilde{\mathbf{B}}\nabla_{\mathbf{x}} E(\mathbf{x}) \tag{4.21}$$

and analogously the Hessian

$$\mathbf{H}_{\mathbf{Q}} = \left[\tilde{\mathbf{B}}^{\top}\right]^{+} (\mathbf{H}_{\mathbf{x}} - \mathbf{C}) \left\{\left[\tilde{\mathbf{B}}^{\top}\right]^{+}\right\}^{\top} = \tilde{\mathbf{G}}^{-1}\tilde{\mathbf{B}}\,(\mathbf{H}_{\mathbf{x}} - \mathbf{C})\,\tilde{\mathbf{B}}^{\top}\tilde{\mathbf{G}}^{-1} \tag{4.22}$$

## 4.4 Iterative Back Transformation

Since the transformation from Cartesian coordinates to internal coordinates is non-linear, there is, in general, no unique back transformation after an optimization step. The finite displacement as in eq. (4.8) is used for the displacements $\Delta\mathbf{Q}$ of the delocalized internal coordinates (DLC) and the Cartesian displacements $\Delta\mathbf{X}$:

$$\Delta\mathbf{Q} = \tilde{\mathbf{B}}\Delta\mathbf{X} \tag{4.23}$$

This can be used as an iterative back transformation [6, 9]

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \tilde{\mathbf{B}}^{+}\left(\mathbf{Q} - \mathbf{Q}_k\right), \tag{4.24}$$

where $\mathbf{Q}$ is the new geometry in internal coordinates after an optimization step. $\mathbf{x}_k$ is started ($k = 1$) with the old Cartesian coordinates.$\mathbf{Q}_k$ are the internal coordinates corresponding to the Cartesian coordinates $\mathbf{X}_k$. To calculate the inverse of $\tilde{\mathbf{B}}$ the generalized inverse of the $\mathbf{G}$-matrix can be used.

$$\tilde{\mathbf{B}}^{+} = \left(\tilde{\mathbf{G}}^{-1}\tilde{\mathbf{B}}\right)^{\top} = \tilde{\mathbf{B}}^{\top}\tilde{\mathbf{G}}^{-1} \tag{4.25}$$

While the matrix $\mathbf{U}^{\top}$ is kept throughout this process, the matrix $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{G}}$ are recalculated for every new Cartesian $\mathbf{x}_k$ in the iterative procedure from eq. (4.24). Throughout this procedure, the Cartesian coordinates are updated until the difference between $\mathbf{Q}$ and $\mathbf{Q}_k$ is small enough.

The primitives are not generated in every step and $\mathbf{G}$ is not diagonalized in every optimization step, because this would be too expensive [9]. Therefore $\tilde{\mathbf{G}}$ is only in the beginning diagonal, but it is reported that the performance does not significantly suffer as long as $\mathbf{G}$ does not become singular [9]. This means that two different coordinate breakdowns can occur. The $\mathbf{G}$ can become linear dependent or the primitives can lie out of their allowed range. In the DLC code, a linear dependence is recognized by a poor convergence of the back transformation [9, 45]. Only a dihedral flip of $360°$ is checked for the movement out of the allowed range for the primitives. If a back transformation fails, the step is rejected and the optimizer is restarted by calculating a new set of internal coordinates.

For each failure of the back transformation, a cyclic failure gauge is incremented by $1000$, which is initially set to zero. If the next back transformations are successful, every time it is divided by two [9]. The program is terminated with an error "Residue conversion error" [45], if the cyclic failure gauge is equal to or above $2000$.

## 4.5 Geometry Optimization in Internal Coordinates – General Scheme

In general geometry optimization (either minimization or saddle point search) in internal coordinates follows the scheme, which can be seen in fig. 4.3. Starting with the initial geometry the connectivity is constructed and the primitive internal coordinates are calculated. This includes the calculation of bond lengths, bond angles, and dihedrals. If the total connection scheme is chosen, no connectivity is created. However, all distances are used as primitive internal coordinates. In the next step, the delocalization of the primitives is calculated to generate the non-redundant set of coordinates. Then the Ab-initio Cartesian gradient is calculated. If the gradient is already below the convergence criteria, the optimization is converged. This is generally not true, so the coordinates and gradients are transformed into non-redundant internal coordinates. Then the optimization step (either for minimization or saddle point search) is performed. The new geometry is iteratively transformed back to new Cartesian coordinates. With this new geometry, a new gradient is calculated and the whole process is repeated until the convergence criteria are fulfilled.

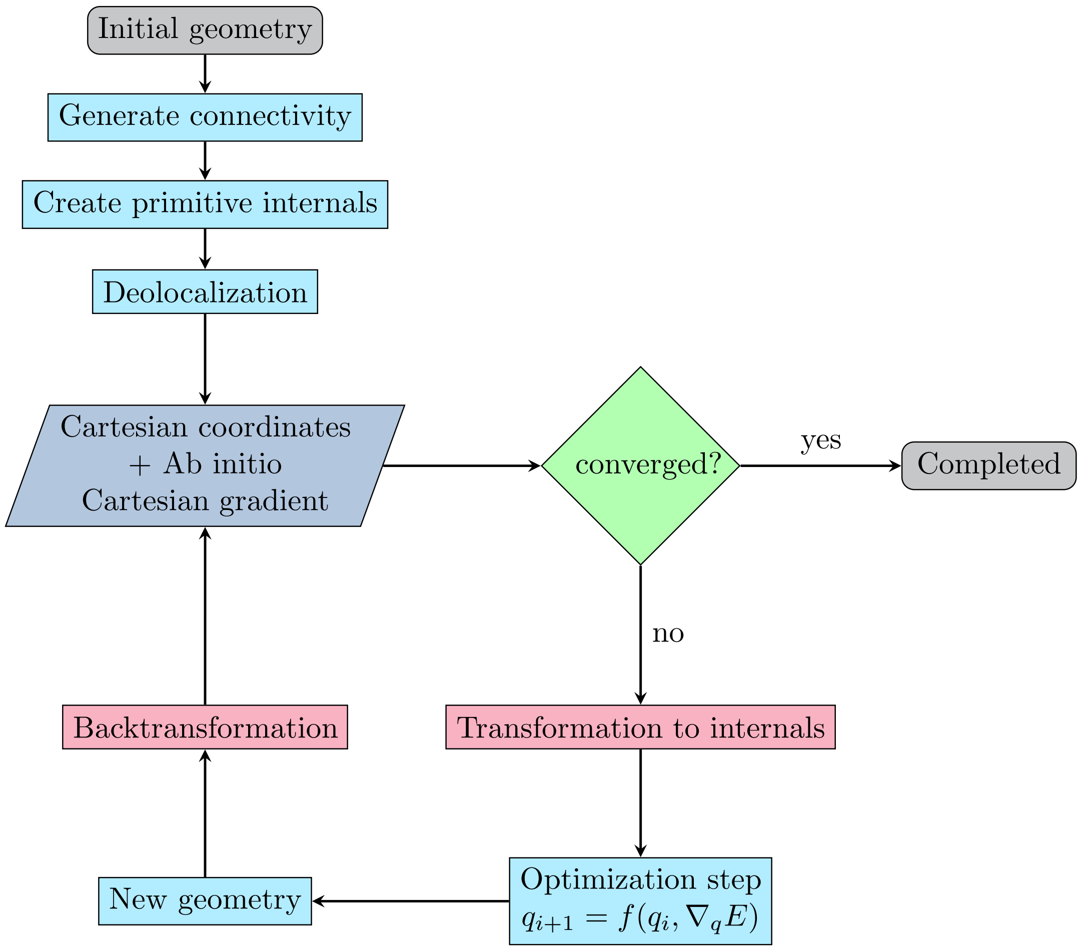

Initial geometry
Generate connectivity
Create primitive internals
Deolocalization
Cartesian coordinates + Ab initio Cartesian gradient
converged?
yes
Completed
no
Transformation to internals
Optimization step $q_{i+1} = f(q_i, \nabla_q E)$
New geometry
Backtransformation

**Figure 4.3:** Schematic representation for a general geometry optimization in delocalized internal coordinates.

# 5 Gaussian Process Regression

Gaussian process regression (GPR) is a widely used machine learning tool [10]. In general, Gaussian process (GP)s are stochastic processes. Stochastic processes are a collection of random variables, which are indexed by a mathematical set. [51]. Since GPs are treated analytically they are relatively simple to handle. Classification and regression, are the two main problems for supervised learning. While classification deals with discrete labels, regression has continuous quantities.

In this chapter, the basics of GPR are presented. GPR is a Bayesian machine learning technique, thus Bayes theorem is applied to obtain the necessary equations for GPR. Therefore, the basics of probability theory are introduced in section 5.1. With this knowledge, how to use GPs for regression is shown in section 5.2. There are two ways to interpret GPR. First, GP can be thought of as a generalization of Gaussian distribution to functions. Thus inference takes place in the space of functions, this is the function-space view. The other equivalent view is the weight-space view, which might be more accessible. As seen in this section, free parameters are left. Those parameters are optimized and thus the GP is trained. This will be explained in section 5.3.

Since GPR provides, as a Bayesian method, a measure for uncertainty. This measure can be used in information theory. Thus in section 5.4, the basics of entropy and information are formulated.

## 5.1 Fundamentals of Probability Theory

A set $\Omega$ is called sample space. Any subset of $\Omega$ is an event. This is well-defined for discrete sample spaces. A probability space $(\Omega, P)$, consisting of the sample space $\Omega$ and a function $P$, is defined on the power set of $\Omega$ into the real space with the following properties [52–54]

1. $P(A) \geq 0, A \subset \Omega$
2. $P(\Omega) = 1$
3. $P\left(\sum\limits_{j=1}^{\infty} A_j\right) = \sum\limits_{j=1}^{\infty} P(A_j)$ for pairwise disjoint sets $A_j$

This function is called probability measure, or simply probability distribution on $\Omega$. $P(A)$ is the probability of the event $A$. In a real random experiment, not everything is known beforehand. Therefore the results of sub-experiments might depend on the previous experiment. This leads to the definition of conditional probability. If $A, B \subset \Omega$ with $P(B) > 0$ then

$$P(A|B) := \frac{P(A \cap B)}{P(B)} \tag{5.1}$$

is the conditional probability of $A$ under the hypothesis or condition $B$ [52]. This defines a conditional distribution under $B$. If $A_1, A_2, \cdots$ are pairwise disjoint events with $\sum\limits_{j=1}^{\infty} A_j = \Omega$ a decomposition of $\Omega$. Then for the event $B$ holds

$$P(B) = \sum_{j=1}^{\infty} P(A_j) P(B|A_j) \tag{5.2}$$

This is the law of total probability. With the help of this law, the theorem of Bayes is obtained [55]

$$P(A_k|B) = \frac{P(A_k)P(B|A_k)}{\sum\limits_{j=1}^{\infty} P(A_j)P(B|A_j)} = \frac{P(A_k)P(B|A_k)}{P(B)} \tag{5.3}$$

where $P(B) > 0$. More on Bayes' theorem is treated later. If $P(A \cap B) = P(A)P(B)$ for two events, then they are called independent regards $P$.

### 5.1.1 Random Variables and Parameters of Distributions

A mapping $X : \Omega \to \Omega'$ is called a $\Omega'$ random variable. If $\Omega' = \mathbb{R}^d$ $X$ is called a $d$-dimensional random vector, and for $\Omega' = \mathbb{R}$ it is simply called random variable [54].

The expectation value of a random variable $X$ exists if

$$\sum_{\omega \in \Omega} |X(\omega)| P(\omega) < \infty \tag{5.4}$$

In that case

$$\mathbb{E}[X] := \sum_{\omega \in \Omega} X(\omega) P(\omega) \tag{5.5}$$

the expectation value of $X$. This can be transformed into

$$\mathbb{E}[X] = \sum_{x \in \mathbb{R}} x P(X = x) \tag{5.6}$$

The message here is that the expectation value is only dependent on the distribution of $X$ and not the probability space. If $\mathbb{E}[X^2]$ exists then

$$\mathbb{V}[X] := \mathbb{E}(X - \mathbb{E}[X])^2 \tag{5.7}$$

is the variance of $X$ and

$$\sigma_X := \sqrt{\mathbb{V}[X]} \tag{5.8}$$

the standard deviation of $X$. For calculation usage, the variance can be written as

$$\mathbb{V}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2. \tag{5.9}$$

If $X$ and $Y$ are random variables on the same probability space and both variances exist, then

$$\text{Cov}(X, Y) := \mathbb{E}[(X - \mathbb{E}X)(Y - \mathbb{E}Y)] \tag{5.10}$$

is the covariance between $X$ and $Y$ If Cov $= 0$ then $X$ and $Y$ are uncorrelated.

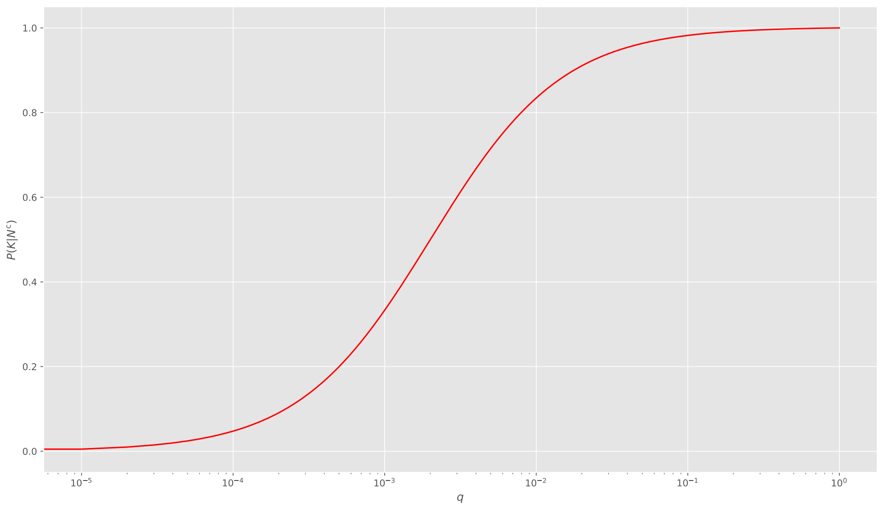


**Figure 5.1:** Probability for an infection with HIV if a positive test result occurs with the a priori probability of being sick $q$.

### 5.1.2 Bayes' Theorem

If there are two events $A$ and $B$ then the a posteriori probability, the conditional probability of $A$ under $B$ is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5.11}$$

Let's consider an example: The interpretation of medical tests. In a medical test false positive as well as false negative results can occur. A false positive result for a certain disease is a false positive diagnosis despite being healthy. If a test result is negative, although the person is sick, is called a false negative. The sensitivity of a test is the probability $p_{\text{se}}$ that a sick person is detected as a person being sick. The probability $p_{\text{sp}}$ is the specificity, which means that a healthy person is detected as a healthy person. A very simple assumption is made. $p_{\text{se}}$ and $p_{\text{sp}}$ are equal for every person. As an example lets consider the ELISA-test for HIV [56], there $p_{\text{se}} = p_{\text{sp}} = 0.998$.

Now a person has made a test for this disease and got a positive result. What is the probability of actually having AIDS? This is dependent on the a priori probability of the person being sick and is $q$. Then Bayes's theorem gives the result. The sample space is $\Omega = \{(0,0),(0,1),(1,0),(1,1)\}$

The first entry is whether the person has the disease (1) or not(0). The second component is the result of the test, positive (1) or negative (0). To calculate the probability of being sick under a positive test, then $K = \{(1,0),(1,1)\}$ is the event of being sick and $N = \{(1,0),(0,0)\}$ the event of a negative test result. Then the assumptions lead to $P(K) = q$, the probability of being sick, $P(N^{\mathrm{c}}|K) = p_{\mathrm{se}}$, which is the probability of a positive (not negative) test under the person being sick (the sensitivity) and $P(N|K^{\mathsf{C}}) = p_{\mathrm{sp}}$ is the probability of a negative test under a healthy (not sick) person (the specificity). Applying Bayes theorem leads to

$$P(K|N^{\mathsf{C}}) = \frac{P(K)P(N^{\mathsf{C}}|K)}{P(N^{\mathsf{C}}|K) + P(K^{\mathsf{C}})P(N^{\mathsf{C}}|K^{\mathsf{C}})} \tag{5.12}$$

Since $K^{\mathrm{c}}$ is the complement of being sick (healthy) $P(K^{\mathsf{C}}) = 1 - q$. $p(N^{\mathrm{c}}|K^{\mathrm{c}})$ is the probability of a positive test (complement to negative test) under a healthy (non-sick) person and is, therefore, the complement to the specificity and $p(N^{\mathrm{c}}|K^{\mathrm{c}}) = 1 - p_{\mathrm{sp}}$. Then eq. (5.12) can be written as

$$P(K|N^{\mathsf{C}}) = \frac{q \cdot p_{\mathrm{se}}}{q \cdot p_{\mathrm{se}} + (1-q)\cdot(1-p_{\mathrm{sp}})} \tag{5.13}$$

For the ELISA-test [56] the dependence of the probability on the risk of being sick $q$ is shown in fig. 5.1.

### 5.1.3 Generalization to Continuous Probability Spaces

To generalize the above-mentioned definitions, measure theory must be applied. Since most distributions handled here are Gaussian (see later), everything is well-defined. There is no need to go into detail. If the reader is interested, a book about probability and measure theory can be found here [57]. The main result is, that only "good" subsets of the whole sample space $\mathcal{A} \subset \Omega$ can be used. Therefore, the focus is only on distributions, where the probability density exits. Then the generalization of the expectation value is, if

$$\int_{\mathcal{A}} |x|p(x)\,\mathrm{d}x < \infty \tag{5.14}$$

exists, then

$$\mathbb{E}X = \int xp(x)\,\mathrm{d}x \tag{5.15}$$

is the expectation value of the random variable $X$.

The joint probability of the random variables $X_1, \cdots, X_N$ have the joint probability $p(X_1,\cdots,X_N) = p(\mathbf{X})$. If the random vector $\mathbf{X}$ is partitioned into $X_A$ and $X_B$, where $A$

and $B$ are disjoint sets and their union is $\mathbf{X}$ then their probability is $p(X_A, X_B)$. Then the marginal probability of $X_A$ is [57]

$$p(X_A) = \int p(X_A, X_B)\,\mathrm{d}X_B \tag{5.16}$$

The random variables are called independent if the joint distribution is equal to the products of marginal probabilities.

The conditional probability function is [57]

$$p(X_A|X_B) = \frac{p(X_A, X_B)}{p(X_B)} \tag{5.17}$$

If $X_A$ and $X_B$ are independent, the n the marginal probability is equal to the conditional probability, i.e. $p(X_A) = p(X_A|X_B)$.

### 5.1.4 The Multivariate Gaussian Distribution

In this section, an example of a continuous random variable is presented, which is the Gaussian distribution. A random variable is Gaussian distributed with $X \sim \mathcal{N}(\mu, \sigma^2)$ and is fully defined by its mean $\mu$ and variance $\sigma^2$. The probability density is given

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{5.18}$$

A plot for different $\sigma^2$ can be seen in fig. 5.2. There three values for the variance $\sigma^2$ are shown. This can be generalized for a random vector $\mathbf{X} \in \mathbb{R}^d$ Then the multivariate Gaussian distribution is defined by its mean vector $\mathbf{m}$ and the covariance matrix $\mathbf{K} \in \mathbb{R}^{d\times d}$ and is written as $\mathbf{X} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$. In the non-degenerate case, the probability density is given by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\,|\mathbf{K}|^{1/2}} \exp\left(-\frac{(\mathbf{x}-\mathbf{m})^\top \mathbf{K}^{-1}(\mathbf{x}-\mathbf{m})}{2}\right) \tag{5.19}$$

In fig. 5.3 an example of the two-dimensional Gaussian distribution is shown. On the top, the covariance matrix is diagonal, thus there is no correlation between $x$ and $y$. On the bottom, the covariance is not diagonal and thus has correlations between $x$ and $y$.

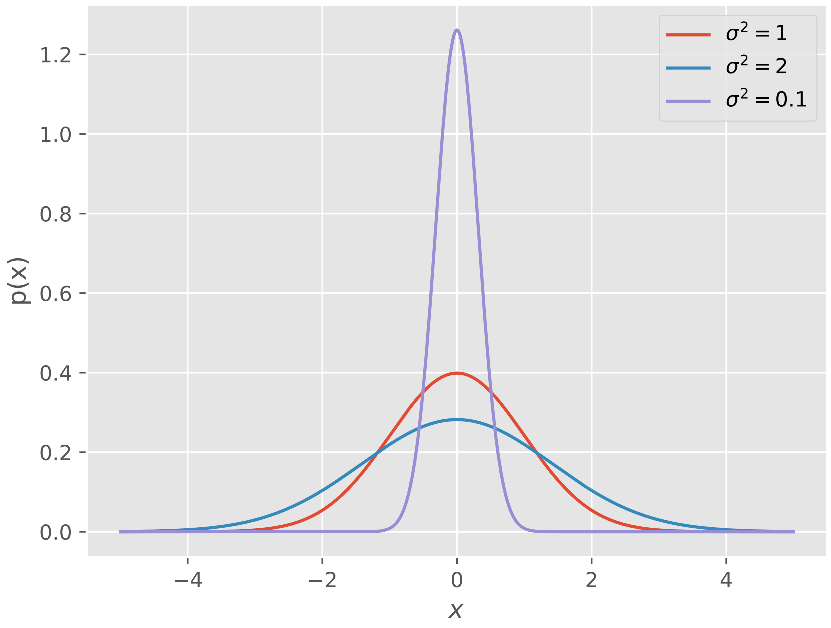


**Figure 5.2:** Probability density function for Gaussian distributions with different variance.

If $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian random vectors, i.e.

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{pmatrix} A & C \\ C^\top & B \end{pmatrix}\right), \tag{5.20}$$

then the marginal distribution of $\mathbf{x}$ is

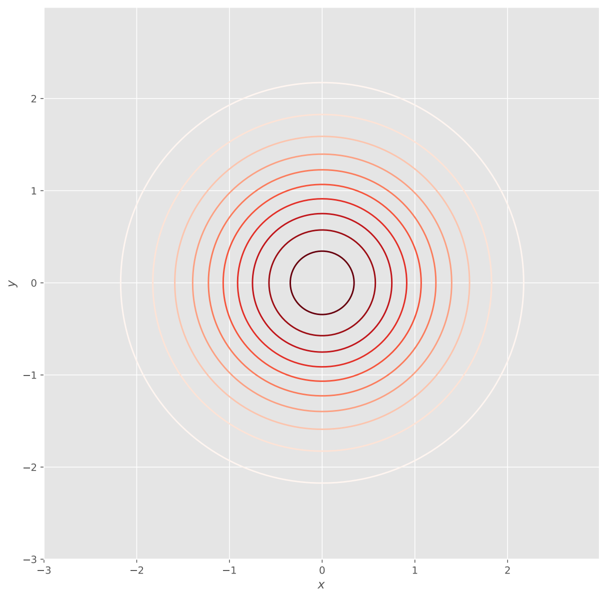
$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, A) \tag{5.21}$$

and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is

$$\mathbf{x}|\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_x + CB^{-1}\left(\mathbf{y} - \boldsymbol{\mu}_y\right), A - CB^{-1}C^\top) \tag{5.22}$$

The derivation of this equation can be seen in [58].

If a product of two Gaussians is made, then another Gaussian is obtained. However, this



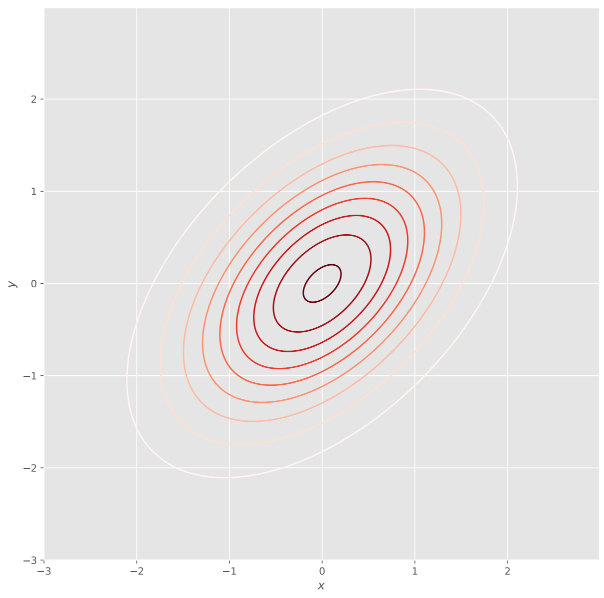


**Figure 5.3:** Probability density of the two dimensional Gaussian distribution. On the top the covariance is diagonal, on the bottom a general covariance matrix is used.

Gaussian is un-normalized [10].

$$\begin{gathered}\mathcal{N}(\mathbf{x}|\mathbf{a}, A)\mathcal{N}(\mathbf{x}|\mathbf{b}, B) = \mathcal{N}(\mathbf{x}|\mathbf{c}, C) \\ \text{with} \quad \mathbf{c} = C\left(A^{-1}\mathbf{a} + B^{-1}\right) \quad \text{and} \quad C = \left(A^{-1} + B^{-1}\right)^{-1}\end{gathered} \tag{5.23}$$

## 5.2 Gaussian Process Regression

There are two equivalent derivations of how to obtain the posterior distribution including the new mean and covariance function. The first one is the weight-space view and the second is the function-space view, where inference is directly in function space. The derivations of this section are based on [10].

### 5.2.1 Weight-Space View

In this section, the weight space view to interpret Gaussian process regression is reviewed. Therefore the linear model, where the outputs are a linear combination of the inputs, is introduced. This might not be a good model if there is no good linear relation between the inputs and outputs. In the beginning, the Bayesian treatment of this model will be applied to get a posterior distribution of the target or outputs given the input values. By projecting the inputs into a so-called "feature space" and applying the "kernel trick" an improvement is obtained. Let $M$ be the number of training points, then the training set is $\mathcal{D} = \{(\mathbf{x}_n, y_n)|n = 1, \cdots M\}$. $\mathbf{x}_n \in \mathbb{R}^d$ are the real input vectors of dimension $d$ and $y_n$ are the scalar outputs, or the dependent variable. Summarizing the input vectors, they can be written as a matrix $X = \{\mathbf{x}_n\}_{n=1}^{M} \in \mathbb{R}^{d \times n}$. The target values can be written as $\mathbf{y} \in \mathbb{R}^n$. Then the training set is $\mathcal{D} = \{X, \mathbf{y}\}$. The point of interest is now to obtain a conditional distribution of the outputs given the inputs and not model the input distribution. In the linear model, the function is a linear combination of the inputs

$$f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\omega} = \sum_{i=1}^{d} \omega_i x_i. \tag{5.24}$$

Since the function values can only be observed with Gaussian noise, the observations or targets are

$$y_n = f(\mathbf{x}) + \epsilon_n \tag{5.25}$$

Assuming this noise is an independent, identically distributed (i.i.d.) Gaussian distribution

$$\epsilon_n \sim \mathcal{N}(0, \sigma^2) \tag{5.26}$$

With this noise the probability density of the observation given the parameters, which can be factorized due to the independence assumption and the fact that $\epsilon_n = y_n - \underbrace{\mathbf{x}_n^\top \boldsymbol{\omega}}_{f(\mathbf{x}_n)}$, is

$$p(\mathbf{y}|\boldsymbol{\omega}, X) = \prod_{n=1}^{M} p(y_n|\boldsymbol{\omega}, \mathbf{x}_n) = \prod_{n=1}^{M} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - \mathbf{x}_n^\top \boldsymbol{\omega})^\mathbf{2}}{2\sigma^2}\right) \tag{5.27}$$

$$= \frac{1}{(2\pi\sigma^2)^{M/2}} \exp\left(-\frac{|\mathbf{y} - X^\top \boldsymbol{\omega}|^2}{2\sigma^2}\right) = \mathcal{N}(X^\top \boldsymbol{\omega}, \sigma^2 \mathbf{I}) \tag{5.28}$$

To apply Bayes' theorem a prior for the parameters is required, that is having a belief or knowledge about the parameters before looking at the observations). The assumption is, that the parameters are multivariate Gaussian distributed with zero mean vector and covariance matrix $\mathbf{\Sigma}$, this is

$$\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \tag{5.29}$$

Now the posterior distribution can be calculated

$$p(\boldsymbol{\omega}|\mathbf{y}, X) = \frac{1}{Z} \exp\left(-\frac{1}{2\sigma^2} \left(\mathbf{y} - X^\top \boldsymbol{\omega}\right)^\top \left(\mathbf{y} - X^\top \boldsymbol{\omega}\right)\right) \exp\left(-\frac{1}{2} \boldsymbol{\omega}^\top \mathbf{\Sigma}^{-1} \boldsymbol{\omega}\right) \tag{5.30}$$

$$= \frac{1}{Z} \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2} \left(\mathbf{y} - X^\top \boldsymbol{\omega}\right)^\top \left(\mathbf{y} - X^\top \boldsymbol{\omega}\right) + \boldsymbol{\omega}^\top \mathbf{\Sigma}^{-1} \boldsymbol{\omega}\right)\right) \tag{5.31}$$

$$= \frac{1}{Z} \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2} \left(\mathbf{y}^2 - \mathbf{y} X^\top \boldsymbol{\omega} - \boldsymbol{\omega}^\top X \mathbf{y} + \boldsymbol{\omega}^\top X X^\top \boldsymbol{\omega}\right) + \boldsymbol{\omega}^\top \mathbf{\Sigma}^{-1} \boldsymbol{\omega}\right)\right) \tag{5.32}$$

where $\left(X^\top \boldsymbol{\omega}\right)^\top = \boldsymbol{\omega}^\top X$ is used. Z is a normalization constant, which does not depend on $\boldsymbol{\omega}$. Introducing $\mathbf{A} = \left(\frac{1}{\sigma^2} X X^\top + \mathbf{\Sigma}^{-1}\right)$ and using $\mathbf{A} = \mathbf{A}^\top$, because $XX^\top$ is a symmetric matrix, $\mathbf{\Sigma}$ is symmetric by definition and the sum of two symmetric matrices leads to a symmetric matrix.

$$p(\boldsymbol{\omega}|\mathbf{y}, X) = \frac{1}{Z} \exp\left(-\frac{1}{2}\left(\boldsymbol{\omega}^\top \mathbf{A} \boldsymbol{\omega} - \frac{1}{\sigma^2} \boldsymbol{\omega}^\top X \mathbf{y} - \frac{1}{\sigma^2} \mathbf{y} X^\top \boldsymbol{\omega} + \frac{\mathbf{y}^\mathbf{2}}{\sigma^2}\right)\right) \tag{5.33}$$

Now $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ can be inserted

$$p(\boldsymbol{\omega}|\mathbf{y}, X) = \frac{1}{Z} \exp\left(-\frac{1}{2}\left(\boldsymbol{\omega}^\top \mathbf{A} \boldsymbol{\omega} - \boldsymbol{\omega}^\top \mathbf{A} \left(\frac{1}{\sigma^2} \mathbf{A}^{-1} X \mathbf{y}\right) - \left(\frac{\mathbf{A}^{-1}}{\sigma^2} X \mathbf{y}\right)^\top \mathbf{A} \boldsymbol{\omega} + \frac{\mathbf{y}^\mathbf{2}}{\sigma^2}\right)\right) \tag{5.34}$$

Introducing $\mathbf{m} = \frac{1}{\sigma^2}\mathbf{A}^{-1}X\mathbf{y}$ leads to

$$p(\boldsymbol{\omega}|\mathbf{y}, X) = \frac{1}{Z}\exp\left(-\frac{1}{2}\left(\boldsymbol{\omega}^\top\mathbf{A}\boldsymbol{\omega} - \boldsymbol{\omega}^\top\mathbf{A}\mathbf{m} - \mathbf{m}^\top\mathbf{A}\boldsymbol{\omega} + \frac{\mathbf{y^2}}{\sigma^2}\right)\right) \tag{5.35}$$

Since the last term does not depend on $\boldsymbol{\omega}$ and introducing $0 = \mathbf{m}^\top\mathbf{A}\mathbf{m} - \mathbf{m}^\top\mathbf{A}\mathbf{m}$, these terms can be moved into the normalizaiton constant $Z$ and then

$$p(\boldsymbol{\omega}|\mathbf{y}, X) = \frac{1}{z}\exp\left(-\frac{1}{2}\left(\boldsymbol{\omega} - \mathbf{m}\right)^\top K\left(\boldsymbol{\omega} - \mathbf{m}\right)\right) \tag{5.36}$$

Thus the posterior is a Gaussian with mean $\mathbf{m}$ and covariance matrix $K^{-1}$:

$$p(\boldsymbol{\omega}|\mathbf{y}, X) \sim \mathcal{N}(\mathbf{m} = \frac{1}{\sigma^2}A^{-1}X\mathbf{y}, A^{-1}) \tag{5.37}$$

where $A = \sigma_n^{-2}XX^\top + \Sigma^{-1}$. Averaging over all possible parameters weighted by the posterior distribution is needed to make predictions. Thus, the predictive distribution of $f_\star := f(\mathbf{x}_\star)$ at $\mathbf{x}_\star$ is received by the average of the outputs for all possible (linear) models with respect to the posterior, calculated above

$$p(f_\star|\mathbf{x}_\star, X, \mathbf{y}) = \int p(f_\star|\mathbf{x}_\star, \boldsymbol{\omega})p(\boldsymbol{\omega}|X, \mathbf{y}) = \mathcal{N}(\sigma_n^{-2}\mathbf{x}_\star^\top A^{-1}X\mathbf{y}, \mathbf{x}_\star^\top A^{-1}\mathbf{x}_\star) \tag{5.38}$$

which is again a Gaussian distribution.

Up to now, only a linear model was considered. However, this suffers from limited expressiveness. To overcome this problem, the inputs are projected into a high-dimensional feature space using basis functions. The linear model is then applied in the feature space and not in the input space. If these functions for projection are fixed, i.e. independent of $\boldsymbol{\omega}$, the model is linear in the parameters and all the calculated distributions are analytically tractable. The answer for choosing the basis function is the GP formalism. Therefore, the basis function. are given for now.

Introducing the function $\phi(\mathbf{x})$, that maps from the $d$-dimensional input space into a $N$-dimensional feature space, leads to the following linear model

$$f(\mathbf{x}) = \phi(\mathbf{x})^\top\boldsymbol{\omega} \tag{5.39}$$

The matrix $\Phi(X)$ is now the columns of$\phi(\mathbf{x})$ for all training points. Replacing $\Phi(X)$ for $X$ in the predictive distribution eq. (5.38) leads to

$$f_\star|\mathbf{x}_\star, X, \mathbf{y} \sim \mathcal{N}(\sigma_n^{-2})\phi(\mathbf{x}_\star)^\top A^{-1}\Phi(X)\mathbf{y}, \phi(\mathbf{x}_\star)^\top A^{-1}\phi(\mathbf{x}_\star) \tag{5.40}$$

with $A = \sigma_n^{-2}\Phi(X)\Phi(X)^\top + \Sigma^{-1}$. To make predicion the inversion of matrix $A$ is required, which is not feasible if $N$ is large. Thus, eq. (5.40) is rewritten, the derivation can be seen in [10],

$$f_\star|\mathbf{x}_\star, X, \mathbf{y} \sim \mathcal{N}(\phi_\star^\top \Sigma\Phi \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} \mathbf{y}, \phi_\star^\top \Sigma\phi_\star - \phi_\star^\top \Sigma\Phi \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} \Phi\Sigma\phi_\star) \tag{5.41}$$

where $\mathbf{K} = \Phi^\top\Sigma\Phi$. This time, a $n \times n$ matrix has to be inverted and is useful if $n < N$. Therefore, the predictive mean is

$$\mathbb{E}[f|\mathbf{x}, X, \mathbf{y}] = \phi(\mathbf{x})^\top \Sigma\Phi(X) \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} \mathbf{y} \tag{5.42}$$

and the variance of the predictive distribution eq. (5.41) is

$$\mathbb{V}[f|\mathbf{x}, X, \mathbf{y}] = \phi(\mathbf{x})^\top \Sigma\phi(\mathbf{x}) - \phi(\mathbf{x})^\top \Sigma\Phi(X) \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} \Phi(X)^\top \Sigma\phi(\mathbf{x}) \tag{5.43}$$

For a given basis set, predictions can now be made. By projecting into feature space, the linear Bayesian model can be used for regression through eq. (5.42).

Note, that always terms like $\Phi^\top\Sigma\Phi$, $\phi_\star^\top\Sigma\Phi$ and $\phi_\star^\top\Sigma\phi_\star$ occur. Thus, term of the form $\phi(\mathbf{x})^\top\Sigma\phi(\mathbf{x}')$ are entries of these matrices. This can be used to define $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\Sigma\phi(\mathbf{x}')$, which is a inner product. $k$ will later be called covariance function or kernel. Since $\Sigma$ is positive definit, $\Sigma^{1/2}$ can be defined, for example with SVD. Using the definition $\psi(\mathbf{x}) = \Sigma^{1/2}\phi(\mathbf{x})$ a inner product representation is obtained, i.e. $k(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x}) \cdot \psi(\mathbf{x}')$. This is called the kernel trick. Then the input space can be lifted into feature space by replacing these inner products by $k(\mathbf{x}, \mathbf{x}')$.

### 5.2.2 Function-Space View

Alternatively to achieve identical results inference can be considered in function space. In this case, GPs are used. Therefore, the following definition is used [10]

A Gaussian process is a collection of random variables, where finite subsets have joint Gaussian distribution.

Similar to Gaussian distribution, a GP is defined by a mean function $m(\mathbf{x})$ and a covariance

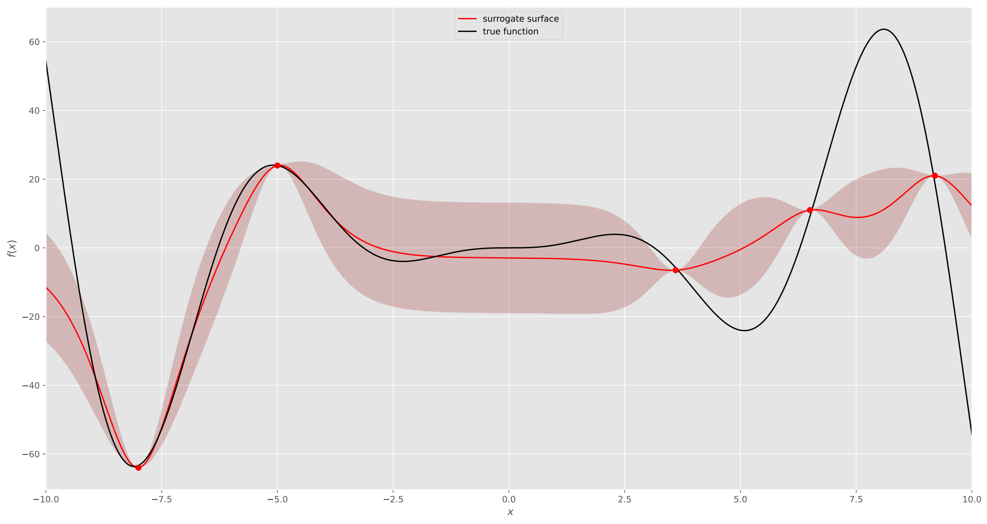


**Figure 5.4:** The red line is the mean function from eq. (5.56) for the underlying function, the black line. Five training points are used, these are the red dots. The red shaded area is the mean function plus/minus the standard deviation $\sqrt{\mathbb{V}}$.

function $k(\mathbf{x}_n, \mathbf{x}_m)$ of a process $f(x)$:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{5.44}$$

$$k(\mathbf{x}_n, \mathbf{x}_m) = \mathbb{E}\left[(f(\mathbf{x}_n - m(\mathbf{x}_n)(f(\mathbf{x}_m - m(\mathbf{x}_m)\right] \tag{5.45}$$

and is noted as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}, k(\mathbf{x}_n, \mathbf{x}_m)) \tag{5.46}$$

For simplicity, in the following the mean is set to zero, i.e. $m = 0$.

From the definition of a GP, where a collection of random variables are mentioned, a certain consistency is required. This is the marginalization property. Thus, if $(y_1, y_2) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ are specified by the GP then $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ is specified too. Hereby $\Sigma_{11}$ is the corresponding submatrix of $\Sigma$ according to eq. (5.21).

An example is the Bayesian liner model $f(\mathbf{x}) = \Phi(\mathbf{x})^\top \boldsymbol{\omega}$, where $\boldsymbol{\omega} \sim \mathcal{N}(0, \Sigma)$ is the prior. Then the mean and covariance function are

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= \phi(\mathbf{x})\mathbb{E}[\boldsymbol{\omega}] = 0 \\ \mathbb{E}[f(\mathbf{x}), f(\mathbf{x}')] &= \phi(\mathbf{x})^\top \mathbb{E}[\boldsymbol{\omega}\boldsymbol{\omega}^\top]\phi(\mathbf{x}') = \phi(\mathbf{x})^\top \Sigma \phi(\mathbf{x}') \end{aligned} \tag{5.47}$$

Then $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian with zero mean and covariance $\phi(\mathbf{x})^\top \Sigma \phi(\mathbf{x}')$. As can be seen, the covariance function specifies the covariance between pairs of random variables

$$\mathrm{cov}(f(\mathbf{x}_n), f(\mathbf{x}_m)) = k(\mathbf{x}_n, \mathbf{x}_m) \tag{5.48}$$

and thus the covariance between the outputs is a function of the inputs. Specifying the covariance function implies a distribution over functions.

First, consider the special case of noise-free observations, i.e. $y_n = f_n$ and therefore $\{(\mathbf{x}_n, f_n)|n = 1, \cdots, M\}$ is known. According to the prior,

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_\star \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X_\star) \\ K(X_\star,X) & K(X_\star,X_\star) \end{bmatrix}\right) \tag{5.49}$$

is the joint distribution between training outputs $\mathbf{f}$ and test outputs $\mathbf{f}_\star$, where $M$ training points and $M_\star$ test points are included. $K(X, X_\star) \in \mathbb{R}^{M \times M_\star}$ is the covariance between all pairs of test and training points. Similar $K(X,X) \in \mathbb{R}^{M\times M}$, $K(X_\star, X) \in \mathbb{R}^{M_\star \times M}$ and $K(X_\star, X_\star) \in \mathbb{R}^{M_\star \times M_\star}$ are defined as the covariance between all pairs of training points, between all training and test points and between all test points. To obtain the posterior distribution, the prior has to be restricted to only those functions, which match the observations. This corresponds to conditioning the joint Gaussian prior to the observations.

$$\begin{aligned} \mathbf{f}_\star | X_\star, X, \mathbf{f} \sim \mathcal{N}\big(&K(X_\star,X)K(X,X)^{-1}\mathbf{f}, \\ &K(X_\star,X_\star) - K(X_\star,X)K(X,X)^{-1}K(X,X_\star)\big) \end{aligned} \tag{5.50}$$

The function values are then obtained by evaluating the mean and covariance function form the joint posterior distribution.

A more realistic model is, that function values can only be observed through noise, i.e. $y = f(\mathbf{x}) + \epsilon$. If the noise is independent, identical distributed Gaussian $\epsilon$ with variance $\sigma_M^2$, the prior is obtained

$$\mathrm{cov}(y_n, y_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \sigma_M^2 \delta_{nm} \quad \text{or} \quad \mathrm{cov}\mathbf{y} = K(X,X) + \sigma_M^2 \mathbf{I} \tag{5.51}$$

With this noise term, the distribution of observed target values and function values at the test points is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_\star \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_M^2 \mathbf{I} & K(X,X_\star) \\ K(X_\star,X) & K(X_\star,X\star) \end{bmatrix}\right) \tag{5.52}$$

The conditional distribution is the predictive distribution of Gaussian process regression

$$\mathbf{f}_\star | X_\star, X, \mathbf{y} \sim \mathcal{N}\left(\bar{\mathbf{f}}_\star, \mathrm{cov}(\mathbf{f}_\star)\right) \tag{5.53}$$

with

$$\begin{aligned} \bar{\mathbf{f}}_\star &= \mathbb{E}[\mathbf{f}_\star | X_\star, X, \mathbf{y}] = K(X_\star, X)\left(K(X,X) + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y} \\ \mathrm{cov}(\mathbf{f}_\star) &= K(X_\star, X_\star) - K(X_\star, X)\left(K(X,X) + \sigma^2\mathbf{I}\right)^{-1} K(X, X_\star) \end{aligned} \tag{5.54}$$

Identifying $K(A,B) = \Phi(A)^\top \Sigma \Phi(B)$ with $A, B$ either $X_\star$ or $X$ the same result from eq. (5.41) is obtained.

If only one test $\mathbf{x}$ is present, then $\mathbf{k}_\star$ is introduced as the vector between this test point and all training points. With this notation, eq. (5.54) can be written as

$$\begin{aligned} \bar{f} &= \mathbb{E}[f(\mathbf{x})] = \mathbf{k}_\star^\top \left(K + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y} \\ \mathbb{V}[f(\mathbf{x})] &= k(\mathbf{x}, \mathbf{x}) - \mathbf{x}_\star^\top \left(K + \sigma^2\mathbf{I}\right)^{-1}\mathbf{k}_\star \end{aligned} \tag{5.55}$$

The mean is a linear combination of the observations, which can also be written as a linear combination of covariance functions

$$\mathbb{E}[f(\mathbf{x})] = \sum_{n=1}^{M} \alpha_n k(\mathbf{x}, \mathbf{x}_n) \tag{5.56}$$

where $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_M)^\top$ is the solution of the linear system of equations $\boldsymbol{\alpha}\left(K + \sigma^2\mathbf{I}\right) = \mathbf{y}$.

The variance in eq. (5.55) depends on the inputs but not on the observations, which is a property of Gaussian distributions. The left term in the variance eq. (5.55) is the prior covariance evaluated at the test points. From that term a positive term, $\mathbf{x}_\star^\top \left(K + \sigma^2\mathbf{I}\right)^{-1}\mathbf{k}_\star$, is subtracted, which is the information, the observations give about the function.

An example of the predicted mean can be seen in fig. 5.4. There, the mean from eq. (5.56) is calculated from data of an underlying function, shown as a black line. Five training points are used. In addition in the red-shaded area the mean plus/minus the standard deviation $\sqrt{\mathbb{V}}$ is plotted.

## 5.2.3 Covariance Function

The covariance function $k$ is an essential part of GPs. They encode assumptions made about the function to learn from. If input points are close together, they usually have similar target or observation values. This similarity is defined by the covariance matrix of the GP. Any function of $\mathbf{x}_n$ and $\mathbf{x}_m$ is a valid covariance function if it is positive semi-definite. A covariance function $k$ is stationary if is a function of $\mathbf{x}_n - \mathbf{x}_m$, i.e. $k(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n - \mathbf{x}_m)$ [10]. Thus it is invariant under translation in the input space. If the covariance function $k$ depends only on the distance $r = |\mathbf{x}_n - \mathbf{x}_m|$, then it is called an isotropic covariance function [10]. $k(r)$ is then also called a radial basis function.

For a given set $\{x_n | n = 1, \cdots, M\}$ the Gram matrix or covariance matrix $\mathbf{K}$ can be calculated with entries $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$. If $\mathbf{K} \in \mathbb{R}^{M \times M}$. satisfies $\mathbf{v}^\top \mathbf{K} \mathbf{v} \geq 0 \; \forall \mathbf{v} \in \mathbb{R}^m$, then this matrix is called positive semi-definite. A symmetric matrix is positive semi- definite if all the eigenvalues are larger or equal to zero and positive definite if they are only positive and non-zero. Covariance functions must be, by definition, positive semi-definite. The definition of positive semi-definiteness for matrices can be generalized for covariance function [10]

$$\int k(\mathbf{x}_n, \mathbf{x}_m) f(\mathbf{x}_n) f(\mathbf{x}_m) \, \mathrm{d}\mathbf{x}_n \, \mathrm{d}\mathbf{x}_m \geq 0 \quad \forall f \in L_2(\mathbb{R}^M). \tag{5.57}$$

In the following two examples of covariance functions will be presented namely the squared exponential and the Matérn covariance function. They are both plotted in fig. 5.5.

### Squared Exponential Covariance Function

The squared exponential function is defined as [10]

$$k_{\mathrm{SE}}(r) = \sigma_f^2 \exp\left(-\frac{r^2}{2\ell^2}\right), \tag{5.58}$$

where $\ell$ is the characteristic length scale. The parameter $\sigma_f$ is a scaling parameter. Since this covariance function is infinitely differentiable, the Gaussian process with this covariance function is very smooth. The squared exponential is one of the most widely used covariance functions used. Stein [59] suggested, that the squared exponential might not be so good for physical properties and that the Matérn covariance function works better.

### Matérn Covariance Function

The classes of the Matérn covariance functions is give by [60]

$$k_{\text{Matérn}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2}\nu r}{\ell}\right)^{\nu} K_\nu \left(\frac{\sqrt{2}\nu r}{\ell}\right) \tag{5.59}$$

with $\nu, \ell > 0$ parameters and $K_\nu$ is a modified Bessel function [61]. For $\nu \to \infty$ the squared exponential is obtained. The Gaussian process for the Matérn covariance function is $k$-times differentiable if $\nu > k$. [10]. Using $\nu = p + 1/2$ leads to simple Matérn covariance functions, the general expression is then, derived from [61],

$$k_{\nu=p+1/2}(r) = \exp\left(-\frac{\sqrt{2\nu}r}{\ell}\right)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\left(\frac{\sqrt{8\nu}r}{\ell}\right)^{p-i} \tag{5.60}$$

The most common used are $\nu = 1/2$, $\nu = 3/2$ and $\nu = 5/2$. They are then given as [10]

$$k_{v=1/2}(r) = \sigma_f^2 \exp\left(-\frac{r}{\ell}\right) \tag{5.61}$$

$$k_{v=3/2}(r) = \sigma_f^2\left(1 + \frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right) \tag{5.62}$$

$$k_M(r) = k_{v=5/2}(r) = \sigma_f^2\left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}r}{\ell}\right) \tag{5.63}$$

### 5.2.4 Including Derivative Information

The ab initio programs to calculate the PES, for which the GPR is used as a surrogate surface, usually provide not only the function value (Energy) but also the gradient. Therefor the observed targets are $\mathbf{y} = \left(f(\mathbf{x}_1), \cdots f(\mathbf{x}_M), \frac{\partial f(\mathbf{x}_1)}{\partial x_{1,1}}, \cdots \frac{\partial f(\mathbf{x}_M)}{\partial x_{M,d}}\right)^\top$. Since the partial derivative of $f(\mathbf{x})$ is a linear operation, this can be included in GPR, see Solak et al. [62]. The corresponding covariance entries can be calculated by

$$\operatorname{cov}\left(f(\mathbf{x}_n), \frac{\partial f(\mathbf{x}_m)}{\partial x_{m,k}}\right) = \frac{\partial k(\mathbf{x}_n, \mathbf{x}_m)}{\partial x_{m,k}} \quad \operatorname{cov}\left(\frac{\partial f(\mathbf{x}_n)}{\partial x_{n,l}}, \frac{\partial f(\mathbf{x}_m)}{\partial x_{m,k}}\right) = \frac{\partial^2 k(\mathbf{x}_n, \mathbf{x}_m)}{\partial x_{n,l}\partial x_{m,k}} \tag{5.64}$$

They are included by calculating derivatives of the covariance function. This leads to an extension of the covariance matrix:

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_n, \mathbf{x}_m) & \nabla_{\mathbf{x}_n} k(\mathbf{x}_n, \mathbf{x}_m) \\ \nabla_{\mathbf{x}_m} k(\mathbf{x}_n, \mathbf{x}_m) & \frac{\partial^2 k(\mathbf{x}_n, \mathbf{x}_m)}{\partial \mathbf{x}_n \partial \mathbf{x}_m} \end{pmatrix} \tag{5.65}$$

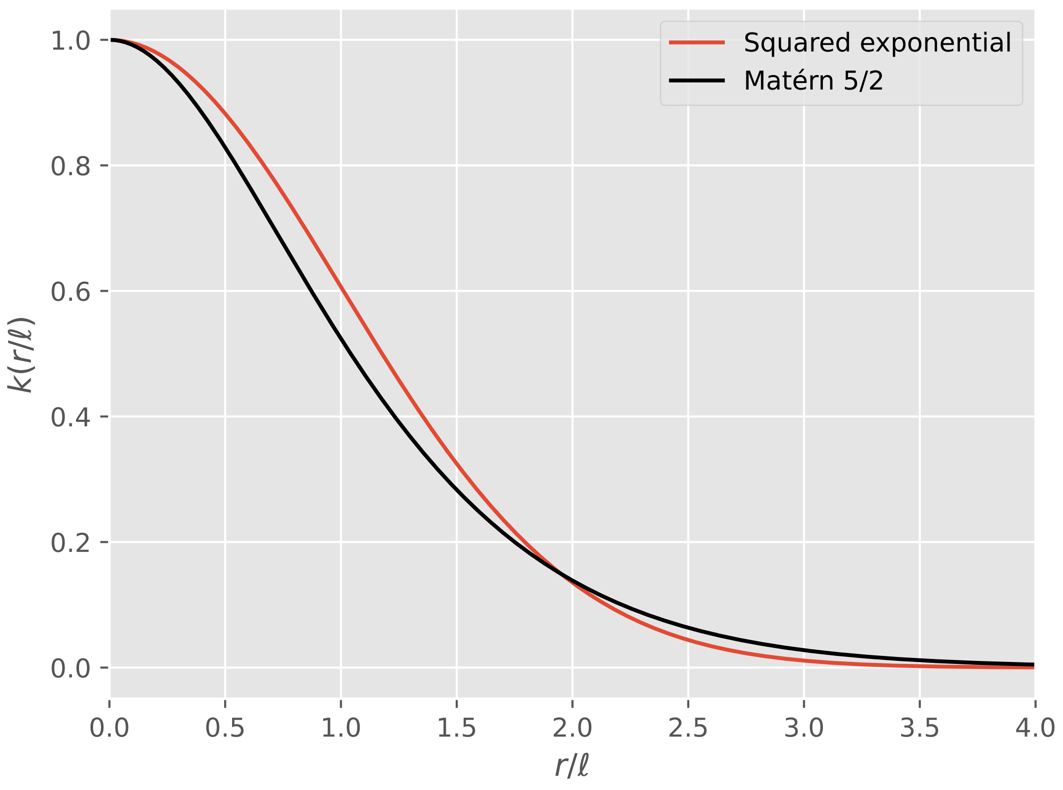


**Figure 5.5:** The squared exponential and the Matérn 5/2 covariance function plotted for normalized $r/\ell$.

The predictive mean function then is

$$f(\mathbf{x}) = \sum_{n=1}^{M} \alpha_n k(\mathbf{x}, \mathbf{x}_n) + \sum_{n=1}^{M} \sum_{i=1}^{d} \beta_{n,i} \frac{\partial k(\mathbf{x}, \mathbf{x}_n)}{\partial x_{n,i}} \tag{5.66}$$

where the weights $\boldsymbol{\omega} = (\alpha_1, \cdots \alpha_M, \beta_{1,1}, \cdots \beta_{M,d})^\top$ are obtained from solving $\boldsymbol{\omega} = (\mathbf{K} + \boldsymbol{\sigma}^2\mathbf{I})^{-1}\mathbf{y}$. Here, separate noises are used, i.e. $\boldsymbol{\sigma} = (\sigma_e, \sigma_g)$, where $\sigma_e$ is the noise for the energy or function targets and $\sigma_g$ for the gradient targets. The variance becomes

$$\mathbb{V}(x) = k(\mathbf{x}, \mathbf{x}) - \sum_{n=1}^{M} \tilde{\alpha}_n k(\mathbf{x}, \mathbf{x}_n) - \sum_{n=1}^{M} \sum_{i=1}^{d} \tilde{\beta}_{n,i} \frac{\partial k(\mathbf{x}, \mathbf{x}_n)}{\partial x_{n,i}} \tag{5.67}$$

where the weights are

$$\tilde{\boldsymbol{\omega}} = \left(\tilde{\alpha}_1, \cdots, \tilde{\alpha}_M, \tilde{\beta}_{1,1}, \cdots \tilde{\beta}_{M,d}\right)^\top. \tag{5.68}$$

These are obtained by solving $(\mathbf{K} + \boldsymbol{\sigma}^2\mathbf{I})\,\tilde{\boldsymbol{\omega}} = \mathbf{k}$. with the vector $\mathbf{k}$ calculated as

$$\mathbf{k} = \left(k(\mathbf{x}, \mathbf{x}_1), \cdots, k(\mathbf{x}, \mathbf{x}_M), \frac{\partial k(\mathbf{x}, \mathbf{x}_1)}{\partial x_{1,1}}, \cdots, \frac{\partial k(\mathbf{x}, \mathbf{x}_M)}{\partial x_{M,d}}\right)^\top \tag{5.69}$$

## 5.3 Training of a Gaussian Process

As previously seen there are at least three hyperparameters involved covariance function: The length scale $\ell$, the noise $\sigma^2$, and the scaling of the covariance function $\sigma_f^2$. Because $\sigma_f$ does not change the mean function, this is neglected here. However, it can have a scaling effect on the variance and is relevant for the dimension of the posterior mean and variance. The hyperparameters are combined in $\mathbf{\Theta} = (\ell, \sigma^2)$.

In general, only vague information about the free hyperparameters $\mathbf{\Theta}$ is known. Therefore, having methods is essential to address the model selection problem, in this case, the choice of the hyperparameters. Thus the selection of the covariance function and the choice of its parameters is often called the training of a GP. This is done based on the training data, where inference about the hyperparameters of the covariance function is made. Therefore, the Bayesian model selection is chosen, where the explicit dependence on the hyperparameters is included [10].

$$p(\boldsymbol{\omega}|\mathbf{y}, X, \mathbf{\Theta}) = \frac{p(\mathbf{y}|X, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{\Theta})}{p(\mathbf{y}|X, \mathbf{\Theta})} \tag{5.70}$$

where the normalizing constant is independent of the parameters and is called the marginal likelihood

$$p(\mathbf{y}|X, \mathbf{\Theta}) = \int p(\mathbf{y}|X, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{\Theta})\,\mathrm{d}\boldsymbol{\omega} \tag{5.71}$$

This includes a trade-off between model fit and model complexity. Simple models only account for a limited range of the target values, while complex models account for a wider range of data sets. The marginal likelihood tends to favor the least complex model, which explains the data.

The eq. (5.71) can be calculated and the logarithm of the marginal likelihood is given by [10]

$$\log p(\mathbf{y}|X, \mathbf{\Theta}) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_y \mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{M}{2}\log(2\pi) \tag{5.72}$$

where $\mathbf{K}_y = \mathbf{K} + \sigma^2\mathbf{I}$ is the covariance matrix for the noisy targets $\mathbf{y}$. Here the marginal likelihood is explicitly conditioned on the hyperparameters $\mathbf{\Theta}$ and is the marginalization over latent functions. The first term is the only term depending on the observed targets, the data fit $-\frac{1}{2}\mathbf{y}^\top\mathbf{K}_y\mathbf{y}$. $-\frac{1}{2}\log|\mathbf{K}_y|$ is the complex penalty term, which only depends on the covariance function and the inputs. The last term is a normalization term. Usually, the more data the more peaked the marginal likelihood is. For a small number of training points, it is rather shallow, because little data are observed. If there are more data, the complex penalty term becomes more severe. Note, that the marginal likelihood can have multiple local optima and every one corresponds to an interpretation of the data.

## 5.4 Information Theory and Entropy

In this section, the concepts of entropy in information theory is given.

The entropy $\mathbb{S}[p(\mathbf{x}]$ of a distribution $p$ measures the uncertainty within the distribution, or a measure of surprise in the distribution. It is defined as [63]

$$\mathbb{S}[p(\mathbf{x}] = -\int p(\mathbf{x}) \log\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{5.73}$$

For a one dimensional Gaussian distribution with mean $\mu$ and variance $\sigma^2$, the entropy is

$$\begin{aligned}\mathbb{S}[\mathcal{N}(\mu,\sigma^2)] &= -\int\limits_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \log\left(\frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}\right), \mathrm{d}x \\ &= \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}\int\limits_{-\infty}^{\infty} \frac{(x-\mu)^2}{2\sigma^2} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x \\ &= \frac{1}{2}\big(\log(2\pi\sigma^2)+1\big) = \frac{1}{2}\log(2\pi e\sigma^2)\end{aligned} \tag{5.74}$$

The relative entropy Kullback-Leibler (KL) divergence $\mathrm{KL}(p||q)$ is defined between two distributions $p(x)$ and $q(x)$ as [64, 65]

$$\mathrm{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)}\, \mathrm{d}x \tag{5.75}$$

For a Gaussian distribution $q(x) = \mathcal{N}(\mu,\sigma^2)$ the KL divergence is

$$\mathrm{KL}(p||q) = \int \frac{(x-\mu)^2}{2\sigma^2} p(x)\, \mathrm{d}x + \frac{1}{2}\log(2\pi e\sigma^2) + \int p(x)\log p(x)\, \mathrm{d}x \tag{5.76}$$

The optimal $q$ can be obtained by differentiating eq. (5.76) with respect to $\mu$ and $\sigma^2$ and setting the derivatives to zero. If the first two moments matches between $p$ and $q$, the optimal $p$ is obtained [10].

### 5.4.1 Conditional Entropy

Consider the random variables $X$ and $Y$. They have the joint probability density function $p(x, y)$. Then the conditional differential entropy is defined as [66]

$$\mathbb{S}[X|Y] := -\int p(x,y) \log\left(p(x|y)\right)\, \mathrm{d}x\, \mathrm{d}y \tag{5.77}$$

There is a chain rule for the conditional entropy [66]

$$\mathbb{S}[X|Y] = \mathbb{S}[X,Y] - \mathbb{S}[X] \tag{5.78}$$

where the joint entropy $\mathbb{S}[X, Y]$ is defined as [66]

$$\mathbb{S}[X,Y] = -\int p(x,y) \log p(x,y)\, \mathrm{d}x\, \mathrm{d}y. \tag{5.79}$$

The conditional entropy and the joint entropy have a connection to the mutual information.

### 5.4.2 Mutual Information

The mutual information measures the dependence between two variables, i.e. it specified the information obtained from one random variable by observing the other. Thus it measures the information, that $x$ and $y$ share. It is defined by the KL divergence between the joint distribution of $x$ and $y$, $p(x, y)$ and the marginal distributions $p(x)$ and $p(y)$ [66]

$$I(x;y) = \mathrm{KL}(p(x,y)||p(x)p(y) = \int\int p(x,y) \log \frac{p(x,y)}{p(x)p(y)}\, \mathrm{d}x\, \mathrm{d}y \tag{5.80}$$

Moreover, the mutual information is symmetric [66], i.e. $I(x; y) = I(y; x)$

With these definitions the mutual information eq. (5.80) can be written as

$$\begin{aligned} I(X;Y) &= \mathbb{S}[X] - \mathbb{S}[X|Y] \\ &= \mathbb{S}[Y] - \mathbb{S}[Y|X] \\ &= \mathbb{S}[X] + \mathbb{S}[Y] - \mathbb{S}[X,Y] \\ &= \mathbb{S}[X,Y] - \mathbb{S}[X|Y] - \mathbb{S}[Y|X] \end{aligned} \tag{5.81}$$

where $\mathbb{S}[X]$ and $\mathbb{S}[Y]$ are the marginal entropies. A Ven-diagram in fig. 5.6 shows this relationship between the entropies of random variable $X$ and $Y$ and the mutual information.

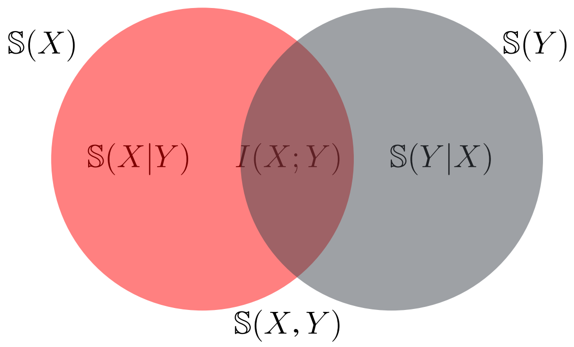


**Figure 5.6:** Relation between entropies and mutual information.

# Part II

# Algorithms Based on Gaussian Process Regression in Internal Coordinates

# 6 Geometry Optimization Based on Gaussian Process Regression in Internal Coordinates

The first part of this chapter, i.e. section 6.1 and section 6.2, is based on "Daniel Born and Johannes Kštner. Geometry optimization in internal coordinates based on gaussian process regression: Comparison of two approaches. Journal of Chemical Theory and Computation, 17(9):59555967, 2021.".

At the beginning of section 6.1, a general introduction to geometry optimization based on Gaussian process regression (GPR) is given. The training data consist of the geometries of the molecules $\mathbf{x}$ as input and energies $E(\mathbf{x})$ and gradients $\nabla E(\mathbf{x})$ of the geometries as observations or labels. In the last part of section 6.1, the combination of GPR based optimization and the benefits of internal coordinates is described.

After explaining the details of the algorithm the performance is tested on 31 test systems containing 30 small molecules and one rhodium complex, containing 78 atoms. These results are presented in section 6.2. In addition the influence of the hyperparameters on the optimization process, especially the length scale $\ell$ and the gradient noise $\sigma_g$, is investigated in section 6.2.2.

In addition two ways that could in principle improve the minimization process will be discussed in section 6.3, namely expected improvement and probability of improvement.where the main ideas are from [67]. While the first one does not require a predefined target for optimization, the second way has an adjustable parameter. The expected improvement is inherently a global search method. However, the probability of improvement, thanks to its adjustable parameter, can be adapted for global and local minimum search. In the end, both ways showed no improvement compared to pure internal 1 or internal 2.

## 6.1 Geometry Optimization Based on Gaussian process regression

Before performing an optimization, the convergence criteria for an algorithm have to be defined. Therefore, the standard criteria of DL-FIND [45] are used, which use a single value $\eta$. It is used to define values for the optimization step, the maximum step entry, the gradient norm and the maximum entry of gradient, which all have to fall below a certain value. Those criteria are in atomic units

$$\max_i \frac{\partial E}{\partial x_i} < \eta_{\text{max\_g}} := \eta \tag{6.1}$$

$$\frac{|\nabla E|}{\sqrt{d}} < \eta_{\text{norm\_g}} := \frac{2}{3}\eta \tag{6.2}$$

$$\max_i s_i < \eta_{\text{max\_s}} := 4\eta \tag{6.3}$$

$$\frac{|\mathrm{s}|}{\sqrt{d}} < \eta_{\text{norm\_s}} := \frac{8}{3}\eta, \tag{6.4}$$

where s is the step, i.e. the difference of old positions and new positions, $d$ the dimension of the system. The system is converged if all four criteria are fulfilled. In general, DL-FIND [45] has an addition energy convergence criterion. However, the energy criterion is not used for geometry optimization based on GPR.

In general, a surrogate surface for the PES is used to reduce the total number of calculations of the PES and an optimization is performed on the surrogate surface, instead of the PES. Here, the surrogate surface for the PES is constructed using Gaussian process regression. The general GPR surrogate for $M$ training points is

$$E(\mathbf{x}) = \sum_{n=1}^{M} \alpha_n k(\mathbf{x}, \mathbf{x}_n) + \sum_{n=1}^{M}\sum_{i=1}^{d} \beta_{n,i} \frac{\partial k(\mathbf{x}, \mathbf{x}_n)}{\partial x_{n,i}} + \bar{E} \tag{6.5}$$

with a constant prior mean $\bar{E}$ and $d = 3N_{\text{at}}$. Here, $N_{\text{at}}$ is the number of atoms. The gradient is a linear combination of derivatives of the covariance function:

$$\frac{\partial E(\mathbf{x})}{\partial x_k} = \sum_{n=1}^{M} \alpha_n \frac{\partial k(\mathbf{x}, \mathbf{x}_n)}{\partial x_k} + \sum_{n=1}^{M}\sum_{i=1}^{d} \beta_{n,i} \frac{\partial^2 k(\mathbf{x}, \mathbf{x}_n)}{\partial x_k \partial x_{n,i}} \tag{6.6}$$

and is used for the search algorithm to find a minimum on the surrogate surface.

The posterior mean function approaches the prior mean $\bar{E}$ from eq. (6.5) at large distances to the training points. Here the mean is chosen to be constant and much higher than the energy observations to restrict the algorithm in reasonable regions around the training points. The prior mean is, like in [15],

$$\bar{E} = \max_n E_n + 10. \tag{6.7}$$

The prior mean was optimized as hyperparameter by Raggi et al. [18] and obtained the same result as in eq. (6.7). Geometry optimization based on GPR is trained on small number of data, in contrast to general machine learning algorithms, where thousands of training points are used. Therefore, the prior mean is chosen empirically because no energy mean can be computed from the training data.

A scheme for the basic algorithm is shown in fig. 6.1. At the beginning of the optimization procedure only a single training point, the starting geometry $\mathbf{x}_1 \in \mathbb{R}^d$, $d = 3N_{\text{at}}$ in Cartesian coordinates with $N_{\text{at}}$ the number of atoms, with its energy $E_1 = E(\mathbf{x}_1)$ and gradient $\nabla_{\mathbf{x}_1} E_1$, is used. The prior mean according to eq. (6.7) is $\hat{E} = E_1 + 10$. The surrogate surface for one training point is

$$E(\mathbf{x}) = -\frac{10}{1+\sigma_e^2} k(\mathbf{x}, \mathbf{x}_1) + \frac{1}{c+\sigma_g^2} \sum_{i=1}^{d} \frac{\partial E_1}{\partial x_{1,i}} \frac{\partial k(\mathbf{x}, \mathbf{x}_1)}{\partial x_{1,i}}. + (E_1 + 10) \tag{6.8}$$

where the solution of the linear system is simple because a diagonal matrix is given. This is only valid for a isotropic covariance function, which here is the case. An isotropic covariance function only depends on the distance between the training points, i.e. $|\mathbf{x}_n - \mathbf{x}_m|$. The calculated weights are $\alpha_1 = \frac{-10}{1+\sigma_e^2}$ and $\beta_{1,i} = \frac{1}{c+\sigma_g^2} \frac{\partial E_1}{\partial x_{1,i}}$. The factor $c$ is a constant for isotropic covariance functions and depends only on $\ell$. In the case of the used Matérn covariance function $c = \frac{5}{3\ell^2}$.

The next step is now to find a minimum on this calculated surrogate surface, which is done by the standard L-BFGS [28–32] algorithm (this is called a micro-iteration in fig. 6.1). This minimum found on the surrogate surface is used as a prediction for a new geometry. With that geometry, new energy and gradient are calculated from the ab initio program, like DFT. A new surrogate surface is built, now including two training points. This iterative process is repeated until the convergence criteria are fulfilled (macro iteration).

In this last paragraph of this section a short summary of the described algorithm for geometry optimization based on GPR is given. The optimization begins with an initial guess of the structure of the molecule close to an assumed minimum. With the calculated energy $E_1$ and gradient $\nabla E_1$ as, probably noisy, observation and the starting geometry $\mathbf{x}_1$ as input a

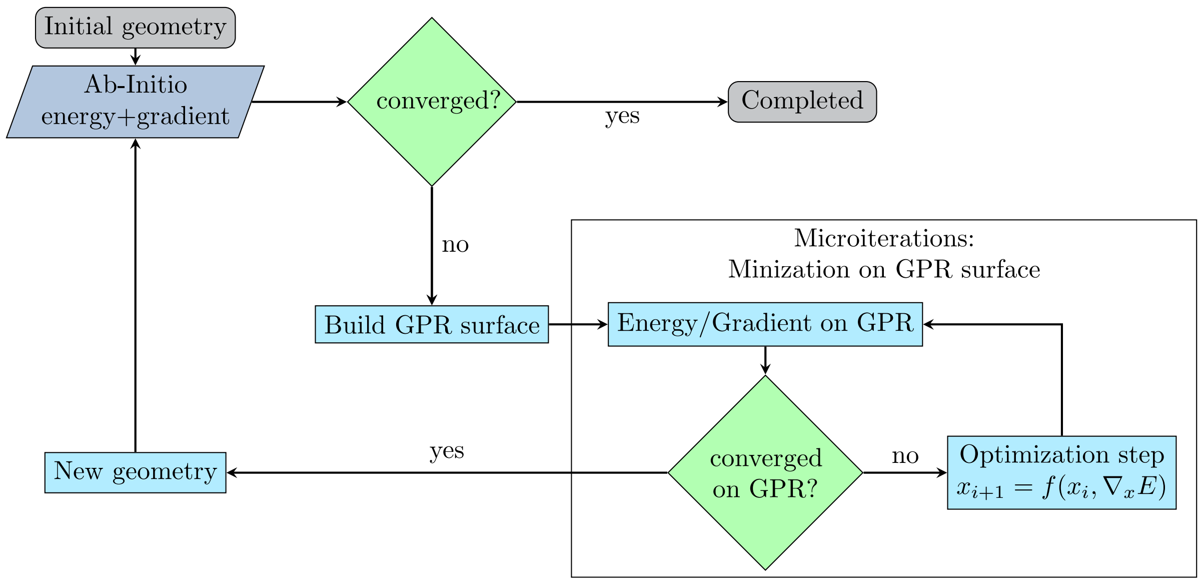

Initial geometry
Ab-Initio energy+gradient
converged?
yes
Completed
no
Microiterations: Minization on GPR surface
Build GPR surface
Energy/Gradient on GPR
yes
New geometry
converged on GPR?
no
Optimization step $x_{i+1} = f(x_i, \nabla_x E)$

**Figure 6.1:** General scheme of geometry optimization using Gaussian process regression. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

GPR-surface with a single training point is built. On the GPR-surface a minimization using L-BFGS, indicated as $f(x_i, \nabla_x E)$ in fig. 6.1 for the step on the GPR-surface, is performed and the new obtained minimum of the GPR-surface $\mathbf{x}_1^{\text{min}}$. This is called a micro iteration. For this new geometry new energy and gradient are calculated by the external ab-initio program. After each calculation convergence test is performed. The whole procedure is repeated until the convergence criteria are fulfilled.

### 6.1.1 Overshooting

To accelerate the optimization process a technique called overshooting, introduced by Denzel and Kästner [15], is used. This procedure is also applied when using internal coordinates. Therefore, overshooting is briefly explained. The angle between the last two training points and the predicted new minimum $x_n^{\text{min}}$ is defined as

$$\gamma_n = \frac{(\mathbf{x}_{n-1} - \mathbf{x}_{n-2}) \cdot (\mathbf{x}_n^{\text{min}} - \mathbf{x}_{n-1})}{|\mathbf{x}_{n-1} - \mathbf{x}_{n-2}|\,|\mathbf{x}_n^{\text{min}} - \mathbf{x}_{n-1}|}. \tag{6.9}$$

If $\gamma_n > 0.9$ then the initial step $\mathbf{s}_n = \mathbf{x}_n^{\text{min}} - \mathbf{x}_{n-1}$ is scaled up to obtain the next optimization step

$$\mathbf{s}_n^{\text{new}} = \lambda(\gamma_n)\mathbf{s}_n \tag{6.10}$$

The scaling factor is calculated as

$$\lambda(\gamma_n) = 1 + (\lambda_{\text{max}} - 1)\left(\frac{\gamma_n - 0.9}{1 - 0.9}\right)^4 . \tag{6.11}$$

The scaling factor $\lambda_{\text{max}}$ is chosen to avoid large overshooting close to the actual minimum of the PES. Then the maximum overshooting factor is limited by

$$\lambda_{\text{max}} = \big(1 + \tanh(\beta^2 - 1)\big)\,\frac{\tilde{\lambda}_{\text{max}} - 1}{2} + 1 \tag{6.12}$$

where

$$\beta = \frac{\max\limits_n(\mathbf{x}_n - \mathbf{x}_{n-1})}{\eta_{\max s}} \tag{6.13}$$

This factor describes how close the optimization is to convergence. No scaling occurs when $\beta \leq 1$. $\tilde{\lambda}_{\text{max}}$ is set to 5 at the beginning and is increased by $5\,\%$ if overshooting occurs more than once, meaning if $\gamma_n > 0.9$ twice or more.

Some dimensions converge faster than others because only one length scale $\ell$ is used instead of one length scale for each dimension. Therefore separate dimension overshooting is introduced, also by Denzel and Kästner [15], where the assumption is to treat every dimension separately. If there is a monotone change over the last 20 steps, one dimensional GPR to represent this single coordinate is built. The training points for those GPRs are the number of steps along the optimization procedure. Equidistant training points are used and the interpolation is the value of the coordinate of the respective step. The minimum or maximum of these GPRs is used as good guesses for the dimension, where the real minimum of the PES lies. Here the coupling between the coordinates is neglected, which is achievable for internal coordinates, but not for Cartesian coordinates. To counteract this for Cartesian coordinates, the separate dimension overshooting is only done every 20 steps.

Separate dimension overshooting is only applied, if the scaling would be higher than the scaling factor for overshooting and the convergence criteria for the maximum entry of the step are not fulfilled.

## 6.1.2 GPR Optimization in Internal Coordinates

In this section, two approaches to combine geometry optimization based on GPR and internal coordinates from chapter 4 is presented.

In internal 1 all training points and their gradients are transformed into internal coordinates.

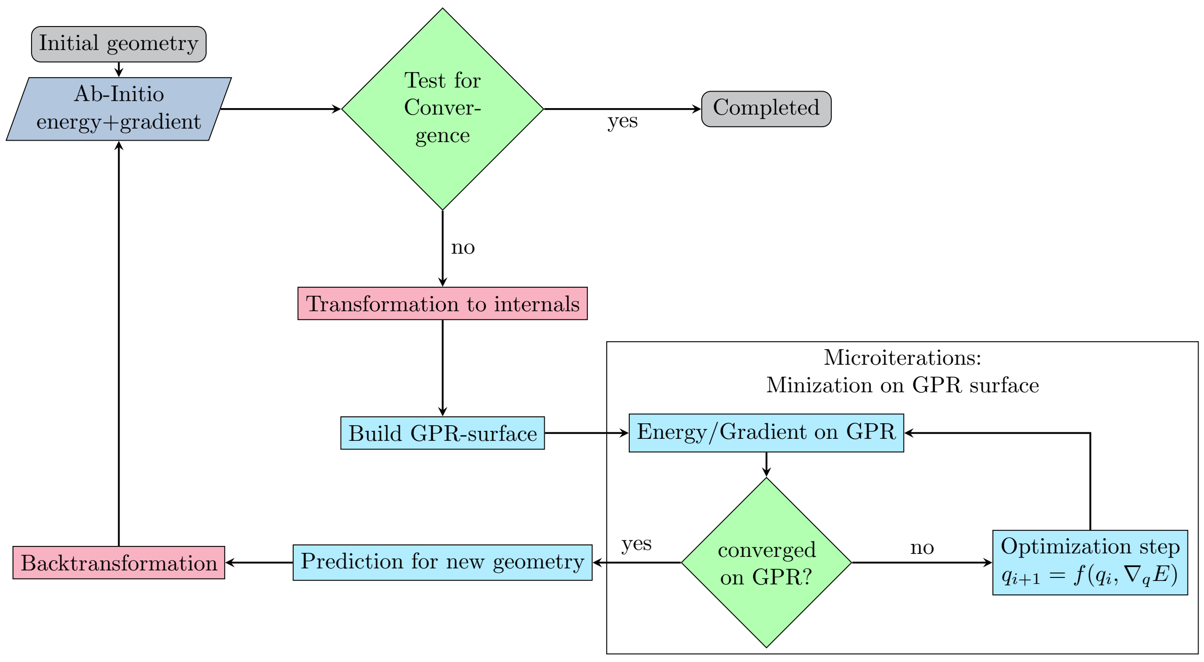

Initial geometry
Ab-Initio energy+gradient
Test for Conver-gence
yes
Completed
no
Transformation to internals
Microiterations: Minization on GPR surface
Build GPR-surface
Energy/Gradient on GPR
converged on GPR?
no
Optimization step $q_{i+1} = f(q_i, \nabla_q E)$
yes
Prediction for new geometry
Backtransformation

**Figure 6.2:** GPR-optimization in internal coordinates for method 1. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

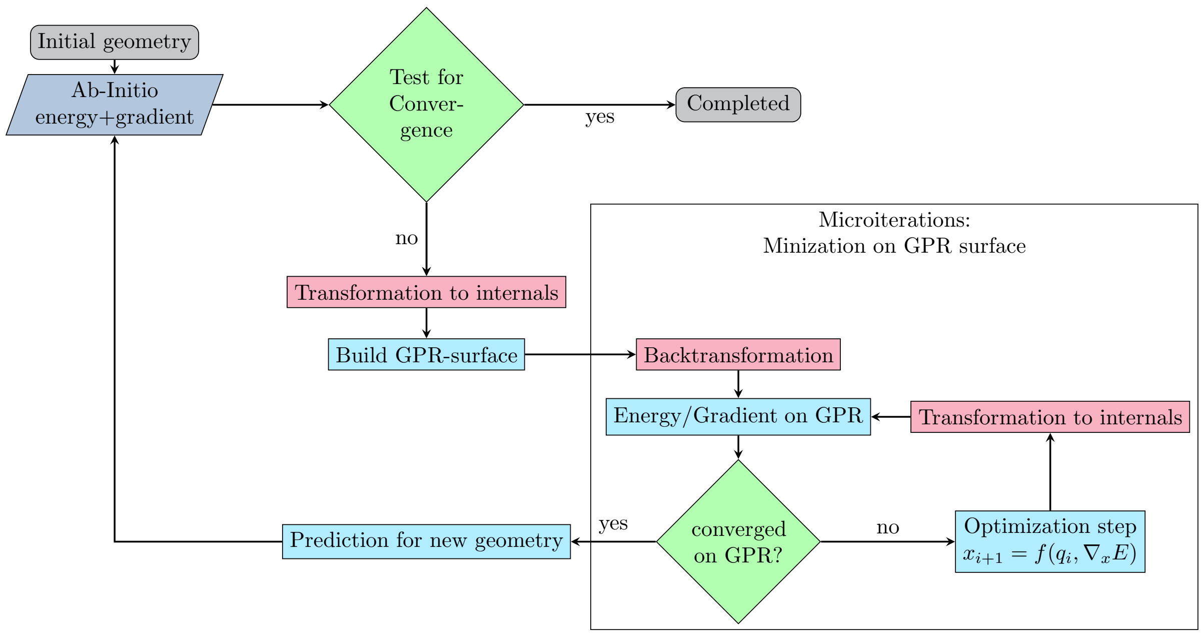

Initial geometry
Ab-Initio energy+gradient
Test for Conver-gence
yes
Completed
no
Microiterations: Minization on GPR surface
Transformation to internals
Build GPR-surface
Backtransformation
Energy/Gradient on GPR
Transformation to internals
converged on GPR?
yes
no
Optimization step $x_{i+1} = f(q_i, \nabla_x E)$
Prediction for new geometry

**Figure 6.3:** GPR-Optimization in internal coordinates for method 2. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

This means in eq. (6.5) the Cartesian coordinates $\mathbf{x}$ are replaced by the delocalized internal coordinates $\mathbf{q}$ with dimension $d = N_i = 3N_{\text{at}} - 6$. The resulting surrogate surface is then

$$E(\mathbf{q}) = \sum_{n=1}^{M} \alpha_n k(\mathbf{q}, \mathbf{q}_n) + \sum_{n=1}^{M} \sum_{i=1}^{N_i} \beta_{n,i} \frac{\partial k(\mathbf{q}, \mathbf{q}_n)}{\partial q_{n,i}} + \bar{E} \tag{6.14}$$

with the following variance

$$\mathbb{V}[\mathbf{q}] = k(\mathbf{q}, \mathbf{q}) - \sum_{n=1}^{M} \tilde{\alpha}_n k(\mathbf{q}, \mathbf{q}_n) - \sum_{n=1}^{M} \sum_{i=1}^{N_i} \tilde{\beta}_{n,i} \frac{\partial k(\mathbf{q}, \mathbf{q}_n)}{\partial q_{n,i}} \tag{6.15}$$

The weights $\tilde{\omega} = \left(\tilde{\alpha}_1, \cdots, \tilde{\alpha}_n, \tilde{\beta}_{1,1}, \cdots, \tilde{\beta}_{M,N_i}\right)^\top$ are calculated as in eq. (5.68).

The whole optimization procedure for internal 1 on the surrogate surface is done completely in internal coordinates. As a consequence, the prediction for the new geometry, where the energy and gradient are calculated, is in internal coordinates. However, the calculation of the new energy and gradient requires Cartesian coordinates. Therefore the iterative back transformation in eq. (4.24) has to be applied to obtain Cartesian coordinates corresponding to the predicted internal coordinates. The newly obtained gradient by ab initio methods is again transformed in internal coordinates and a new surrogate surface is built. This is repeated until the convergence criteria are fulfilled. If the back transformation fails, which happens when the $\mathbf{G}$-matrix, introduced in section 4.3, becomes singular, a new set of internal coordinates is constructed. The previously constructed surrogate surface is then used as new prior mean $\bar{E} = E_{\text{old}}(\mathbf{q})$. This case did not appear in the tested systems but is a general backup if a back-transformation failure occurs. A flow chart of this algorithm is shown in fig. 6.2.

The overshooting procedure from section 6.1.1 is used for internal 1. Here the angle is calculated using the internal coordinates.

For internal 2 only the coordinates are transformed into internal coordinates, while the gradient remains unchanged in Cartesian coordinates. The evaluation of the covariance function is done in internal coordinates, the derivatives of the covariance function are "transformed" into Cartesian, which simply means applying the chain rule. Therefore, the entries for the covariance matrix $\mathbf{K}$ are

$$k(\mathbf{q}(\mathbf{x}_n), \mathbf{q}(\mathbf{x}_m)) = k(\mathbf{q}_n, \mathbf{q}_m) \tag{6.16}$$

$$\frac{\partial k(\mathbf{q}(\mathbf{x}_n), \mathbf{q}(\mathbf{x}_m))}{\partial x_{n,k}} = \sum_{i=1}^{N_i} \frac{\partial k(\mathbf{q}_n, \mathbf{q}_m)}{\partial q_{n,i}} \frac{\partial q_{n,i}}{\partial x_{n,k}} \tag{6.17}$$

and similar for derivatives with respect to training point $m$. The entry for the second derivatives (which are needed for including derivative information) is

$$\frac{\partial^2 k(\mathbf{q}(\mathbf{x}_n), \mathbf{q}(\mathbf{x}_m))}{\partial x_{n,k} \partial x_{m,l}} = \sum_{i=1}^{N_i} \sum_{j=1}^{N_i} \frac{\partial q_{n,i}}{\partial x_{n,k}} \frac{\partial^2 k(\mathbf{q}_n, \mathbf{q}_m)}{\partial q_{n,i} \partial q_{m,j}} \frac{\partial q_{m,j}}{\partial x_{m,l}} + \underbrace{\sum_{i=1}^{N_i} \frac{\partial^2 q_{n,i}}{\partial x_{n,k} \partial x_{m,l}} \frac{\partial k(\mathbf{q}_n, \mathbf{q}_m)}{\partial q_{n,i}}}_{=0} \tag{6.18}$$

The last term cancels because, for different training points, the second derivative of the internal coordinates with respect to different Cartesian coordinates is zero due to the independence of the training points. Since an isotropic covariance function is used, the first derivative is zero if $\mathbf{q}_n = \mathbf{q}_m$ (the only term the derivative of the $\mathbf{B}$-matrix is not zero). The derivatives of the covariance function can also be written in matrix form

$$\nabla_{\mathbf{x}_n} k(\mathbf{q}(\mathbf{x}_n), \mathbf{q}(\mathbf{x}_m)) = \mathbf{U}^\top \mathbf{B}_n \nabla_{\mathbf{q}_n} k(\mathbf{q}_n, \mathbf{q}_m) \tag{6.19}$$

$$\nabla^2_{\mathbf{x}_n} k(\mathbf{q}(\mathbf{x}_n), \mathbf{q}(\mathbf{x}_m)) = \mathbf{U}^\top \mathbf{B}_n \nabla^2_{\mathbf{q}} k(\mathbf{q}_n, \mathbf{q}_m) \mathbf{B}_m^\top \mathbf{U} \tag{6.20}$$

Using these definitions the surrogate surface for the PES is

$$E(\mathbf{q}(\mathbf{x})) = \sum_{n=1}^{M} \alpha_n k(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}_n)) + \sum_{n=1}^{M} \sum_{i=1}^{N_x} \beta_{n,i} \sum_{j=1}^{N_i} \frac{\partial k(\mathbf{q}, \mathbf{q}_n)}{\partial q_{n,j}} \frac{\partial q_{n,j}}{\partial x_{n,i}} + \bar{E} \tag{6.21}$$

and its gradient can be calculated as

$$\frac{\partial E(\mathbf{q}(\mathbf{x}))}{\partial x_k} = \sum_{n=1}^{M} \alpha_n \sum_{i=1}^{N_i} \frac{\partial k(\mathbf{q}, \mathbf{q}_n)}{\partial q_i} \frac{\partial q_i}{\partial x_k} + \sum_{n=1}^{M} \sum_{i=1}^{N_x} \beta_{n,i} \sum_{j=1}^{N_i} \sum_{l=1}^{N_i} \frac{\partial q_j}{\partial x_k} \frac{\partial^2 k(\mathbf{q}, \mathbf{q}_n)}{\partial q_j \partial q_{n,l}} \frac{\partial q_{n,l}}{\partial x_{n,i}} \tag{6.22}$$

where $N_x = 3N_{\text{at}}$. The basis of the calculated covariance matrix is necessarily incomplete due to the transformation from a $3N_{\text{at}} - 6$ to $3N_{\text{at}}$ dimensional space. The remaining six gradient components are zero and therefore the whole covariance matrix becomes singular. This means that a necessary large gradient noise $\sigma_g$ has to be added to the diagonal elements to allow the Cholesky decomposition to work properly. The variance $\mathbb{V}$ also has to be transformed properly

$$\mathbb{V}[\mathbf{q}(\mathbf{x})] = k(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x})) - \sum_{n=1}^{M} \tilde{\alpha}_n k(\mathbf{q}, \mathbf{q}_n) - \sum_{n=1}^{M} \sum_{i=1}^{N_x} \tilde{\beta}_{n,i} \sum_{j=1}^{N_i} \frac{\partial k(\mathbf{q}, \mathbf{q}_n)}{\partial q_{n,j}} \frac{\partial q_{n,j}}{\partial x_{n,i}} \tag{6.23}$$

where the vector $\mathbf{k}$ from eq. (5.69) also requires a transformation

$$\mathbf{k} = \left( k(\mathbf{q},\mathbf{q}_1), \cdots, k(\mathbf{q},\mathbf{q}_M), \sum_{i=1}^{N_i} \frac{\partial k(\mathbf{q},\mathbf{q}_1)}{\partial q_{1,i}} \frac{\partial q_{1,i}}{\partial x_{1,1}}, \cdots, \sum_{i=1}^{N_i} \frac{\partial k(\mathbf{q},\mathbf{q}_M)}{\partial q_{M,i}} \frac{\partial q_{M,i}}{\partial x_{M,N_x}} \right)^{\top} . \tag{6.24}$$

The micro iterations are performed in Cartesian coordinates and after each step, the new Cartesian coordinates have to be transformed into internal coordinates. After the convergence criteria for the geometry optimization on the surrogate surface are fulfilled the predicted geometry for the macro iteration is in Cartesian coordinates. Therefore, no iterative back transformation has to be applied. A flow chart for the algorithm of internal 2 is shown in fig. 6.3.

The overshooting procedure described in section 6.1.1 is applied for internal 2 too. However, in contrast to internal 1, the angle is calculated using Cartesian coordinates, because the prediction for the new geometry is also in Cartesian coordinates.

The convergence critera for the optimization on the GPR surrogate is, if the norm of the gradient evaluated on the GPR is smaller than $10^{-7}\, E_\mathrm{h}/a_0$ or the step length is smaller than $10^{-7}\, a_0$ or a maximum number of 1000 steps is reached. These criteria are used for both methods. The used Matérn covariance function, see eq. (5.63), has a scaling factor $\sigma_f^2$, which is set to one for GPR internal 1 and internal 2. For both methods the step length on the GPR surface during the micro iteration is restricted only, if $\mathbb{V}[\mathbf{q}] > 10^{-4}$. In that case, the step is restricted to a maximum of $10^{-2}\, a_0$. To prevent a too large step on the real PES due to large overshooting, a maximum step length as an adjustable parameter is used. In this case, the maximum step is restricted to $0.5\, a_0$.

**Table 6.1:** Statistical data for the box plot in fig. 6.6 with different optimization coordinate combinations. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

| Optimizer | L-BFGS | | GPR | | |
|---|---|---|---|---|---|
| Coordinate System | Cartesian | internal | Cartesian | internal 1 | internal 2 |
| first quartile ($Q_1$) | 11.0 | 7.5 | 9.0 | 7.0 | 7.5 |
| median | 17.0 | 10.0 | 15.0 | 9.0 | 9.0 |
| third quartile ($Q_3$) | 38.5 | 23.0 | 33.5 | 21.5 | 24.0 |
| mean | 32.8 | 20.3 | 26.7 | 16.4 | 19.3 |

## 6.2 Results

The geometry optimization algorithms were tested on a test set of 30 small molecules and a large rhodium complex. All energy and gradients were calculated on DFT level with BP86 [68] [69] as functional and def2-SVP [70] as a basis set. These calculations were done in TURBOMOLE [71] with ChemShell [72] as the interface to DL-FIND [45]. The convergence criteria in DL-FIND were set to the default values, meaning $\eta$ from eq. (6.2) is set to $4.5 \cdot 10^{-4}\,E_\mathrm{h}/a_0$. Then the criteria are $4.5 \cdot 10^{-4}\,E_\mathrm{h}/a_0$ for the change of the largest gradient component, $3 \cdot 10^{-4}\,E_\mathrm{h}/a_0$ for the change of the root mean square of the gradient. The maximum step was set to $1.8 \cdot 10^{-3}\,a_0$ and the root mean square step length was set to $1.2 \cdot 10^{-3}\,a_0$.

### 6.2.1 Efficiency of The Optimizer

The molecules used for the tests are shown in fig. 6.4 and fig. 6.5. The rhodium complex was originally investigated by Kirchhof et al. [73]. The distribution of the number of steps that different optimization algorithms rquire to reach convergence is shown in fig. 6.6. The GPR optimizer is compared to the standard optimizer of DL-FIND, the L-BFGS optimizer. Additionally, Cartesian coordinates are compared to internal coordinates. They are compared by their required number of steps until convergence. Each of those steps needs one calculation of energy and gradient from the external program. The respective number of steps can be found in table 6.2 and fig. 6.7 as a visualization. The box plot from fig. 6.6 is used to illustrate the distribution of the number of steps. In that plot, the red bar represents the median, which separates $50\,\%$ of the number of steps above the median, $50\,\%$ are below. It cuts the data set in half. Therefore it is also called the second quartile ($Q_2$). The lower black bar is the first quartile $Q_1$, which splits the $25\,\%$ of the lowest data from the $75\,\%$ highest values. The upper

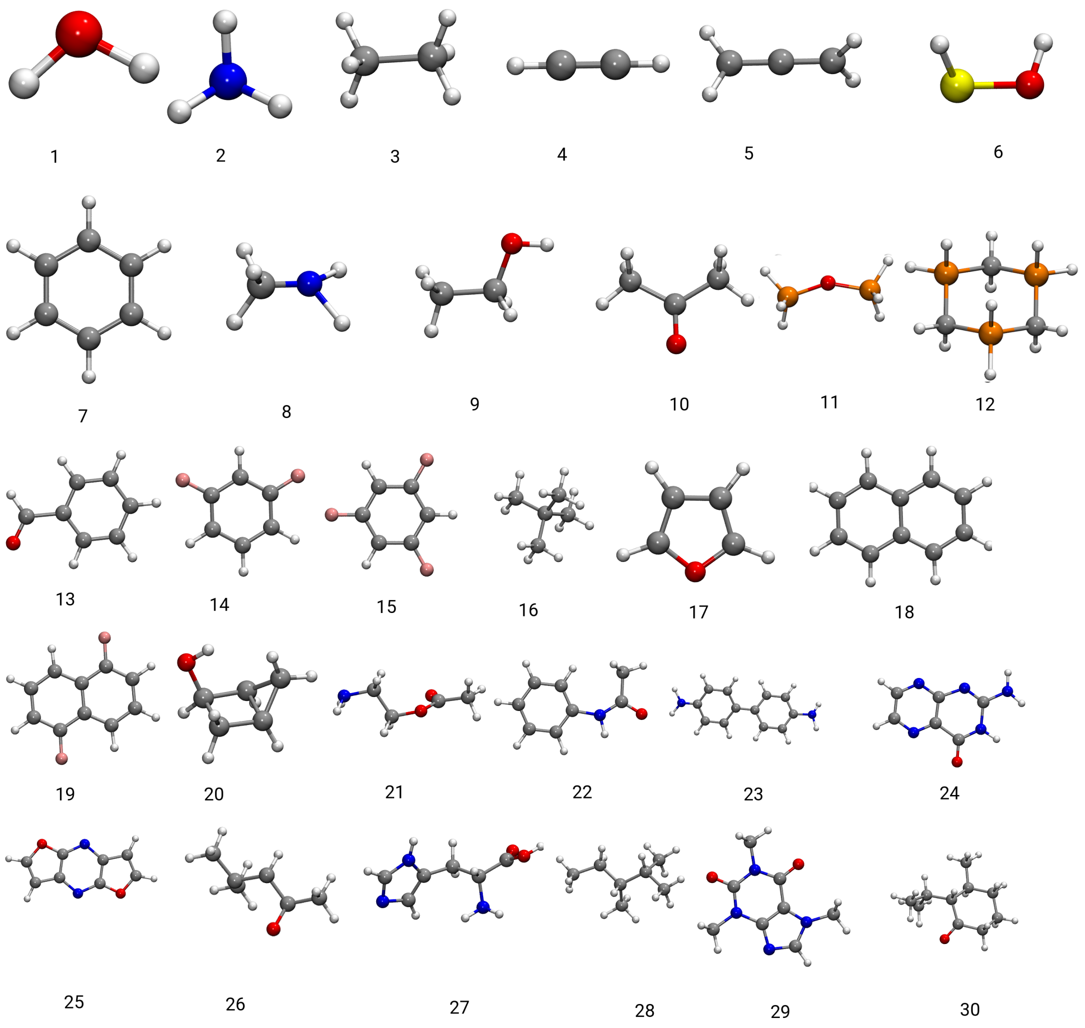


**Figure 6.4:** 30 molecules used as the test set. The color code is: Oxygen red, nitrogen blue, carbon gray, sulfur yellow, fluorine light red, silicon orange and hydrogen white. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

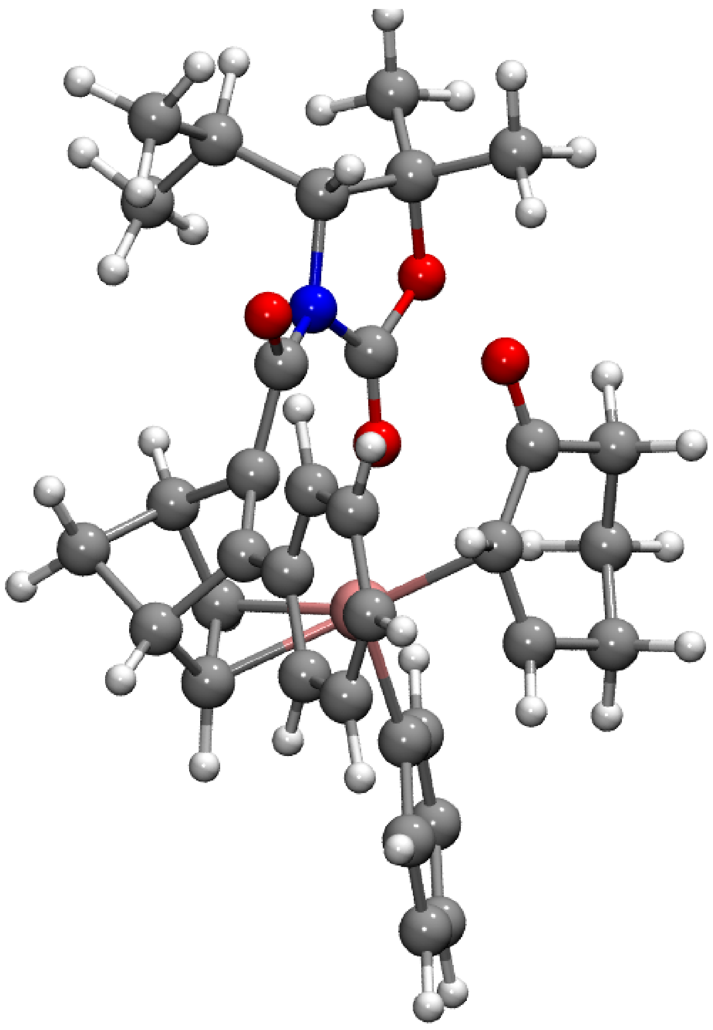


**Figure 6.5:** Rhodium complex as a large test molecule. The rhodium atom in the center is in light red, the color code for the other atoms is the same as in fig. 6.4. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

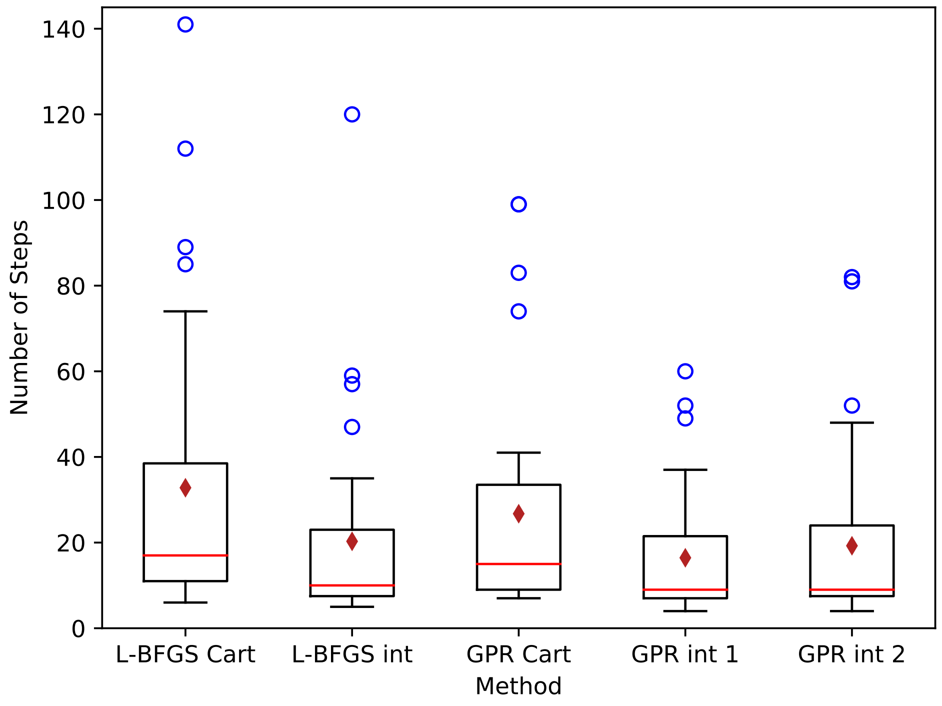


**Figure 6.6:** Performance of L-BFGS and GPR in Cartesian (Cart) and internal coordinates (int). Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

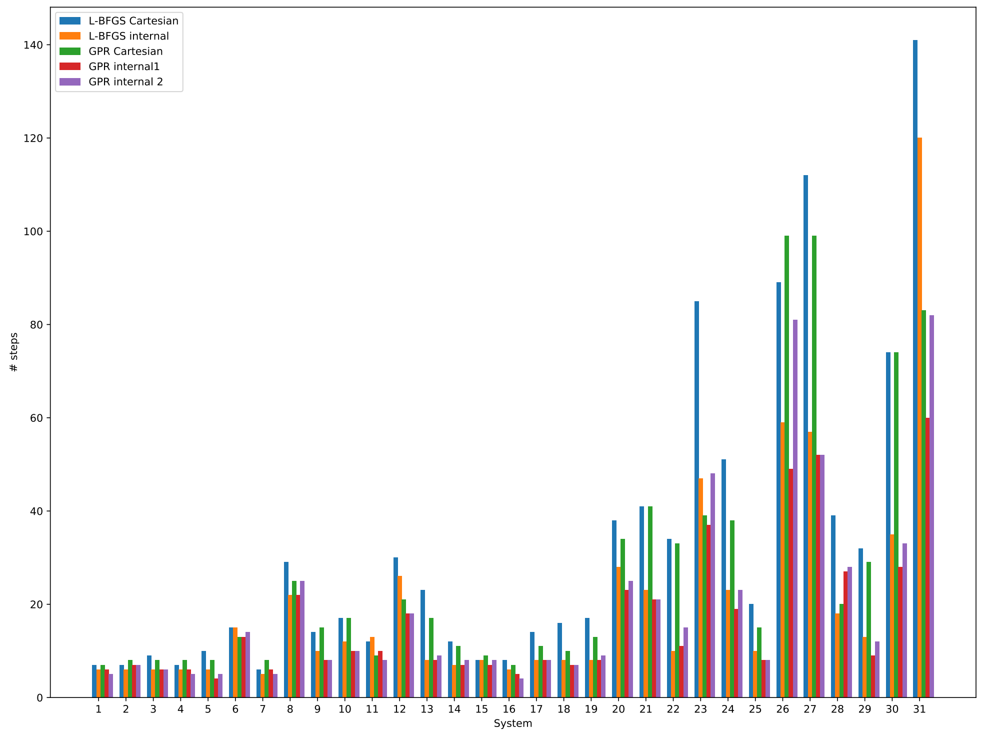


**Figure 6.7:** Bar plot of results with the total amount of steps required for convergence. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

bar is the third quartile $Q_3$, meaning $75\,\%$ of the data lies below this point. The interquartile range IQR $= Q_3 - Q_1$ is used to indicate outliers. An outlier above the third quartile is identified if the number of steps is above $Q_3 + 1.5$IQR. Similarly, there are no outliers for the low number of steps. The outliers are represented as blue circles. The (arithmetic) mean is the red diamond. The upper whisker is the highest black line and is $1.5$ times the IQR above the third quartile. The lowest black line is the lower whisker and is defined as 1.5 times below the first quartile. However, since there are no outliers for low number of steps, it is the lowest number of steps. The values for the quartiles and the mean are given in table 6.1.

The efficiency of both optimizers, GPR and L-BFGS, is increased if internal coordinates are used instead of Cartesian coordinates. The median for L-BFGS internal coordinates is reduced from 17 to 10, which is $41\,\%$, while the mean is reduced by $38\,\%$, from 32.8 to 20.3. The median is more stable towards outliers and is therefore used for comparison. The first quartile is reduced by $32\,\%$, meaning from 11 to 7.5, and the third quartile is reduced from 38.5 to 23, or $40\,\%$. In addition, the total amount of steps for all systems is reduced from 1017 in Cartesian coordinates to 629, this is also $38\,\%$. For the GPR optimizer internal coordinates reduce the median number of steps from 15 to 9, or $40\,\%$. The first quartile is for both internal methods reduced by $22\,\%$ or from 9 to 7 for internal 1 and 7.5 for internal 2,respectively. The mean is reduced from 26.7 to 16.4 for GPR internal 1 and to 19.3 for GPR internal 2. That is by $39\,\%$ and $28\,\%$,respectively. The third quartile is reduced by $36\,\%$ for internal 1 from 33.5 to 21.5, and for internal 2 it is reduced by $28\,\%$ down to 24. Comparing the total number of steps, GPR internal 1 reduces the required steps from 829 to 510, that is $38\,\%$, while GPR internal 2 reduces the steps to 597 or by $28\,\%$. Comparing GPR internal 1 with internal 2, the first quartile is reduced by $7\,\%$, the median is the same, and the third quartile is reduced by $10\,\%$. The mean and the total number of steps are reduced by $15\,\%$, if using GPR internal 1 compared to GPR internal 2.

Having no differences between the first and second quartiles, GPR internal 1 and 2 perform equally well for small molecules. The difference however increases for the total number of steps, the third quartile, and the mean number of steps, which indicates that GPR internal 1 performs, in general, better for larger molecules. This can also be seen in the differences in each statistical value comparing GPR internal 1 and GPR internal 2. Consequently, molecules that require few steps to converge, which are usually small molecules, internal 1 and internal 2 perform similarly. This can also be seen that the first quartile is almost the same (7 vs. 7.5) and the median is the same. Differences arise for larger molecules as can be seen in the third quartile and the mean.

Comparing the median between all methods, GPR in internal coordinates performs best compared to all the other combinations of optimizers and coordinates, especially for large molecules. The largest molecule is the rhodium complex, which consists of 78 Atoms. This means a 234-dimensional PES in Cartesian coordinates and a 228-dimensional PES has to be optimized in internal coordinates. For all combinations of optimizer and coordinates, this results in the highest point in fig. 6.6. L-BFGS in Cartesian coordinates requires 141 energy and gradient evaluations, or steps. In internal coordinates, the L-BFGS optimizer still requires 120 steps. Only the GPR optimizer can significantly reduce the number of required steps. In Cartesian coordinates, 83 steps are needed. Internal 1 requires the least amount of steps, namely 60 steps, while internal 2 still requires 82 steps. Thus GPR internal 1 performs best for large molecules. Concluding GPR internal 1 is the most efficient method.

In some test cases, there are huge differences between internal 1 and internal 2. The system

**Table 6.2:** Number of steps for each system shown in fig. 6.4 and the rhodium complex from fig. 6.5. Additionally the total amount of steps was calculated. The energy difference to L-BFGS in Cartesian coordinates is shown in parentheses and are given in Hartree. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society

| Optimizer | L-BFGS | | GPR | | |
|---|---|---|---|---|---|
| Coordinate System | Cartesian | internal | Cartesian | internal 1 | internal 2 |
| 1 Water | 7 | 6 ($2.3 \cdot 10^{-9}$) | 7 ($1.5 \cdot 10^{-8}$) | 6 ($2.1 \cdot 10^{-9}$) | 5 ($3.6 \cdot 10^{-9}$) |
| 2 Ammonia | 7 | 6 ($7.2 \cdot 10^{-8}$) | 8 ($7.1 \cdot 10^{-8}$) | 7 ($7.5 \cdot 10^{-8}$) | 7 ($7.5 \cdot 10^{-8}$) |
| 3 Ethane | 9 | 6 ($3.3 \cdot 10^{-8}$) | 8 ($6.4 \cdot 10^{-8}$) | 6 ($2.6 \cdot 10^{-9}$) | 6 ($2.7 \cdot 10^{-9}$) |
| 4 Acetylene | 7 | 6 ($5.7 \cdot 10^{-10}$) | 8 ($4.4 \cdot 10^{-9}$) | 6 ($5.4 \cdot 10^{-10}$) | 5 ($4.8 \cdot 10^{-10}$) |
| 5 Allene | 10 | 6 ($6.3 \cdot 10^{-9}$) | 8 ($1.8 \cdot 10^{-8}$) | 4 ($1.1 \cdot 10^{-7}$) | 5 ($1.6 \cdot 10^{-8}$) |
| 6 Hydroxysulfane | 15 | 15 ($1.2 \cdot 10^{-7}$) | 13 ($4.5 \cdot 10^{-7}$) | 13 ($1.7 \cdot 10^{-7}$) | 14 ($6.9 \cdot 10^{-8}$) |
| 7 Benzene | 6 | 5 ($1.1 \cdot 10^{-9}$) | 8 ($1.9 \cdot 10^{-7}$) | 6 ($1.1 \cdot 10^{-9}$) | 5 ($1.1 \cdot 10^{-9}$) |
| 8 Methylamine | 29 | 22 ($3.1 \cdot 10^{-7}$) | 25 ($3.0 \cdot 10^{-7}$) | 22 ($3.4 \cdot 10^{-7}$) | 25 ($3.6 \cdot 10^{-7}$) |
| 9 Ethanol | 14 | 10 ($3.6 \cdot 10^{-7}$) | 15 ($2.3 \cdot 10^{-7}$) | 8 ($2.4 \cdot 10^{-7}$) | 8 ($2.5 \cdot 10^{-7}$) |
| 10 Acetone | 17 | 12 ($5.1 \cdot 10^{-8}$) | 17 ($3.1 \cdot 10^{-9}$) | 10 ($1.4 \cdot 10^{-8}$) | 10 ($1.5 \cdot 10^{-8}$) |
| 11 Disilyl-ether | 12 | 13 ($5.7 \cdot 10^{-6}$) | 9 ($3.1 \cdot 10^{-7}$) | 10 ($5.7 \cdot 10^{-6}$) | 8 ($5.6 \cdot 10^{-6}$) |
| 12 1,3,5-Trisilacyyclohexane | 30 | 26 ($3.5 \cdot 10^{-8}$) | 21 ($1.0 \cdot 10^{-5}$) | 18 ($1.1 \cdot 10^{-7}$) | 18 ($1.3 \cdot 10^{-7}$) |
| 13 Benzaldehyde | 23 | 8 ($7.6 \cdot 10^{-7}$) | 17 ($8.6 \cdot 10^{-6}$) | 8 ($7.6 \cdot 10^{-7}$) | 9 ($6.9 \cdot 10^{-7}$) |
| 14 1,3-Difluorobenzene | 12 | 7 ($8.7 \cdot 10^{-7}$) | 11 ($6.2 \cdot 10^{-7}$) | 7 ($2.8 \cdot 10^{-7}$) | 8 ($2.9 \cdot 10^{-8}$) |
| 15 1,3,5-Trifluorobenzene | 8 | 8 ($4.9 \cdot 10^{-9}$) | 9 ($4.2 \cdot 10^{-9}$) | 7 ($5.3 \cdot 10^{-8}$) | 8 ($1.6 \cdot 10^{-7}$) |
| 16 Neopentane | 8 | 6 ($9.1 \cdot 10^{-8}$) | 7 ($4.6 \cdot 10^{-8}$) | 5 ($8.7 \cdot 10^{-7}$) | 4 ($8.7 \cdot 10^{-7}$) |
| 17 Furan | 14 | 8 ($8.4 \cdot 10^{-7}$) | 11 ($2.1 \cdot 10^{-6}$) | 8 ($2.5 \cdot 10^{-7}$) | 8 ($2.5 \cdot 10^{-7}$) |
| 18 Napthalene | 16 | 8 ($2.5 \cdot 10^{-7}$) | 10 ($6.5 \cdot 10^{-7}$) | 7 ($9.6 \cdot 10^{-8}$) | 7 ($1.4 \cdot 10^{-7}$) |
| 19 1,5-Difluoronaphtalene | 17 | 8 ($1.3 \cdot 10^{-7}$) | 13 ($1.2 \cdot 10^{-6}$) | 8 ($4.9 \cdot 10^{-6}$) | 9 ($2.2 \cdot 10^{-7}$) |
| 20 2-Hydroxybicyclopentane | 38 | 28 ($8.6 \cdot 10^{-7}$) | 34 ($9.4 \cdot 10^{-8}$) | 23 ($2.3 \cdot 10^{-6}$) | 25 ($2.9 \cdot 10^{-6}$) |
| 21 ACHTAR10 | 41 | 23 ($2.5 \cdot 10^{-6}$) | 41 ($1.7 \cdot 10^{-7}$) | 21 ($2.9 \cdot 10^{-6}$) | 21 ($5.7 \cdot 10^{-7}$) |
| 22 ACANIL01 | 34 | 10 ($3.4 \cdot 10^{-7}$) | 33 ($2.6 \cdot 10^{-7}$) | 11 ($2.5 \cdot 10^{-7}$) | 15 ($2.5 \cdot 10^{-7}$) |
| 23 Benzidine | 85 | 47 ($3.5 \cdot 10^{-7}$) | 39 ($1.2 \cdot 10^{-4}$) | 37 ($9.6 \cdot 10^{-8}$) | 48 ($1.4 \cdot 10^{-7}$) |
| 24 Pterin | 51 | 23 ($3.3 \cdot 10^{-6}$) | 38 ($2.6 \cdot 10^{-6}$) | 19 ($4.9 \cdot 10^{-6}$) | 23 ($5.2 \cdot 10^{-6}$) |
| 25 Difuropyrazine | 20 | 10 ($6.2 \cdot 10^{-8}$) | 15 ($2.0 \cdot 10^{-6}$) | 8 ($6.1 \cdot 10^{-8}$) | 8 ($7.6 \cdot 10^{-8}$) |
| 26 Mesityl-oxide | 89 | 59 ($7.7 \cdot 10^{-4}$) | 99 ($3.3 \cdot 10^{-5}$) | 49 ($3.8 \cdot 10^{-6}$) | 81 ($1.1 \cdot 10^{-6}$) |
| 27 Histidine | 112 | 57 ($6.9 \cdot 10^{-6}$) | 99 ($3.3 \cdot 10^{-5}$) | 52 ($8.9 \cdot 10^{-6}$) | 52 ($8.6 \cdot 10^{-6}$) |
| 28 Dimethylpentane | 39 | 18 ($1.3 \cdot 10^{-5}$) | 20 ($3.3 \cdot 10^{-5}$) | 27 ($2.5 \cdot 10^{-5}$) | 28 ($2.5 \cdot 10^{-5}$) |
| 29 Caffeine | 32 | 13 ($3.5 \cdot 10^{-7}$) | 29 ($9.2 \cdot 10^{-7}$) | 9 ($3.4 \cdot 10^{-7}$) | 12 ($3.7 \cdot 10^{-7}$) |
| 30 Menthone | 74 | 35 ($4.5 \cdot 10^{-7}$) | 74 ($2.4 \cdot 10^{-7}$) | 28 ($8.7 \cdot 10^{-7}$) | 33 ($2.7 \cdot 10^{-6}$) |
| 31 Rhodium complex | 141 | 120 ($7.2 \cdot 10^{-5}$) | 83 ($2.4 \cdot 10^{-4}$) | 60 ($2.3 \cdot 10^{-5}$) | 82 ($4.0 \cdot 10^{-5}$) |
| total number of steps | 1017 | 629 | 829 | 510 | 597 |

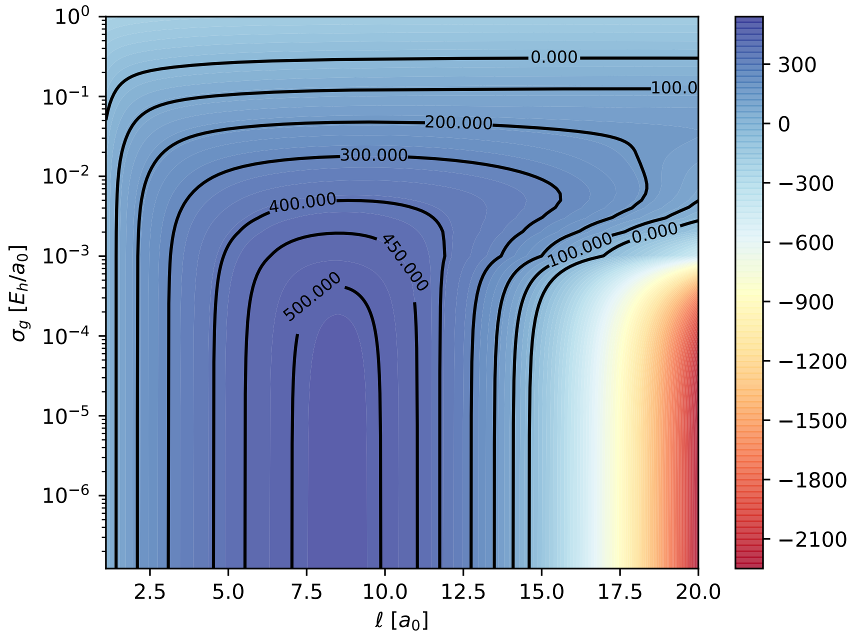
with the most difference is system 26, where internal 1 requires 49 steps, while internal 2 requires almost the double amount of steps, namely 81 steps. Also in systems 23 and 31 a difference can be observed, however not as significant. The reason lies in the overshooting procedure from section 6.1.1. The overshooting uses the angle of the latest three training points. Since the GPR surface for internal 2 is in Cartesian coordinates, in contrast to internal 1, which is in internal coordinates, at different steps during the optimization overshooting is performed. This can lead, like in those three cases, to a different number of steps.

In addition, the energy difference for all methods is compared to L-BFGS in Cartesian coordinates to indicate, if the same minimum is found. The respective results are shown in table 6.2. The difference is for most systems equal or below $10^{-6}\,E_\mathrm{h}$. For two systems only an energy difference of $10^{-4}\,E_\mathrm{h}$ has been found. The first one is system 26, mesityl-oxide for L-BFGS in internal coordinates. This system did not converge to a minimum, instead, it converged to a first-order saddle point. The other system is benzidine (23) for the GPR in Cartesian coordinates, which converged to a slightly different minimum. In all other cases, the same minimum was found for all methods.

In principle, a comparison to Meyer and Hauser [74] is possible for internal 2, because they also investigated this method for the 30 molecules in fig. 6.4. In contrast to this work, they used Hartree-Fock with STO-3G basis set as the respective level of theory. In this work DFT was used. They also used the same starting geometry like in this work, except for systems 8, 23, 24, and 26. These were here slightly distorted because the optimizer found always a saddle point due to symmetry reasons. If we compare all but these four molecules, they require 395 steps for delocalized internal coordinates. Internal 2 in this work required fewer steps, i.e. 338. This difference can result from a different set of primitive internal coordinates, a different covariance function (they used the squared exponential), or a different ab-initio method.

### 6.2.2 Influence of The Hyperparameters

The performance of the GPR optimizer is potentially influenced by two hyperparameters, the length scale $\ell$ and the noise parameter $\sigma$. One way to choose them is to optimize the marginal likelihood in eq. (5.72) Since the optimization of $\log p$, and its derivative is computationally quite expensive, a different approach is chosen, which will be explained in the following. In fig. 6.8 the dependence of $\log p$ on $\ell$ and $\sigma_g$ is plotted, for both methods internal 1 and internal 2 for the furan system. fig. 6.8 on the top shows the dependence for internal 1, while the dependence for internal 2 is shown in fig. 6.8 on the bottom. In these plots, six training points are included. The hyperparameters are quite decoupled and treating them separately is



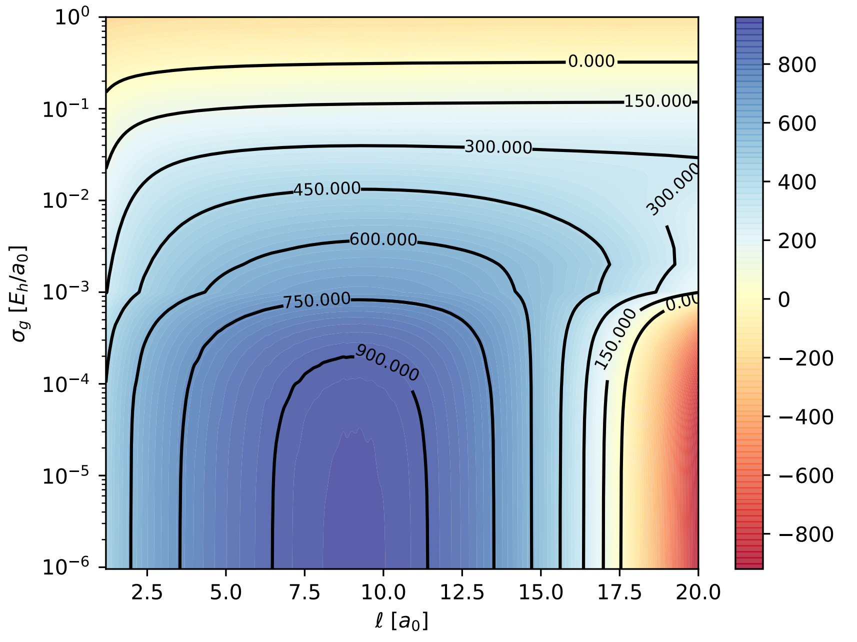


**Figure 6.8:** Heat maps of the marginal likelihood $\log\, p$ in dependence of the length scale $\ell$ and the gradient noise $\sigma_g$, internal 1 on the top and internal 2 on the bottom. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

justified.

First, the dependence of the log marginal likelihood $\log\, p$ on the length scale $\ell$ is discussed. Therefore it is plotted in fig. 6.11 for the molecule ethanol. In fig. 6.11 it is shown on the top for internal 1 and in fig. 6.11 on the bottom for internal 2. Looking at the plots the peak for internal 1 is more peaked compared to internal 2. The maximum for internal 1 is at $\ell = 8.5\, a_0$, while for internal 2, a shallow maximum at $\ell = 10.5\, a_0$ can be observed. This fact results in a lesser dependence on the length scale for internal 2. Other systems show similar results. For example furan and the largest system, the rhodium complex, can be seen in fig. 6.10 and fig. 6.9. However, the maximum is at larger values of $\ell$. The maximum value of $\ell$ is between $10.5\, a_0$ and $13\, a_0$. Including more training data, i.e. with an increasing number of steps, the length scale with the highest $\log\, p$ becomes smaller. The reason behind this is, that with more training data at more similar geometries the training points are more correlated, resulting in a lower length scale. This is used to get an appropriate length scale without an expensive optimization of $\log\, p$. In the beginning a fixed length scale of $\ell = 13\, a_0$ is chosen. Every time during the optimization $\ell$ is reduced by approximately $5\,\%$ if the norm of the gradient of the previous ab-initio calculation is lower than the current one. Particularly, $1/\ell^2$ is increased by $10\,\%$, which is a decrease of $\ell$ by $4.65\,\%$. This approach decreases the total number of steps. For example, system 31 requires 143 steps if a fixed length scale of $\ell = 13\, a_0$ is used. With the automatically adapted length scale, only 60 steps are required. The proposed approach leads to a too large length scale for small systems like water. However, they anyway need a small number of optimization steps. Since the goal of the optimization algorithms are large, realistic molecules, like system 31, this approach works very well without choosing every time the length scale manually.

An alternative way to optimize the length scale is to calculate values for $\log\, p$ at three different values of $\ell$ and find a maximum from a quadratic or a GPR fit, without calculating the expensive derivative of $\log\, p$. In fig. 6.12 an example of this method is demonstrated. As can be seen, the GPR fit fits quite well the true $\log\, p$ function around the area of the maximum function value. This method for computing the optimal hyperparameters can be applied for example for a system that is much further away from a minimum than in our tests. This could also be used for molecules where the maximum is not as flat as in the test systems investigated here.

The dimension of the energy and the gradient are different, i.e. the energy is measured in Hartree $E_\mathrm{h}$, the gradient as energy per length, $E_\mathrm{h}/a_0$. Therefore the separation of the noise parameter $\sigma$ can be split into one for the energy $\sigma_e$ and one for the gradient $\sigma_g$. Then the

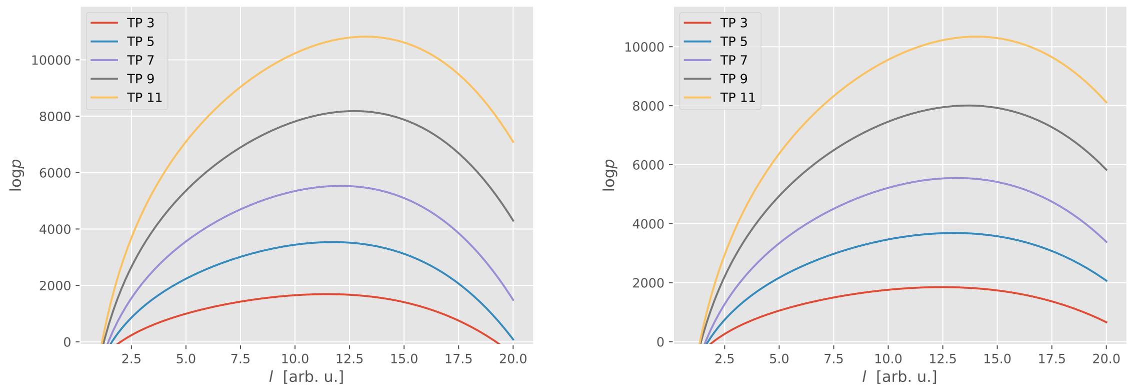


**Figure 6.9:** The logarithm of the marginal likelihood $\log p$ depended on the lengthscale for the rhodium complex for internal 1(left) and internal 2 (right). The number of training points, TP in the plot, refers to the number of steps.

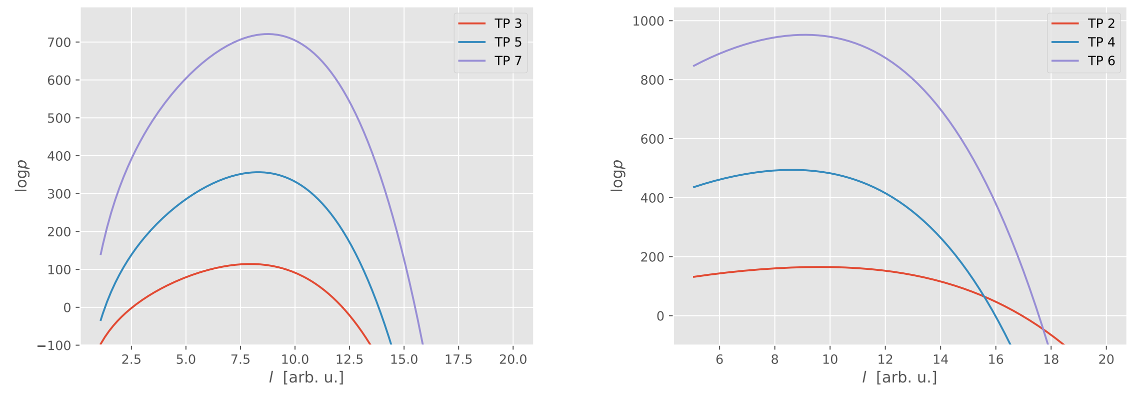
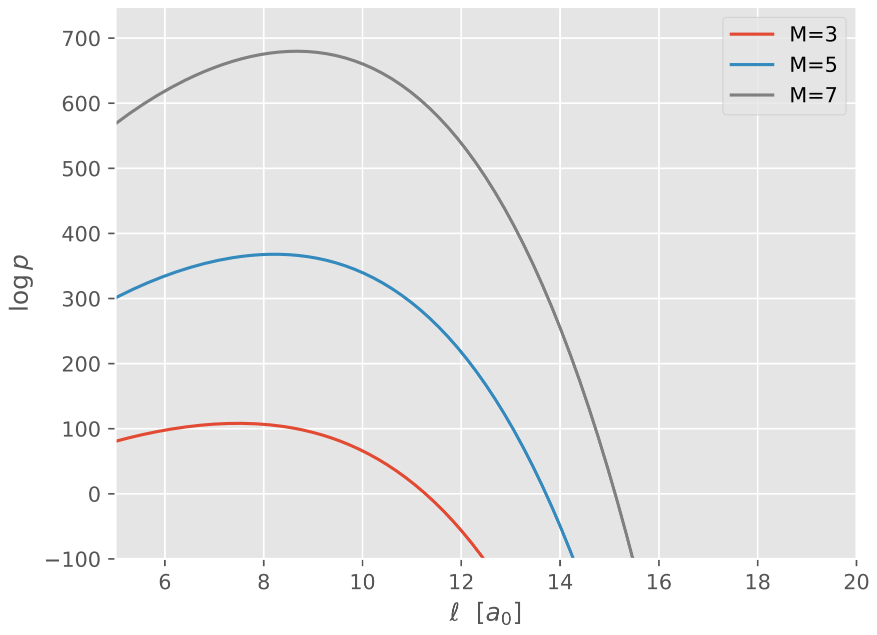


**Figure 6.10:** The dependece of the logratihm of the marginal likelihood on the length scale $\ell$ for the furan molecule. Internal 1 is shown on the left, internal 2 on the right. The number of training points, TP in the plot, refers to the number of steps.



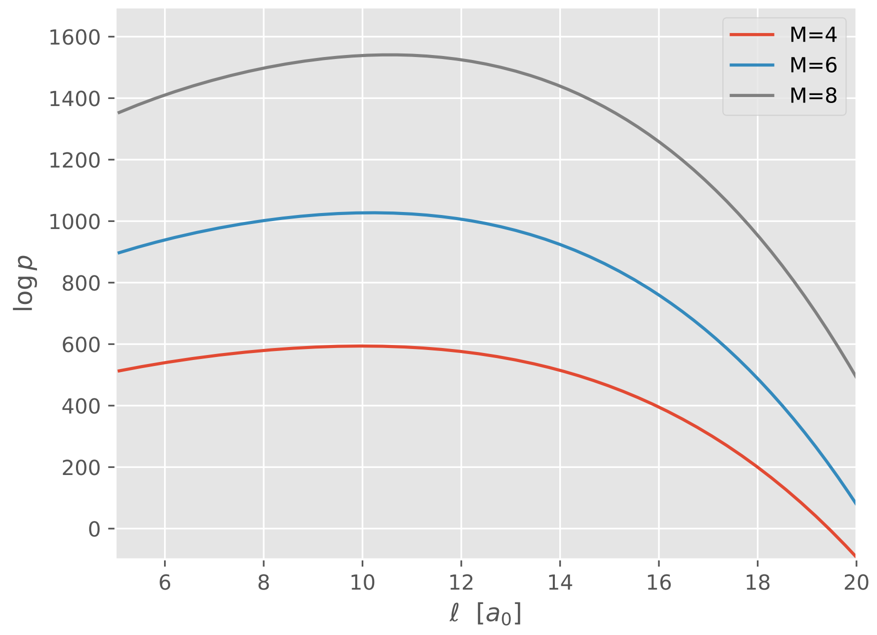


**Figure 6.11:** The logarithm of the marginal likelihood $\log p$ is plotted against the length scale $\ell$, for internal 1 (top) and for internal 2 (bottom). The number of steps, $M$ in the plot, refers to the number of training points. For this plot the molecule ethanol was used. Reprinted with permission from [1]. Copyright 2021 American Chemical Society.

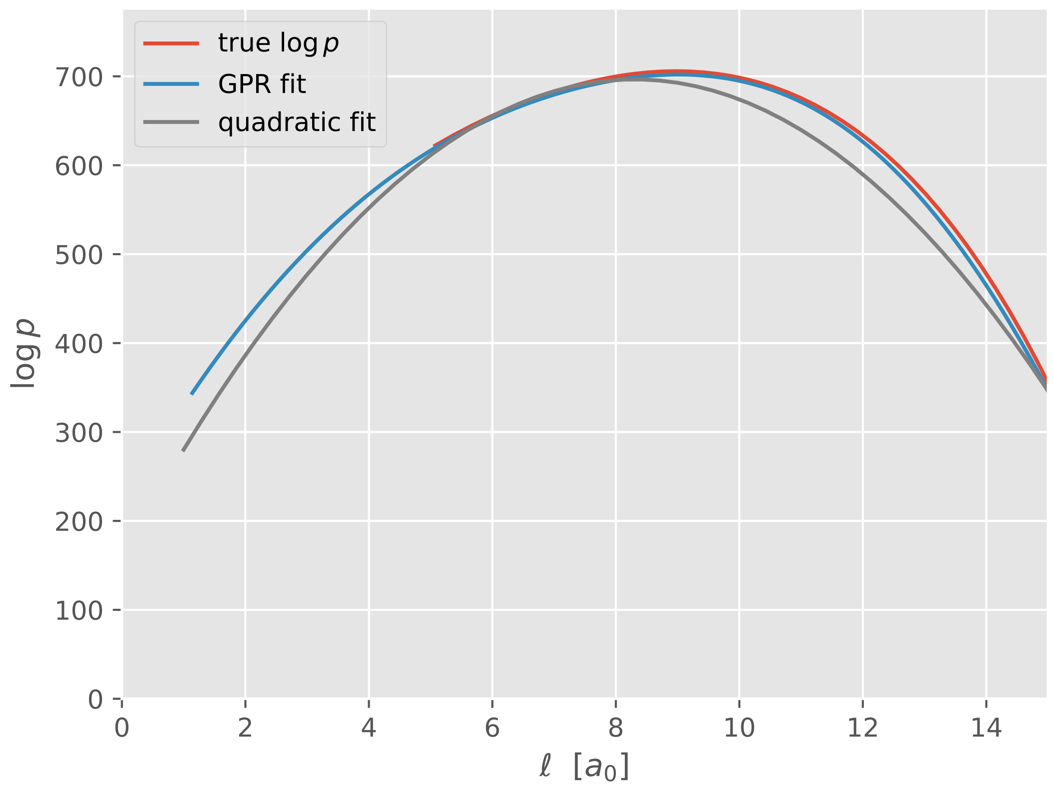
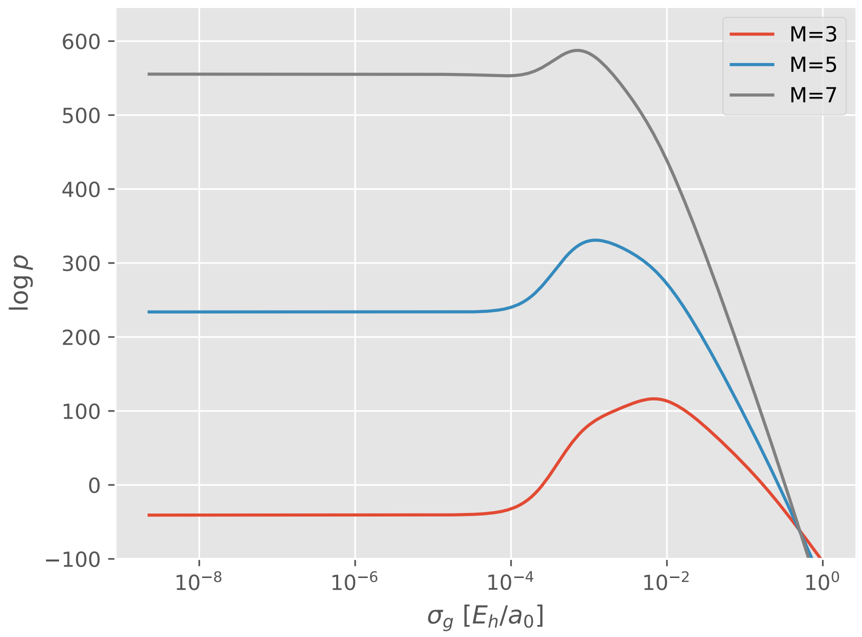


**Figure 6.12:** Optimizing $\log\, p$ with three points. As can be seen GPR fits better to the true log marginal likelihood than a quadratic fit.



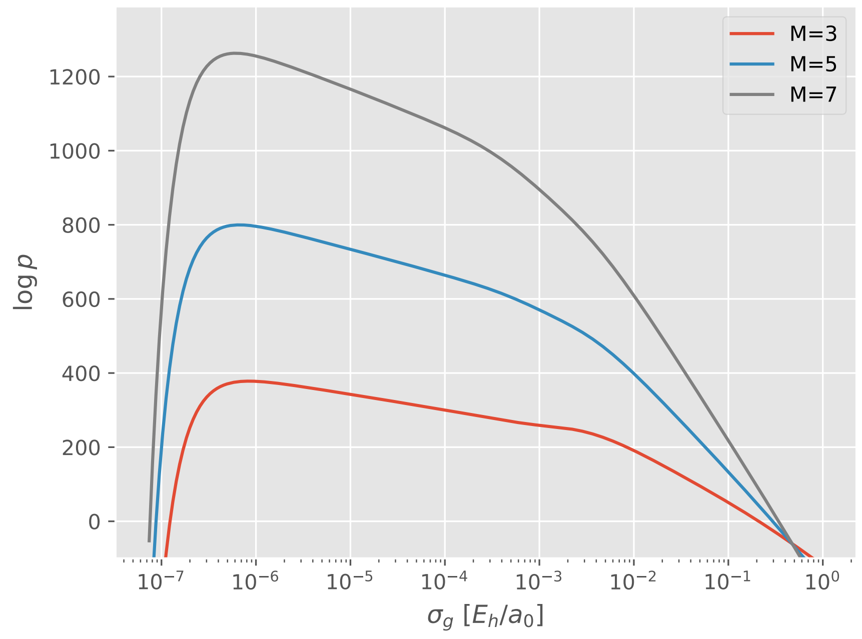
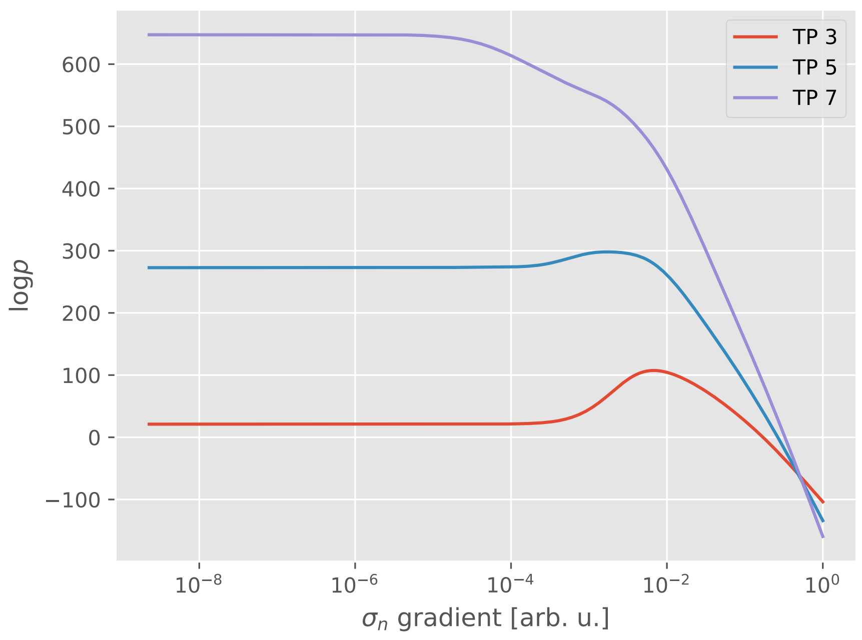


**Figure 6.13:** The logarithm of the marginal likelihood $\log p$ plotted against the gradient noise $\sigma_g$ for internal 1(top) and internal 2 (bottom). The gradient noise is plotted in a logarithmic scale. The molecule used for this plot is ethanol. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.



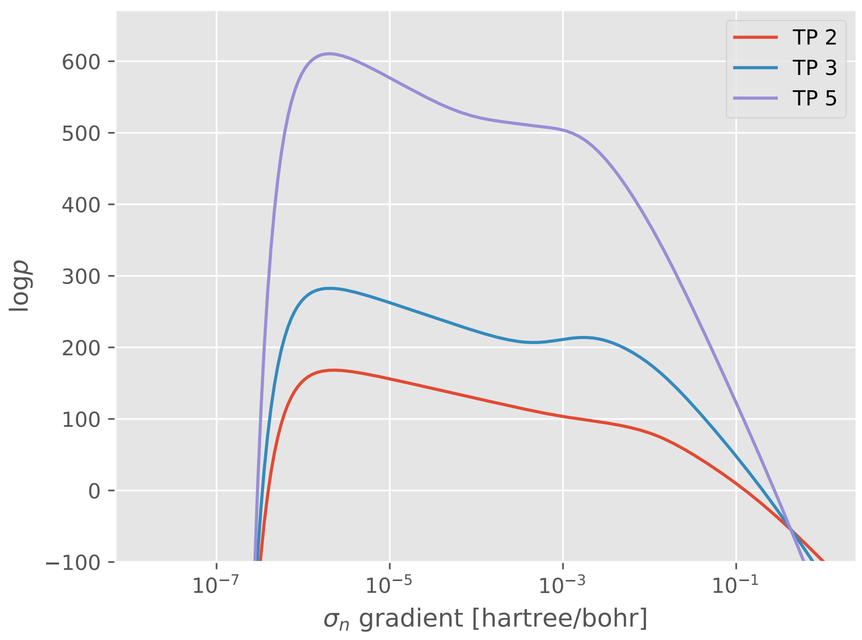


**Figure 6.14:** The logarithm of the marginal likelihood is plotted against the gradient noise, like in fig. 6.13. The molecule used in this plot is furan. The number of training points, TP in the plot, refers to the number of steps.

scaled unit matrix $\sigma\mathbf{I}$ in eq. (5.55) is replaced with a diagonal matrix. The first $M$ entries contain the noise for the energy, the remaining entries are filled with the noise for the gradient. While the energy noise parameter is set to $10^{-7}\,E_\mathrm{h}$, the gradient noise is variable. For both methods the marginal likelihood $\log p$ is plotted in fig. 6.13 for ethanol and in fig. 6.14 for the molecule furan. Both systems result in a similar plot for both methods. For internal 1 the marginal likelihood $\log p$ is independent of the gradient noise $\sigma_g$, if $\sigma_g < 10^{-4}\,E_\mathrm{h}/a_0$. For larger values, there are some differences between the molecules ethanol and furan. If the number of steps, i.e. training points, is below 7 there is a small maximum of $\log p$, which disappears at seven training points. In contrast for ethanol, the maximum remains for seven training points. Consequently all values below $10^{-4}\,E_\mathrm{h}/a_0$ can be chosen for method 1. The marginal likelihood $\log p$ in contrast looks different. They are shown in fig. 6.13 for ethanol and fig. 6.14 for furan. As can be seen in the plots, at a certain small noise parameter the marginal likelihood drops steeply. Therefore a distinct maximum can be found at around $\sigma_g \approx 10^{-6}\,E_\mathrm{h}/a_0$, for both systems ethanol and furan. At a low number of training points for ethanol (three in that case) a second lower maximum at around $\sigma_g \approx 10^{-3}\,E_\mathrm{h}/a_0$ appears. In contrast to ethanol, where this maximum disappears for a higher number of training points, furan shows a second maximum for more than five training points. This lower maximum is also roughly at $\sigma_g \approx 10^{-3}\,E_\mathrm{h}/a_0$. Since this maximum is either always smaller than the higher maximum at lower noise parameters or disappears at a higher amount of training points, this maximum can be ignored. Therefore the noise parameter for both methods is set to a constant value. For method 1 $\sigma_g = 10^{-7}\,E_\mathrm{h}/a_0$ is chosen and $\sigma_g = 10^{-6}\,E_\mathrm{h}/a_0$ for method 2.

The steep decrease for small values of $\sigma_g$ for method 2 is probably caused by the coordinate transformation, i.e. the chain rule. The $3N-6$ dimensional internal gradient is transformed into the $3N$ dimensional Cartesian gradient. This can be seen easily when looking at the sub-matrix $\frac{\partial^2 k(\mathbf{x}_m,\mathbf{x}_n)}{\partial \mathbf{x}_m \partial \mathbf{x}_n}$ of $\mathbf{K}$ from eq. (5.65). In matrix form, the transformation is written as

$$\nabla_x^2 k(\mathbf{x}_n,\mathbf{x}_m) = \tilde{B}_n^\top \nabla_q^2 k(\mathbf{q}_n,\mathbf{q}_m)\tilde{B}_m \tag{6.25}$$

with $\nabla_x^2 k(\mathbf{x}_n,\mathbf{x}_m) \in \mathbb{R}^{3N\times 3N}$, $\nabla_q^2 k(\mathbf{q}_n,\mathbf{q}_m) \in \mathbb{R}^{3N-6\times 3N-6}$ and $\tilde{B} \in \mathbb{R}^{3N-6\times 3N}$. The remaining six components correspond to translation and rotation. As the energy is invariant with respect to rotation and translation, these compounds are equal to zero. Therefore the sub-matrix is a singular matrix. However, this would lead to difficulties in the Cholesky decomposition. Therefore, a sufficiently large noise parameter for the gradient $\sigma_g$ has to be added. If it is too small the Cholesky decomposition will become unstable resulting in a decrease of $\log p$. This can be avoided by using suitable values for $\sigma_g$.

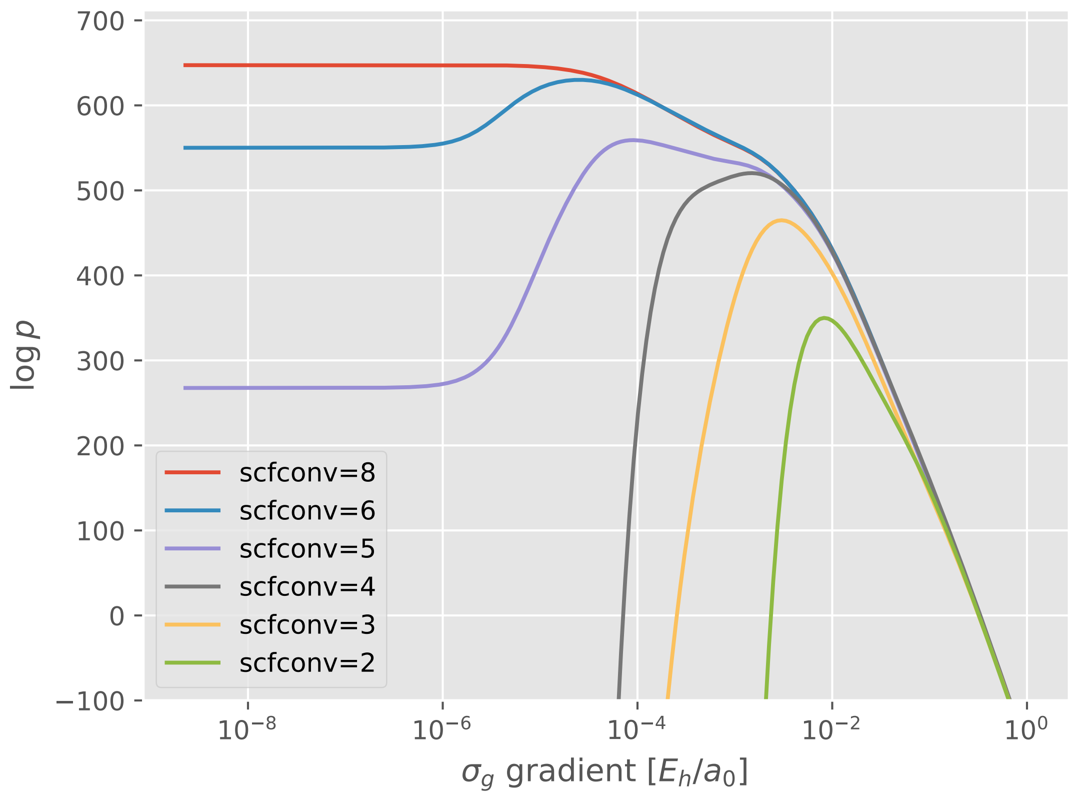


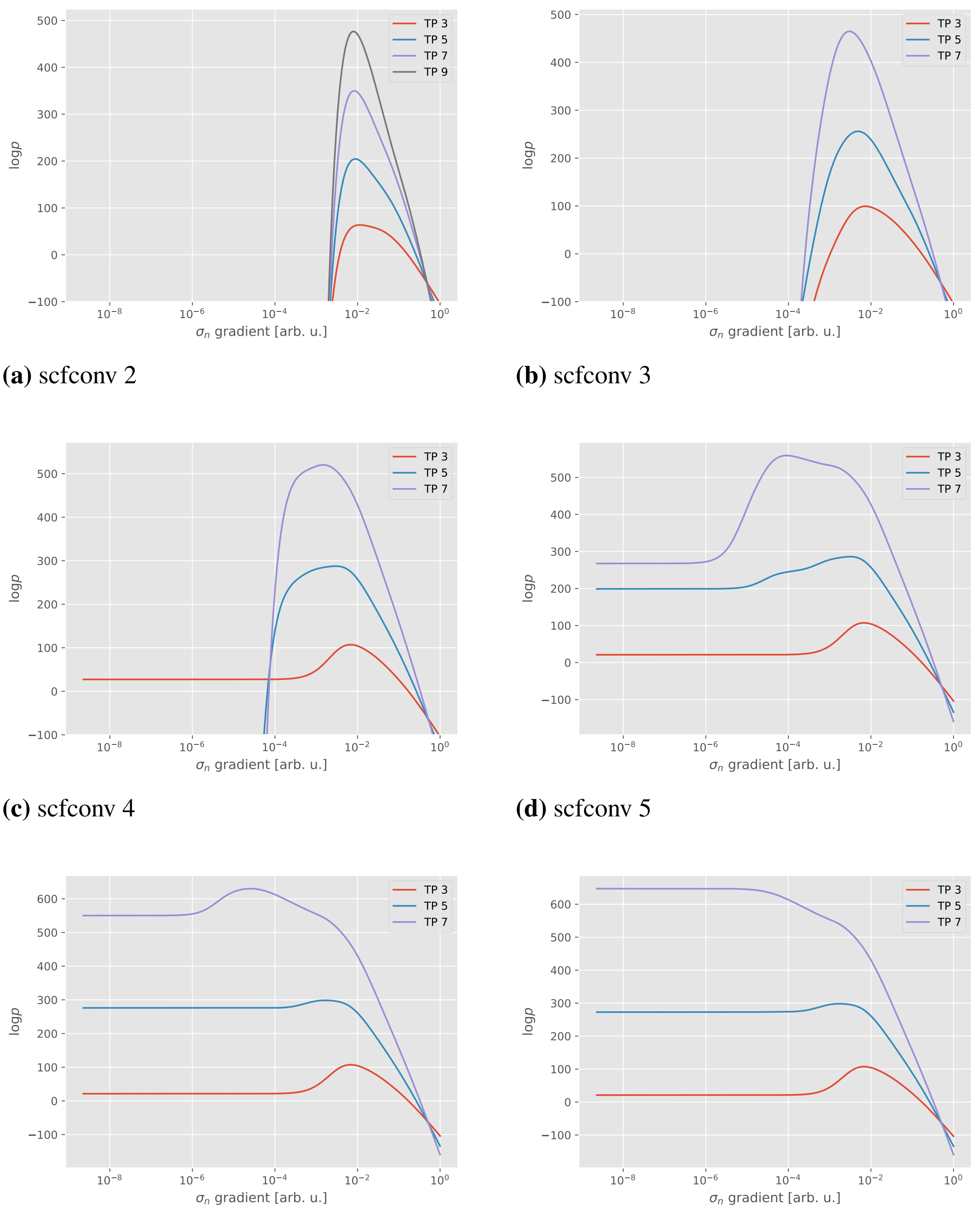
**Figure 6.15:** The logarithm of the marginal likelihood $\log\, p$ plotted against the gradient noise $\sigma_g$ for different scfconv. scfconv refers to different converegence criteria for the self-consistent field (SCF) cycles in TURBOMOLE. The method is internal 1 and the molecule is furan. Reprinted (adapted) with permission from [1]. Copyright 2021 American Chemical Society.

Since $\sigma_g$ can be interpreted as the noise of the gradient, it should depend on the level of theory as well as the SCF convergence behavior in the underlying DFT program. On the one hand, TURBOMOLE is known for still having good convergence of the SCF cycles for low noise values. On the other hand, GPR can deal with quite high noise levels, because they are directly incorporated as noise parameters. Therefore it can deal with less ideal ab initio programs. The Turbomole parameter scfconv, which indicates that SCF cycles are converged until two consecutive energies differ by less than $10^{-\text{scfconv}}\,E_\text{h}$, is used to show the consequences of higher noise in the gradient. Using different scfconv values for the SCF cycles, the corresponding $\log\,p$ are plotted against $\sigma_g$. For internal 1 and the molecule furan, this is shown in fig. 6.15. The maximum of $\log\,p$ shifts to higher values for lower values of scfconv, where the lowest value is achieved for scfconv $=\;2$ at $\sigma_g\;=\;10^{-2}\,E_\text{h}/a_0$. If the noise parameter $\sigma_g$ is chosen correctly, the geometry optimization still converges for scfconv $=\;3$. Approximately the same number of steps is required. This demonstrates the stability of the optimizer at high noise levels of the gradient. This can be useful for more difficult electronic structure methods, which have noise in the gradient, such as solvation methods.

For a different number of training points, the marginal likelihood plotted against the noise parameter of the gradient is shown for different scfconv in fig. 6.16 for internal 1 and internal 2 in fig. 6.17. The molecule is furan. As can be seen in fig. 6.16a, where scfconv $=\;2$ for internal 1, the noise is pretty good detected by the marginal likelihood. The maximum value for the gradient noise is $\sigma_g\;\approx\;10^{-2}$ and remains the same for different number of training points. Going to scfconv $=\;3$, which is the plot in fig. 6.16b, the maximum of the marginal likelihood is at $\sigma_g=10^{-3}$. Up to now, the maximum value for the marginal likelihood seems to correspond to the value set for scfconv. However, setting scfconv $=\;4$ seems to contradict, because the maximum now is slightly higher at $\sigma_g\approx10^{-3}$, the same as for scfconv $=\;3$.

Increasing the scfconv value to $5$ in fig. 6.16d shows a shift to lower noise values. For three respectively five training points a gradient noise $\sigma_g\;\leq\;10^{-3}$, can be chosen showing that the GPR can handle scfconv $=\;5$ quite well. Increasing the number of training points to 7 results in a clear maximum at $\sigma_g=10^{-5}$. If scfconv $=\;6$, as shown in fig. 6.16e the result is almost the same as for scfconv $=\;8$ in fig. 6.16f, which has already been discussed. This clearly shows that lower convergence criteria for the scf cycle in TURBOMOLE can be handled by GPR internal 1 quite well. This can be useful for systems, which converge for smaller scfconv values but not for tighter values and still lead to good geometry optimization results.

The logarithm of marginal likelihood $\log\,p$ dependent on the gradient noise $\sigma_g$ for different scfconv values for internal 2 can be seen in fig. 6.17. As can be seen in fig. 6.17a for scfconv



**(a)** scfconv 2

**(b)** scfconv 3

**(c)** scfconv 4

**(d)** scfconv 5

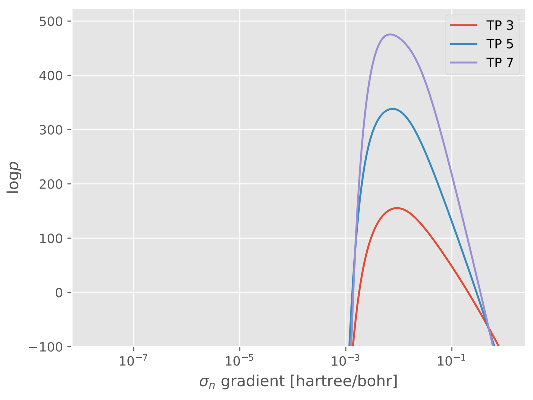
**(e)** scfconv 6

**(f)** scfconv 8

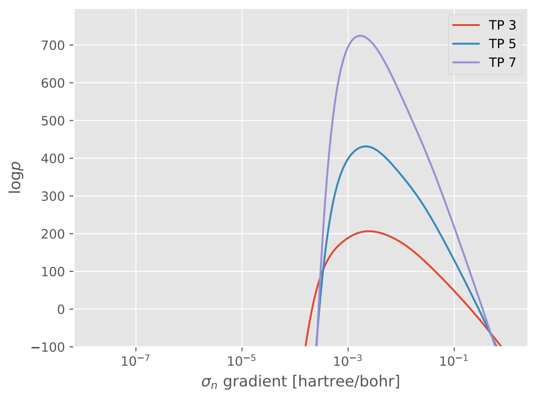
**Figure 6.16:** The logarithm of the marginal likelihood $\log\, p$ is plotted over the gradient noise $\sigma_g$ for different scfconv criteria in TURBOMOLE for internal 1 and different number of training points. The molecule used is furan. The number of training points, TP in the plot, refers to the number of steps.

$= 2$, the resulting plots for different training points is similar to internal 1. This is also the case for scfconv $= 3$, as shown in fig. 6.17b. The first differences between internal 1 and 2 arise for scfconv $= 4$, which can be seen in fig. 6.17c. For three training points there is already a steep decrease of the marginal likelihood at $\sigma_g \approx 10^{-6}$ indicating the instability of the Cholesky decomposition for too low noise values for internal 2. Increasing the number of training points to five, and seven results in a maximum at $\sigma_g \approx 10^{-4}$, which corresponds to the value for scfconv.

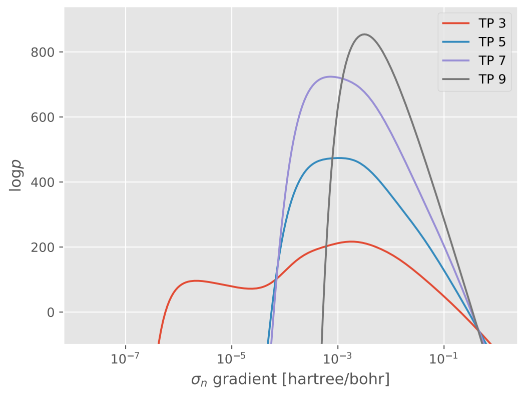
At scfconv $= 5$, the steep decrease at $\approx 10^{-6}$ appears for all training points. For a lower number of training points, a similar result is obtained as for scfconv $= 8$, which has been discussed before. For five training points, a maximum at $\sigma_g \approx 10^{4}$ can be seen. Increasing the convergence criteria in TURBOMOLE to scfconv $= 6$, shown in fig. 6.17e, leads to a similar result as for scfconv $= 8$. This shows that, as for internal 1, internal 2 can handle higher noise in the gradient quite well, and adapting the gradient noise $\sigma_g$ approximately to the scfconv value can lead to good results in geometry optimization.



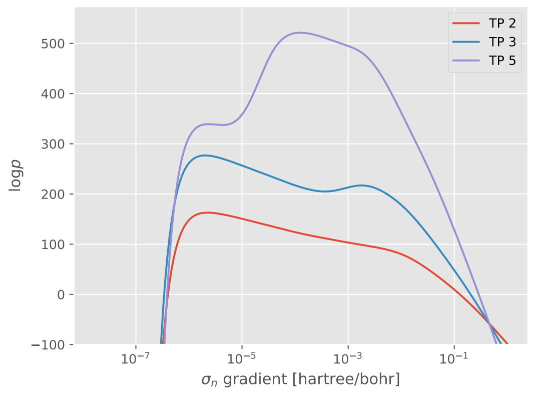
**(a)** scfconv 2



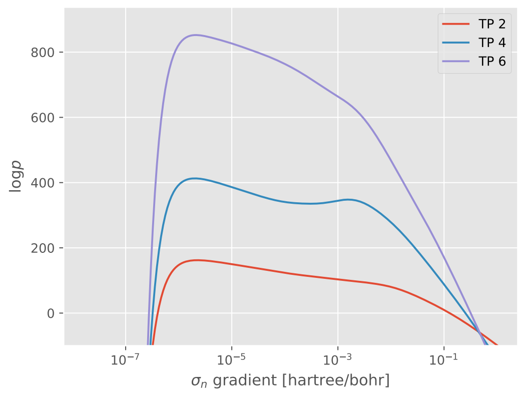
**(b)** scfconv 3



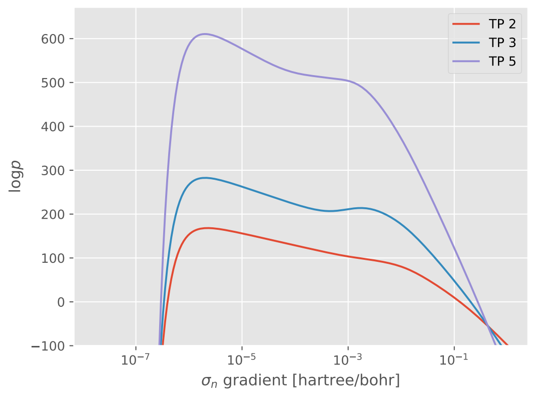
**(c)** scfconv 4



**(d)** scfconv 5



**(e)** scfconv 6



**(f)** scfconv 8

**Figure 6.17:** Different plots for the scfconv of TURBOMOLE of the logarithm of the marginal likelihood depend on the gradient noise $\sigma_g$ for internal 2. The molecule is furan. The number of training points, TP in the plot, refers to the number of steps.

## 6.3 Ideas for Improving Geometry Optimization Based on Gaussian Process Regression

Several algorithms to speed up the geometry optimization were also tested. They have in common that they belong to the class of active learning methods. For these approaches the next training point is selected according to an acquisition function. Two examples of acquisition functions will be presented in the following. The first one is the expected improvement expected improvement (EI) and the other is the probability of improvement probability of improvement (POI). Both acquisition functions did not improve the overall performance of the optimizer.

### 6.3.1 Expected Improvement

The expected improvement EI calculates how much improvement can be expected if a specific point would be added. The idea was introduced by Schonlau et al. [75] and Sasena [76]. Let's define $Y(\mathbf{x})$ as a random variable, which models the uncertainty of a function value at a certain point $\mathbf{x}$. This random variable is assumed to be normally distributed with mean and variance from the posterior of a GP. Let $f_{\text{min}}$ be the current best minimum, then an improvement $\xi$ is achieved if $Y(\mathbf{x}) = f_{\text{min}} - \xi$. The probability density is then given as

$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2(\mathbf{x})}} \exp\left(-\frac{(f_{\text{min}} - \xi - \mu(\mathbf{x}))^2}{2\sigma^2(\mathbf{x})}\right) \tag{6.26}$$

The EI is simply the expectation value of this distribution regarding $I$. This means the EI is obtained by integrating the above expression

$$\mathbb{E}[\xi] = \int\limits_0^\infty \frac{\xi}{\sqrt{2\pi\sigma^2(\mathbf{x})}} \exp\left(-\frac{(f_{\text{min}} - \xi - \mu(\mathbf{x}))^2}{2\sigma^2(\mathbf{x})}\right) \mathrm{d}\xi \tag{6.27}$$

Substituting $x = \frac{f_{\text{min}} - \xi - \mu}{\sigma}$ leads to

$$\int\limits_{\frac{f_{\text{min}} - \mu}{\sigma}}^{-\infty} \frac{\sigma x - (f_{\text{min}} - \mu)}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x \tag{6.28}$$

By using $\int_a^b f(x)\,\mathrm{d}x = -\int_b^a f(x)\,\mathrm{d}x$ and setting $u = \frac{f_{\text{min}}-\mu}{\sigma}$ this can be written as

$$-\int_{-\infty}^{u} \frac{\sigma x}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) \mathrm{d}x + \sigma u \int_{-\infty}^{u} \frac{\exp\left(\frac{-x^2}{2}\right)}{\sqrt{2\pi}} \mathrm{d}x \tag{6.29}$$

Since $\frac{\mathrm{d}}{\mathrm{d}x} \exp\left(-\frac{x^2}{2}\right) = -x \exp\left(-\frac{x^2}{2}\right)$, the left side can be easily evaluated as

$$-\int_{-\infty}^{u} \frac{\sigma x}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) \mathrm{d}x = \sigma\varphi(u), \tag{6.30}$$

where $\varphi(u)$ is the density function of a normal distribution. The right side can be expressed as the cumulative density function $\Phi$

$$\sigma u \int_{-\infty}^{u} \frac{\exp\left(\frac{-x^2}{2}\right)}{\sqrt{2\pi}} \mathrm{d}x = \sigma u(\Phi(u) - \Phi(-\infty)) = \sigma u \Phi(u) \tag{6.31}$$

In total, the EI is

$$\mathbb{E}[\xi] = \sigma(\mathbf{x}) \left(u\Phi(u) + \varphi(u)\right). \tag{6.32}$$

To maximize the EI the derivative with respect to $\mathbf{x}$ can be used for a quasi-Newton optimizer. The derivative is obtained by applying the product rule and the chain rule

$$\begin{aligned} \frac{\partial \mathbb{E}[\xi](x)}{\partial x_k} = {} & \frac{\partial \sigma(\mathbf{x})}{\partial x_k} u(\mathbf{x})\Phi(u(\mathbf{x})) + \sigma(\mathbf{x})\frac{\partial u(\mathbf{x})}{\partial x_k}\Phi(u(\mathbf{x})) \\ & + \sigma(\mathbf{x})u(\mathbf{x})\frac{\partial \Phi(u)}{\partial u}\frac{\partial u}{\partial x_k} + \frac{\partial \sigma(\mathbf{x})}{\partial x_k}\varphi(u) + \sigma(\mathbf{x})\frac{\partial \varphi}{\partial u}\frac{\partial u}{\partial x_k} \end{aligned} \tag{6.33}$$

Since $\sigma = \sqrt{\mathbb{V}}$ of the GP posterior, the first derivative term is written as

$$\frac{\partial \sigma(\mathbf{x})}{\partial x_k} = \frac{\partial \mathbb{V}[\mathbf{x}]}{\partial x_k} \frac{1}{2\sqrt{\mathbb{V}[\mathbf{x}]}}. \tag{6.34}$$

The next derivative term is

$$\frac{\partial u(\mathbf{x})}{\partial x_k} = \frac{\partial \mathbb{V}[\mathbf{x}]}{\partial x_k} \frac{m(\mathbf{x}) - f_{\text{min}}}{\mathbb{V}^{3/2}[\mathbf{x}]} - \frac{\partial m(\mathbf{x})}{\partial x_k} \frac{1}{\sqrt{\mathbb{V}[\mathbf{x}]}} \tag{6.35}$$

which is also required for using the chain rule. The next relevant derivatives are $\frac{\partial\Phi(u)}{\partial u} = \varphi(u)$ and $\frac{\partial\varphi(u)}{\partial u} = -u\varphi(u)$. Plugging all those derivative parts together leads to the following expression

$$\frac{\partial\mathbb{E}[\xi](x)}{\partial x_k} = \frac{\partial\mathbb{V}[\mathbf{x}]}{\partial x_k}\frac{\varphi(u)}{2\sqrt{\mathbb{V}[\mathbf{x}]}} - \frac{\partial m(x)}{\partial x_k}\Phi(u). \tag{6.36}$$

In fig. 6.18 an example is shown for the one-dimensional test function

$$f(x) = \left(-x^2 + x + 1\right)^2 - \cos(x). \tag{6.37}$$

Starting with 5 training points, it requires 5 iterations to converge to the minimum. In step 1 the EI has the maximum closer to the real minimum, compared to the minimum of the GPR surface. The green cross in fig. 6.18 shows the maximum position of the EI function. Adding this point to the GPR-surface as new training point results in an EI function whose value is smaller compared to step 1. This shows that the improvement of further points is not as good as previously. Again the maximum of the EI lies closer to the real minimum, compared to the GPR minimum. In the third step, the EI gets again smaller, indicating that the real minimum is pretty close. This time the maximum of the EI is near the real minimum. In step 4 the values for the EI seem to be zero everywhere, however, there is still a small maximum, that gives small improvement regards a minimum. Using this training point, the GPR-surface minimum corresponds to the real minimum of the function. Also, the function is pretty good described and interpolated around the minimum by the GPR surface.

In contrast to the probability of improvement POI, this acquisition function does not need a specific improvement (the $T$ value for the POI). In addition, the EI has a stopping rule. When the EI is less than some small positive number, then one can stop the optimization. Both methods, the EI and the POI, use the variance of the GPR to force the exploration of regions with few sample points, i.e. where the variance is high. However, these methods treat the variance as if it would be correct, which is not always the case.

### 6.3.2 Probability of Improvement

The goal is to maximize the probability of improvement (POI) regards some target $\mathfrak{T}$ to select iterative new points. The model of the uncertainty of the function value is realized by a random variable $Y(\mathbf{x})$ with mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$. The current best function value is the minimum $f_{\text{min}}$. Then the value for improvement is some number $\mathfrak{T} < f_{\text{min}}$. The probability of improving is then $P(Y(\mathbf{x}) \leq \mathfrak{T})$. By assuming that $y(\mathbf{x})$ is normally distributed, the POI is

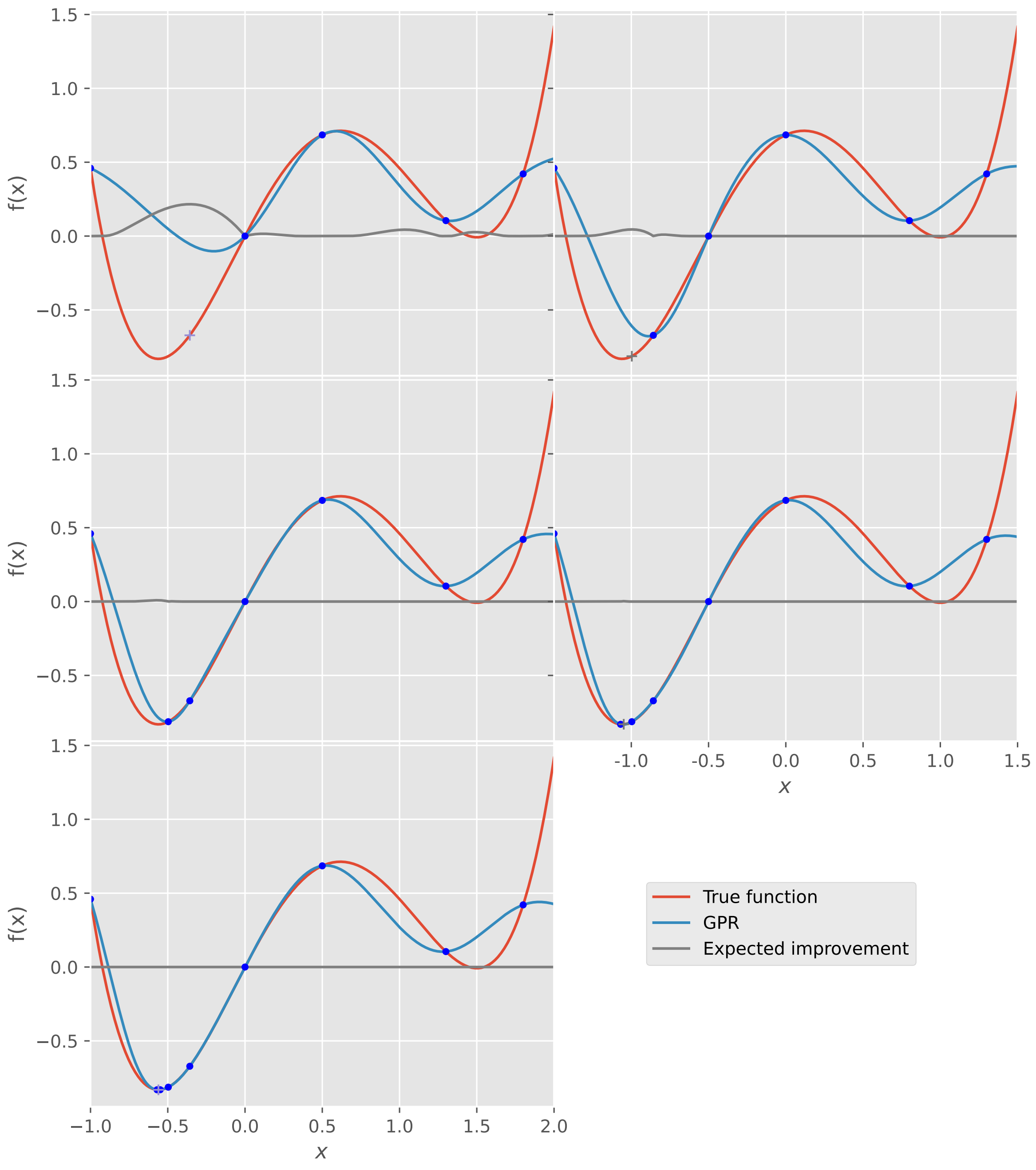


**Figure 6.18:** EI for the minimum search. The blue dots in each plot correspond to the up to then included training points with which the GPR was constructed. The grey cross is the maximum of the expected improvement and used as a training point, where the real function is evaluated next. In the last plot it is the found minimum of the function.

given

$$P(Y(\mathbf{x}) \leq \mathfrak{T}) = \Phi\left(\frac{\mathfrak{T} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right), \tag{6.38}$$

where $\Phi$ is the cumulative density function of a normal distribution. This idea was introduced by Kushner [77] for one dimension and improved by Perttunen [78], Elder [79] and Mockus [80] to higher dimension. The standard error in the region, where more and more points around the current best points are sampled, becomes small. Therefore, $\frac{\mathfrak{T}-\mu(\mathbf{x})}{\sigma(\mathbf{x})}$ is negative, because $T$ is usually smaller then $\mu(\mathbf{x})$. As a result, the POI is very small. At some point this probability is so small, the algorithm is forced to search in areas, where the standard error is higher, i.e. in regions that have been sampled not so well.

The derivative can be simply calculated by applying the chain rule:

$$\frac{\partial P(Y(\mathbf{x}) \leq \mathfrak{T})}{\partial x_k} = \frac{\partial \Phi(u)}{\partial u}\frac{\partial u}{\partial x_k} = \varphi(u)\left(\frac{\mu(\mathbf{x}) - \mathfrak{T}}{2\mathbb{V}[\mathbf{x}]^{3/2}}\frac{\partial \mathbb{V}[\mathbf{x}]}{\partial x_k} - \frac{\partial \mu(x)}{\partial x_k}\frac{1}{\sqrt{\mathbb{V}[\mathbf{x}]}}\right). \tag{6.39}$$

The task is now to select an improvement $\mathfrak{T}$. An easy requirement is, that $\mathfrak{T} < f_{\text{min}}$ below the current best-evaluated function value. Using the GPR surrogate minimum value $s_{\text{min}}$, a value for $\mathfrak{T}$ can be constructed as follows Jones [67]

$$\mathfrak{T} = s_{\text{min}} - \alpha\left(f_{\text{max}} - f_{\text{min}}\right) \tag{6.40}$$

where $\alpha$ is an adaptable value. Intuitively, choosing large values for $\alpha$ will lead to a more global search, while small values will force the algorithm to search more locally. An idea would be to start with a medium to large value of $\alpha$, for example, $\alpha = 0.25$, and then steadily decrease the value to enforce a more local search. An example of how the POI selects the next points to evaluate is shown in fig. 6.19

Looking at the first step, three maxima of the acquisition function can be seen. If the highest maximum is selected, the right direction is chosen. Adding this maximum to the GPR-surface results in the second step in an acquisition function with only one maximum. The POI at that point is also lower indicating that only a small improvement of the surrogate surface will occur. In the third step, no maximum in the acquisition function can be seen, however numerically there is a maximum. In the fourth step, the complete acquisition function is almost zero, indicating that locally there are no improvements possible regards a minimum. And indeed the minimum of the GPR surface corresponds to the minimum of the real function. In contrast to the EI, the GPR surface around the minimum is somewhat better described. This is the advantage of the POI to describe the surrounding of the minimum better if the correct

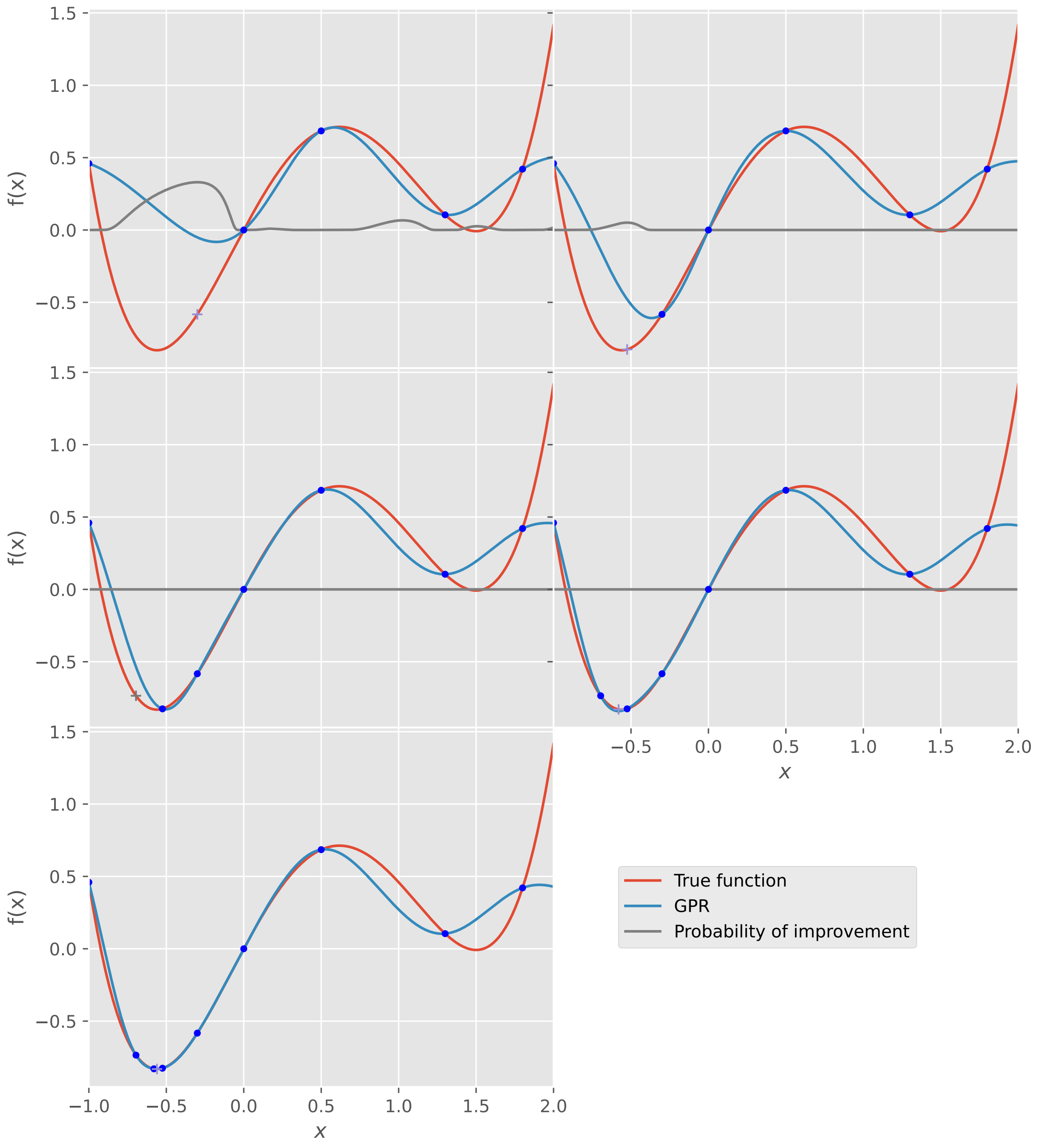


**Figure 6.19:** Probability of improvement for the minimum search. The blue dots in each plot are the training points for constructing the GPR. The grey cross is the maximum of the POI used as a point where the function should be evalueated next.

target is chosen. In general, this is difficult to choose a correct value for $\alpha$, and therefore sampling with different $\alpha$ can be advantageous.

### 6.3.3 Results and Discussion

EI and POI are tested for the 30 molecules from fig. 6.4. In table 6.3 the number of steps for those two methods is listed. They were again tested for internal 1 and internal 2. For internal 1 it can be easily seen that the required number of steps are similar to using local optimization on on the surrogate surface with overshooting, except for system 23, 26, 27, 28, and 30. There, no convergence after 100 steps could be achieved. These are the systems that already required more steps compared to the other systems without any acquisition function. The reason lies in the global search character of the EI algorithm. If it does not find a minimum in the beginning, like for the other systems, the algorithm tries to explore the PES more globally to find a minimum. Using more steps would probably lead to convergence, but for a different minimum (and probably lower minimum) than without using EI.

The $\alpha$-value in the target $\mathfrak{T}$ for the POI was set to zero. This means that the probability is maximized regards the local minimum of the GPR surface. The reason for this value for $\alpha$ lies in the locality of the minimum search, only local minima are of interest, because a global optimization would be too expansive for large molecules, which have a large number of minima. That is different from the EI, which is essentially for global optimization. With both algorithms, a global search would be theoretically possible. For the POI larger $\alpha$-values would allow this. As can be seen from table 6.3 internal 1 and internal 2 perform similarly, however for both, EI and POI, the results are either the same or a bit worse compared to those obtained wihtout them. The reason is the minimum of the surrogate surface is already a good prediction regards minimization and therefore mostly similar results can be obtained.

Subtracting the number of steps for the rhodium complex in table 6.2, a 450 total number of steps are required for internal 1, compared to internal 1 with thePOI, which required 481 steps. This means that internal 1 is a bit better when the POI has not been used.

If the convergence criteria are not fulfilled after 100 steps for EI the number of steps is set to 100. Internal 1 with EI required 750 steps, while internal 2 required 858.

The POI required in total 521 steps, compared to 515 steps for pure internal 2. The difference here is only 6 steps, meaning that the POI for internal 2 performs better than for internal 1.

A possibility to improve at least the POI could be to choose different values for $\alpha$ for the target $T$. This would lead on the one hand to a more global search, and on the other hand, if

**Table 6.3:** The number of steps for EI and POI as acquisition functions are compared between internal 1 and internal 2. nc means, the algorithm did not converged after 100 steps.

| Acquisition function | Expected Improvement | | Probability of Improvement | |
|---|---|---|---|---|
| Coordinate System | internal 1 | internal 2 | internal 1 | internal 2 |
| 1 Water | 6 | 5 | 6 | 5 |
| 2 Ammonia | 7 | 8 | 7 | 7 |
| 3 Ethane | 6 | 6 | 6 | 6 |
| 4 Acetylene | 6 | 5 | 6 | 5 |
| 5 Allene | 4 | 5 | 4 | 5 |
| 6 Hydroxysulfane | 13 | 13 | 13 | 12 |
| 7 Benzene | 6 | 5 | 6 | 5 |
| 8 Methylamine | 24 | 22 | 22 | 27 |
| 9 Ethanol | 8 | 8 | 8 | 8 |
| 10 Acetone | 10 | 10 | 10 | 10 |
| 11 Disilyl-ether | 10 | 9 | 10 | 9 |
| 12 1,3,5-Trisilacyyclohexane | 22 | nc | 18 | 18 |
| 13 Benzaldehyde | 8 | 9 | 8 | 9 |
| 14 1,3-Difluorobenzene | 7 | 8 | 7 | 8 |
| 15 1,3,5-Trifluorobenzene | 7 | 8 | 7 | 8 |
| 16 Neopentane | 5 | 4 | 5 | 4 |
| 17 Furan | 8 | 8 | 8 | 8 |
| 18 Napthalene | 7 | 7 | 7 | 7 |
| 19 1,5-Difluoronaphtalene | 8 | 9 | 8 | 9 |
| 20 2-Hydroxybicyclopentane | 23 | 25 | 23 | 25 |
| 21 ACHTAR10 | 18 | 22 | 18 | 21 |
| 22 ACANIL01 | 13 | 20 | 13 | 17 |
| 23 Benzidine | nc | nc | 43 | 45 |
| 24 Pterin | 17 | 22 | 19 | 23 |
| 25 Difuropyrazine | 8 | 8 | 8 | 8 |
| 26 Mesityl-oxide | nc | nc | 76 | 75 |
| 27 Histidine | nc | nc | 55 | 62 |
| 28 Dimethylpentane | nc | nc | 15 | 23 |
| 29 Caffeine | 9 | 12 | 11 | 12 |
| 30 Menthone | nc | nc | 32 | 40 |
| total number of steps | 750 | 858 | 481 | 521 |

close to a minimum, a more local search.

## 6.4 Conclusion

GPR optimization in internal coordinates reduces the number of steps compared to Cartesian coordinates for both internal 1 and internal 2. The number of required energy and gradient evaluations of the external ab-initio program are similar for both methods. The main difference between the two methods is the iterative back transformation. While internal 1 does require the back transformation for each ab initio calculation, internal 2 does not require it. The back transformation can fail for special geometries of molecules, which are not treated safely by the primitive internal coordinates to generate the delocalized internal coordinates. Most problems arise for linearly connected atoms since they cannot be described as a single angle. Usually, three or fewer linearly connected atoms can be treated quite safely by the algorithm. However, four and more linearly connected atoms cause problems, but these molecules are rather rare in practice.

Internal 2 produces more computational costs than internal 1 because the covariance matrix is transformed from internal to Cartesian coordinates in each micro-iteration. This can be seen for example in system 27 (histidine) where internal 1 needs 14.2 s for all GPR operations. Internal 2 requires 133.1 s on an Intel Core i5-4590. In total, internal 2 is more robust compared to internal 1, while internal 1 is in general faster.

The hyperparameters relevant for geometry optimization in internal coordinates for GPR are the length scale $\ell$ and the gradient noise $\sigma_g$. Since they can be treated separately, both were investigated independently of each other. The length scale is fixed at $\ell = 13\,a_0$ for both methods and is adapted throughout the optimization procedure. The noise parameter for the gradient is somehow different for both methods. For internal 1 everything below $\sigma_g \leq 10^{-4}\,E_h/a_0$ is well suited, and $\sigma_g = 10^{-7}\,E_h/a_0$ was chosen, because the noise parameter also acts as a regularization and allows a numerically stable Cholesky decomposition.

Internal 2, however, required a higher noise. Here, a clear maximum of the marginal likelihood could be observed at $\sigma_g = 10^{-6}\,E_h/a =$ and is, therefore, chosen as gradient noise. The reason for this behavior lies in the transformation from a $3N-6$-dimensional gradient to a $3N$-dimensional Cartesian gradient.

Additionally, the dependence of the gradient noise on different convergence criteria of the SCF cycles of TURBOMOLE is investigated. Here, the maximum of the marginal likelihood shifts to higher gradient noise values for lower convergence criteria. This is useful for the GPR-optimizer because it can handle more noise in the ab initio program by adapting the noise parameter. At the convergence criteria of scfconv=3 still, the same amount of steps are required for the convergence of the optimizer. This demonstrates the robustness of the optimizer.

To reduce the required number of steps, two other methods were tested. Both methods use an acquisition function to find the next point. The first acquisition function is the EI. It uses a random variable to model the uncertainty of the function value, i.e. the observations, in this case, the energy from the ab initio program. Then the expectation for an improvement $\xi$ was calculated.

The other acquisition function is the POI. It also uses a random variable with mean and variance of the posterior of a Gaussian process. In contrast to the EI, a target $\mathfrak{T}$ has to be specified. The random variable is a Gaussian distribution and, therefore, with the target $\mathfrak{T}$ the probability to be better than the target can be calculated and in the end, maximized. The target here was chosen for local optimization. In total, the results of internal 1 and 2 without any acquisition function were similar or better compared to using an acquisition function. The main reason is that in geometry optimization no global minimum is searched for, but a local minimum. Using the minimum of the GPR surface in combination with overshooting leads in general to better results compared to an algorithm, that is designed for global optimization.

In summary, two methods for combining internal coordinates with GPR to find a local minimum of the PES has been presented. Internal 1 uses a GPR surrogate surface in internal coordinates and internal 2 in Cartesian coordinates. The GPR covariance function is in both methods calculated in internal coordinates. They show similar results and both are better than Cartesian coordinates. In addition, a simple approach for the hyperparameter optimization is introduced. There was no significant improvement when an acquisition function was used to improve the optimization procedure. EI and POI are more relevant for global optimization.

# 7 Transition State Search using Gaussian Process Regression

In this chapter, the transition state search by using a GPR-surrogate surface is extended to internal coordinates. The transition state (TS) is the surface separating two minima. The lowest point is a first-order saddle point. In contrast to GPR-minimization, only internal method 1 will be applied. The reason is, that the method internal 2 is too expensive to use because additional matrix-matrix multiplications occur. This will be discussed in the next section 7.1. In addition, the problems of internal 1 are discussed, mainly the unstable back-transformation. An attempt to overcome this problem is to use an acquisition function, namely predictive entropy searching, originally used for optimization [81]. Predictive entropy search is extended to be useful for transition state search, by incorporating the corresponding constraint for saddle points of first order. This will be discussed in section 7.1.2. In the end, the result for transition state search based on Gaussian process regression in internal coordinates (GPRTS internal) and the transition state search based on predictive entropy search in internal coordinates (GPRTS-PES) will be compared in section 7.2. In this section, the choice of hyperparameters is discussed too. Predictive entropy search leads to similar or slightly better results than purely using internal 1. All quantities in this chapter are in atomic units, unless mentioned otherwise.

## 7.1 GPR-Transition-State Search

In this section, the transition state search based on GPR is explained. It is abbreviated as Gaussian process regression for transition state search (GPRTS). Since the Hessian of the posterior mean function of the GPR can easily be calculated, P-RFO [42], as explained in section 3.2.2, is the method of choice on the GPR-surrogate surface. Since P-RFO requires a qualitative good Hessian, at least in the direction of the eigenvector corresponding to the lowest eigenvalue, two approaches are presented in the next sections.

In the next section, section 7.1.1, a detailed description of the algorithm is presented, while in section 7.1.2 an alternative approach to sample points is given.

If enough points are sampled, both approaches use the same technique to find a saddle point of first order on the GPR-surrogate surface.

The same convergence criteria as in section 6.1 are used. However, there is an additional criterion for the change in energy in Hartree

$$\Delta E < \eta_{\mathrm{E}} := \frac{\eta}{450}. \tag{7.1}$$

The prior mean from eq. (6.7) is set to

$$\bar{E} = \frac{1}{M}\sum_{n=1}^{M} E_n, \tag{7.2}$$

which is the mean of the observed energies, which is also used by Denzel et al. [22].

### 7.1.1 GPR-TS in Internal Coordinates

Method 2 from section 6.1.2 suggests extending it to transition state search. Therefore the evaluation of the Hessian on the surrogate surface is required. To calculate the second derivative $\frac{\partial^2 E(\mathbf{q}(\mathbf{x}))}{\partial x_k \partial x_l}$, the first derivative requires the term, that is canceled out in eq. (6.18). Then the

Hessian of the GPR surrogate is

$$\begin{aligned}
\frac{\partial^2 E(\mathbf{q}(\mathbf{x}))}{\partial x_k \partial x_l} = \sum_{n=1}^{M} \alpha_n \Bigg[ & \sum_{i=1}^{N_i}\sum_{j=1}^{N_i} \frac{\partial q_i}{\partial x_k} \frac{\partial^2 k(\mathbf{q},\mathbf{q}_n)}{\partial q_i \partial q_j} \frac{\partial q_j}{\partial x_l} + \sum_{i=1}^{N_i} \frac{\partial k(\mathbf{q},\mathbf{q}_n)}{\partial q_i} \frac{\partial^2 q_i}{\partial x_k \partial x_l} \Bigg] \\
+ \sum_{n=1}^{M}\sum_{i=1}^{N_x} \beta_{n,i} \Bigg[ & \sum_{j=1}^{N_i}\sum_{r=1}^{N_i} \frac{\partial^2 q_j}{\partial x_k \partial x_l} \frac{\partial k(\mathbf{q},\mathbf{q}_n)}{\partial q_j \partial q_{n,r}} \frac{\partial q_{n,r}}{\partial x_{n,i}} \\
& + \sum_{j=1}^{N_i}\sum_{r=1}^{N_i}\sum_{t=1}^{N_i} \frac{\partial q_t}{\partial x_k} \frac{\partial q_j}{\partial x_l} \frac{\partial q_{n,r}}{\partial x_{n,i}} \frac{\partial^3 k(\mathbf{q},\mathbf{q}_n)}{\partial q_j \partial q_t \partial q_{n,r}} \\
& + \sum_{j=1}^{N_i}\sum_{r=1}^{N_i} \frac{\partial q_j}{\partial x_l} \frac{\partial^2 k(\mathbf{q},\mathbf{q}_n)}{\partial q_j \partial q_{n,r}} \frac{\partial^2 q_{n,r}}{\partial x_k \partial x_{n,i}} \\
& + \sum_{j=1}^{N_i}\sum_{r=1}^{N_i} \frac{\partial q_r}{\partial x_k} \frac{\partial^2 k(\mathbf{q},\mathbf{q}_n)}{\partial q_r \partial q_{n,r}} \frac{\partial^2 q_{n,j}}{\partial x_l \partial x_{n,i}} \\
& + \cancel{\sum_{j=1}^{N_i} \frac{\partial k(\mathbf{q},\mathbf{q}_n)}{\partial q_{n,j}} \frac{\partial^3 q_{n,j}}{\partial x_k \partial x_l \partial x_{n,i}}} \Bigg]
\end{aligned} \tag{7.3}$$

The last term including the third derivative of the internal coordinates $\mathbf{q}$ with respect to the Cartesian coordinates $\mathbf{x}$ drops out because of the independence of the training points and the symmetry of the covariance function. However, the second derivative of the internal coordinates with respect to the Cartesian coordinates has to be calculated. In addition, there are more matrix-matrix multiplications for each calculation of the Hessian, making it impractical to use this approach for transition-state search. At least five summation loops are necessary for the calculation of the Hessian, including at least six loops for the second derivatives of the internal coordinates with respect to the Cartesian coordinates, for one single Hessian evaluation on the GPR surface. This leaves only method internal 1 as a possibility to include internal coordinates for the GPR-based transition-state search.

The Hessian for internal 1 can be calculated using

$$\frac{\partial^2 E(\mathbf{q})}{\partial q_k \partial q_l} = \sum_{n=1}^{M} \alpha_n \frac{\partial k(\mathbf{q},\mathbf{q}_n)}{\partial q_k \partial q_l} + \sum_{n=1}^{M}\sum_{i=1}^{N_i} \beta_{n,i} \frac{\partial^3 k(\mathbf{q},\mathbf{q}_n)}{\partial q_k \partial q_l \partial q_{n,l}}, \tag{7.4}$$

which can be used for transition state search via P-RFO on the GPR-surface.

To have a good Hessian required for P-RFO, first, a rotation of a dimer has to be performed to align the Hessian in the correct direction. Since the evaluation of the Hessian with only one training point is zero, a random point is selected, whereas in [22] the point $\mathbf{x}_1 + R/\sqrt{N_x} \cdot \mathbf{1}$ in

Cartesian coordinates is used, where $\mathbf{x}_1$ is the intitial Cartesian geometry and $N_x = 3N_{\text{at}}$.

Now, $\mathbf{q}_1$ is the initial geometry in internal coordinates and $d = N_i$ is the dimension of the internal coordinates. With this random point and its energy and gradient a new GPR surface is built. Then a dimer is constructed using the first training point, $\mathbf{q}_1$, as the dimer midpoint $\mathbf{q}_{\text{mid}}$ and the random point, $\mathbf{q}_{\text{rand}}$, as the dimer endpoint $\mathbf{q}_{\text{end}}$. Their distance is $R$. Here, $R$ is set to $R = 0.5\,a_0$. This dimer is then rotated, see section 3.2.1 for details, by calculating the Hessian at $\mathbf{q}_{\text{mid}}$ and aligning the dimer in the direction of the lowest eigenvalue of the Hessian, i.e. performing a rotation in the direction of the corresponding eigenvector. Thus, the new endpoint $\mathbf{q}'_{\text{end}}$ is obtained through

$$\mathbf{q}'_{\text{end}} = \mathbf{q}_{\text{mid}} + \frac{R}{|\mathbf{v}|}\mathbf{v} \tag{7.5}$$

where $\mathbf{v}$ is the eigenvector of the Hessian evaluated at $\mathbf{q}_{\text{mid}}$, which corresponds to the lowest eigenvalue.

Then this new endpoint is selected as a new training point, where energy and gradient have to be evaluated. These rotations are repeated until the convergence criterion for the rotation is fulfilled. Therefore the cosine of the angle

$$\cos(\gamma) = \left| \frac{(\mathbf{q}_{\text{end}} - \mathbf{q}_{\text{mid}}) \cdot (\mathbf{q}'_{\text{end}} - \mathbf{q}_{\text{mid}})}{R^2} \right| \tag{7.6}$$

is calculated, with $\mathbf{q}_{\text{mid}}$ the midpoint, $\mathbf{q}_{\text{end}}$ the new endpoint after rotation, $\mathbf{q}'_{\text{end}}$ the old endpoint and $R$ the dimer length. For the first rotation, the convergence criterion is $\cos(\gamma) > 1 - 10^{-4}$ and all other rotations are converged when $\cos(\gamma) > 1 - 10^{-3}$. After the rotation is converged, the Hessian, at least in the direction of the lowest eigenvalue, is good enough for a translation on the GPR surface using P-RFO. Then the dimer is translated via P-RFO to this new saddle point guess from the GPR surface. Afterwards, energy and gradient are calculated, repeating this procedure until the global convergence criteria are fulfilled. If no convergence occurred after $\lceil \frac{5\sqrt{g}}{6N_i} \rceil$ translations, where $g$ is the maximum size of the covariance matrix, as implemented in dl-find [22, 45], a rotation is performed again, until the dimer rotation converges again and a translation on the GPR-surface can be performed again. Overshooting is performed too, however only after a translation, and no separate dimension overshooting is applied because only the direction is relevant.

In contrast to the minimization internal 1, the back transformation causes trouble for the transition state search. Thus an alternative approach is presented in the next section, section 7.1.2, to handle this problem.

### 7.1.2 Predictive Entropy Search – GPRTS-PES

Instead of using a dimer rotation to get an accurate Hessian, here an alternative approach is presented. The idea is to gain as much information about the Hessian as possible with the least amount of training points. The main idea is to adapt the predictive entropy search, which was originally used for optimization processes [81]. In this section, a detailed description of how to use it for transition state search is given.

The entropy search, proposed by Henning and Schuler [82], uses an information-based acquisition function, based on earlier information theoretic suggestions [83]. It tries to maximize information about the functions' optimum (either minimum or maximum)

$$\alpha_{\text{ES}} = \mathbb{S}\left[p(\mathbf{x}_\star|\mathcal{D})\right] - \mathbb{E}_{p(y|\mathcal{D},\mathbf{x})}\left\{\mathbb{S}\left[p(\mathbf{x}_\star|\mathcal{D}\cup\{\mathbf{x},y\})\right]\right\} \tag{7.7}$$

The left term is the posterior's entropy with respect to the current data $\mathcal{D}$ of the maximizer or minimizer $\mathbf{x}_\star$. This is a constant for all values $\mathbf{x}$. The right term describes the same entropy, however under expectation, because all values of $y = f(\mathbf{x})$ are considered. This means maximizing $\alpha_{\text{ES}}$ is minimizing the posterior entropy after the new observation $\{\mathbf{x}, y\}$. The problem is, eq. (7.7) is hard to compute, because first of all $p(\mathbf{x}_\star|\mathcal{D}\cup\{\mathbf{x},y\}$ has to be evaluated for all values of $y$ and $\mathbf{x}$. Secondly, the entropy terms have no closed form, therefore there is no guarantee, that the posterior over the optimal value has an analytic form. However, Hernández-Lobato et al. [81] rewrote this equation

$$\begin{aligned}\alpha_{\text{ES}}(\mathbf{x}) &= \mathbb{S}\left[p(\mathbf{x}_\star|\mathcal{D})\right] - \mathbb{E}_{p(y|\mathcal{D},\mathbf{x})}\left\{\mathbb{S}\left[p(\mathbf{x}_\star|\mathcal{D}\cup\{\mathbf{x},y\}\right]\right\}\\ &= \mathbb{S}\left[p(y|\mathcal{D},\mathbf{x})\right] - \mathbb{E}_{p(\mathbf{x}_\star|\mathcal{D})}\left\{\mathbb{S}\left[p(y|\mathcal{D},\mathbf{x},\mathbf{x}_\star)\right]\right\}\\ &= \alpha_{\text{PES}}(\mathbf{x})\end{aligned} \tag{7.8}$$

To reformulate this equation, the concept of mutual information, introduced in section 5.4.2 has to be used [84]. The mutual information between $\mathbf{x}_\star$ and a random variable $Y$ is

$$\begin{aligned}I(\mathbf{x}_\star, Y) &= \mathbb{S}[\mathbf{x}_\star] - \mathbb{S}[\mathbf{x}_\star|Y]\\ &= \mathbb{S}[\mathbf{x}_\star] - \int p(y)\mathbb{S}[\mathbf{x}_\star|Y=y]\,\mathrm{d}y\\ &= \mathbb{S}[\mathbf{x}_\star] - \mathbb{E}_y[\mathbb{S}[\mathbf{x}_\star|Y=y]]\end{aligned} \tag{7.9}$$

If the distribution of $y$ is $p(y|\mathcal{D},\mathbf{x})$ and for $\mathbf{x}_\star$ is $p(\mathbf{x}_\star|\mathcal{D})$, eq. (7.7) is obtained. Since the mutual information is symmetric, the following reformulation can be made

$$\begin{aligned} I(\mathbf{x}_\star, Y) &= I(y, \mathbf{x}_\star) \\ &= \mathbb{S}[Y] - \mathbb{S}[Y|\mathbf{x}_\star] \\ &= \mathbb{S}[Y] - \int p(\mathbf{x})\mathbb{S}[Y|\mathbf{x}_\star = \mathbf{x}]\,\mathrm{d}\mathbf{x} \\ &= \mathbb{S}[Y] - \mathbb{E}_{\mathbf{x}_\star}[\mathbb{S}[Y|\mathbf{x}_\star]] \end{aligned} \tag{7.10}$$

with the distributions on $\mathbf{x}_\star$ and $y$ this leads to

$$\mathbb{S}[p(y|\mathcal{D},\mathbf{x})] - \mathbb{E}_{p(\mathbf{x}_\star|\mathcal{D}}[\mathbb{S}[p(y|\mathcal{D},\mathbf{x}_\star,\mathbf{x})]] \tag{7.11}$$

The key idea is to maximize the information about the transition state (or in the original papers maximum/minimum) $\mathbf{x}_\star$, which has the posterior distribution $p(\mathbf{x}_\star|\mathcal{D})$, i.e. select a new point that minimizes the uncertainty of the model. The current information is expressed in terms of the negative differential entropy, see eq. (5.73), of this distribution. Therefore, the point $\mathbf{x}_{n+1}$ is selected, which maximizes the expected reduction in this quantity. Thus, Henning and Schuler [82] introduced an acquisition function

$$\alpha(\mathbf{x}) = \mathbb{S}\left[p(\mathbf{x}_\star|\mathcal{D})\right] - \mathbb{E}_{p(y|\mathcal{D},\mathbf{x})}\left\{\mathbb{S}\left[p(\mathbf{x}_\star|\mathcal{D}\cup\{\mathbf{x},y\})\right]\right\} \tag{7.12}$$

where $\mathbb{S}[p(\mathbf{x})]$ is the differential entropy from eq. (5.73). The left term is the entropy of the transition state with respect to the current data, for any point $\mathbf{x}$ this is a constant. The right term is the same posterior entropy, however under expectation, because all possible $y$ are considered. In summary, the maximum of the acquisition function minimizes the entropy of the posterior after a new observation $\{y,\mathbf{x}\}$. However, there are two difficulties:

1. $p(\mathbf{x}_\star|\mathcal{D}\cup\{\mathbf{x},y\})$ has to be evaluated for different $\mathbf{x}$ and $y$.

2. the entropy calculations are not analytically.

To counter these difficulties Hernández-Lobato et al. [81] reformulated the acquisition function in eq. (7.10), where $p(y|\mathcal{D},\mathbf{x},\mathbf{x}_\star)$ is the posterior predictive distribution for $y$ given the data and $\mathbf{x}_\star$.

In other words: conditioning on $\mathbf{x}_\star$ pushes the posterior distribution up in the location of $\mathbf{x}_\star$ and down in regions away from $\mathbf{x}_\star$. Notice, that both terms are entropies of predictive distributions, which are analytically tractable or easier to approximate. The first term is the posterior marginal for $f(\mathbf{x})$, i.e.:

$$\mathbb{S}[p(y|\mathcal{D},\mathbf{x})] = \frac{1}{2}\log\left(2\pi\mathrm{e}\left(\mathbb{V}[\mathbf{x}] + \sigma_n^2\right)\right) \tag{7.13}$$

where the noise $\sigma_n^2$ has to be added, because $y$ are noisy observations. $e$ is Euler's number. The variance $\mathbb{V}[\mathbf{x}]$ is from eq. (5.67). The term $\mathbb{E}_{p(\mathbf{x}_\star|\mathcal{D})}[\mathbb{S}[p(y|\mathcal{D},\mathbf{x}_\star,\mathbf{x})]]$ has to be approximated. The expectation are taken by averaging over samples of $\mathbf{x}_\star^i$ drawn from $p(\mathbf{x}_\star|\mathcal{D})$. However, only collecting data with respect to the initial geometry is relevant in this case, so only one value, the starting geometry, has to be taken. The derivation of the following algorithm is based on [81].

### The Calculation of The Predictive Entropy Search

Now, the entropy $\mathbb{S}[p(y|\mathcal{D},\mathbf{x},\mathbf{x}_\star)]$ is approximated. Therefore, the distribution of this expression can be written as

$$p(y|\mathcal{D},\mathbf{x},\mathbf{x}_\star) = \int p(y|f(\mathbf{x})p(f(\mathbf{x})|\mathcal{D},\mathbf{x}_\star)\,\mathrm{d}f(\mathbf{x}) \tag{7.14}$$

The distribution $p(f(\mathbf{x})|\mathcal{D},\mathbf{x}_\star)$ is the posterior on $f(\mathbf{x})$ given the data and the location of the transition state of $f$, $\mathbf{x}_\star$. If the likelihood $p(y|f(\mathbf{x}))$ is Gaussian, $p(f(\mathbf{x})|\mathcal{D})$ is analytical because it is the predictive distribution of a GP, see eq. (5.41). However, constraints on $\mathbf{x}_\star$ are introduced and thus these constraints make $p(f(\mathbf{x})|\mathcal{D},\mathbf{x}_\star)$ intractable. The following constraints are introduced: $\mathbf{x}_\star$ should be a transition state, i.e. the Hessian at $\mathbf{x}_\star$ has one negative eigenvalues, while the others are positive. In addition, a diagonal Hessian is assumed to simplify the constraint for a transition state. Then the constraints are:

C1. $\nabla f(\mathbf{x}_\star) = \mathbf{0}$ and $\nabla^2_{i>j} f(\mathbf{x}_\star) = 0$

C2. $\nabla^2_{ii} f(\mathbf{x}_\star) = \begin{cases} < 0, & i = 1 \\ > 0, & i \neq 1 \end{cases}$

### The Constraint C1

The constraints are introduced into $p(f|\mathcal{D})$ to approximate $p(f|\mathcal{D},\mathbf{x}_\star)$. Therefore, the two random variables $\mathbf{c}$ and $\mathbf{z}$ are introduced

$$\mathbf{z} = \left(\nabla^2_{ii} f(\mathbf{x}_\star)\right) \qquad \mathbf{c} = \begin{pmatrix} \mathbf{y} \\ \nabla f(\mathbf{x}_\star) \\ \nabla^2_{i>j} f(\mathbf{x}_\star) \end{pmatrix} \tag{7.15}$$

To include constraint C1 the data are conditioned on "observation" $\nabla f(\mathbf{x}_\star) = \mathbf{0}$ and $\nabla^2_{i>j} f(\mathbf{x}_\star) = 0$. The random variable $\mathbf{c}$ enforces C1. With the input $\mathbf{x}$ and $\mathbf{x}_\star$, the covariance matrix $\mathbf{K}_{\text{PES}}$ is constructed, which contains the covariance evaluated on the vector $\begin{pmatrix}\mathbf{z}\\ \mathbf{c}\end{pmatrix}$. Those parts of $\mathbf{K}_{\text{PES}}$, which depend on $\mathbf{y}$, have to add the noise of the observed energy and gradients. The covariance matrix consists of three sub-matrices $\mathbf{K}_\mathbf{z}$, $\mathbf{K}_\mathbf{zc}$ and $\mathbf{K}_\mathbf{c}$. Thus, $\mathbf{K}$ can be written as

$$\mathbf{K}_{\text{PES}} = \begin{pmatrix} \mathbf{K}_\mathbf{z} & \mathbf{K}_\mathbf{zc} \\ \mathbf{K}_\mathbf{zc}^\top & \mathbf{K}_\mathbf{c} \end{pmatrix} \in \mathbb{R}^{[2d+nk+d\cdot(d-1)/2]\times[2d+nk+d\cdot(d-1)/2]} \tag{7.16}$$

where $d$ is the dimension of the system, i.e. $d = N_i$ and $nk$ is the size of the covariance matrix from eq. (5.65). These sub-matrices are calculated as follows:

$$\mathbf{K}_\mathbf{z} = \left(\text{cov}\left(\nabla^2_{ii} f(\mathbf{x}_\star, \nabla^2_{jj} f(\mathbf{x}_\star))\right)\right) = \frac{\partial^4 k(\mathbf{x}_\star,\mathbf{x}_\star)}{\partial^2 x_i^\star \partial^2 x_j^\star} \in \mathbb{R}^{d\times d} \tag{7.17}$$

$$\mathbf{K}_\mathbf{c} = \begin{pmatrix} \text{cov}(\mathbf{y},\mathbf{y}) & \text{cov}(\mathbf{y},\nabla f(\mathbf{x}_\star)) & \text{cov}(\mathbf{y},\nabla^2_{i>j} f(\mathbf{x}_\star)) \\ \text{cov}(\nabla f(\mathbf{x}_\star),\mathbf{y}) & \text{cov}(\nabla f(\mathbf{x}_\star),\nabla f(\mathbf{x}_\star)) & \text{cov}(\nabla f(\mathbf{x}_\star),\nabla^2_{i>j} f(\mathbf{x}_\star)) \\ \text{cov}(\nabla_{k>l} f(\mathbf{x}_\star),\mathbf{y}) & \text{cov}(\nabla_{k>l} f(\mathbf{x}_\star),\nabla f(\mathbf{x}_\star)) & \text{cov}(\nabla_{k>l} f(\mathbf{x}_\star),\nabla_{i>j} f(\mathbf{x}_\star)) \end{pmatrix} \tag{7.18}$$

and

$$\mathbf{K}_\mathbf{zc} = \text{cov}(\mathbf{z},\mathbf{c}) = \left(\text{cov}(\nabla^2_{ii} f(\mathbf{x}_\star,\mathbf{y}) \quad \text{cov}(\nabla^2_{ii} f(\mathbf{x}_\star,\nabla f(\mathbf{x}_\star)) \quad \text{cov}(\nabla^2_{ii} f(\mathbf{x}_\star,\nabla^2_{i>j} f(\mathbf{x}_\star))\right) \tag{7.19}$$

The individual covariances are computed as in eq. (5.64). With the conditioning on the observed values of $\mathbf{c}$, the first constraint can be included into

$$p(\mathbf{z}|\mathcal{D}, \mathrm{C1}) = p(\mathbf{z}|\mathbf{c}) = \mathcal{N}(\mathbf{z}|\mathbf{m}_0, \mathbf{V}_0) \tag{7.20}$$

Since both are Gaussian, the conditioning of two Gaussian are given as in eq. (5.22), and the resulting distribution is Gaussian with mean

$$\mathbf{m}_0 = \mathbf{K_{zc}}\mathbf{K_c}^{-1}\mathbf{c} \tag{7.21}$$

and covariance matrix

$$\mathbf{V}_0 = \mathbf{K_z} - \mathbf{K_{zc}}\mathbf{K_c}^{-1}\mathbf{K_{zc}}^\top \tag{7.22}$$

### Calculating Constraint C2 using Expectation Propagation

To incorporate constraint C2 a technique called Expectation Propagation (EP) [85] is used. The final algorithm is similar to Gaussian process regression for classification [10].

The second constraint is included as a product of one Gaussian and $d$ non-Gaussian factors

$$p(\mathbf{z}|\mathcal{D}, \mathrm{C1}, \mathrm{C2}) \sim \mathcal{N}(\mathbf{z}|\mathbf{m}_0, \mathbf{V}_0) \prod_{i=1}^{d} t_i(z_i) \tag{7.23}$$

This is applying Bayes' rule, where $p(\mathbf{z}|\mathcal{D}, \mathrm{C1}, \mathrm{C2})$ is the posterior, $p(\mathbf{z}|\mathbf{c}) = p(\mathbf{z}|\mathcal{D}, \mathrm{C1})$ the prior and $\prod_{i=1}^{d} t_i(z_i) = \prod_{i=1}^{d} p(\mathrm{C2}|z_i)$ the likelihood. Since the likelihood is not Gaussian, it makes the integration of the normalization, the marginalization of the likelihood, intractable. In the expectation propagation (EP) framework, the likelihood is approximated by local likelihoods, which are Gaussian functions in $z_i$:

$$\tilde{t}_i(z_i) = \mathcal{N}(z_i; \tilde{\mu}_i, \tilde{\sigma}_i^2) \tag{7.24}$$

Then the product of the local is a multivariate Gaussian

$$\prod_{i=1}^{d} \tilde{t}_i(z_i) = \mathcal{N}(\mathbf{z}|\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \tag{7.25}$$

where $\tilde{\boldsymbol{\mu}}$ is a vector of $\tilde{\mu}_i$ and $\tilde{\boldsymbol{\Sigma}}$ is diagonal with $\tilde{\Sigma}_{ii} = \sigma_i^2$. Now $p(\mathbf{z}|\mathcal{D}, \mathrm{C1}, \mathrm{C2})$ is approximate by a single multivariate Gaussian $q(\mathbf{z})$, i.e.

$$q(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim N(\mathbf{z}|\mathbf{m_0}, \mathbf{V}_0) \prod_{i=1}^{d} \mathcal{N}(z_i; \tilde{m}_i, \tilde{\sigma}_i^2) \tag{7.26}$$

which is parameterized, and comes from eq. (5.23), with mean

$$\boldsymbol{\mu} = \boldsymbol{\Sigma} \left( \tilde{\boldsymbol{\Sigma}}^{-1} \tilde{\boldsymbol{\mu}} + \mathbf{V}_0^{-1} \mathbf{m}_0 \right), \tag{7.27}$$

and the covariance matrix

$$\boldsymbol{\Sigma} = \left( \tilde{\boldsymbol{\Sigma}}^{-1} + \mathbf{V}_0^{-1} \right)^{-1}. \tag{7.28}$$

In other words $\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}$ are the local likelihood approximations and $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ are the approximated posterior parameters. The first idea would be to minimize the Kullback-Leibler divergence eq. (5.75) between the real posterior and its approximation: KL $(p(\mathbf{z}|\mathcal{D}, \mathrm{C1}, \mathrm{C2})||q(\mathbf{z}))$. However, direct minimization is intractable, instead each $\tilde{t}_i$ is updated sequentially. This is done by the following four steps:

1. Approximate the posterior by leaving out $\tilde{t}_i$ to form a marginal cavity distribution.
2. Get the marginal by combining cavity distribution with exact likelihood.
3. Approximate the non-Gaussian marginal by a Gaussian.
4. Compute the $\tilde{t}_i$ in such a way, that the posterior has the properties from 3.

These four steps are then repeated until convergence is reached. Therefore, the approximated posterior for $z_i$ contains:

1. prior $p(\mathbf{z}|\mathbf{c})$
2. local approximation $\tilde{t}_j$ with $i \neq j$
3. exact likelihood for the case $i$, $t_i(z_i) = \begin{cases} \chi(z_i < 0), & i = 1 \\ \chi(z_i > 0), & i \neq i \end{cases}$

Here $\chi_T(x)$ is the indicator function and is defined as

$$\chi_T(x) := \begin{cases} 1, & \text{if} \, x \in T \\ 0, & \text{if} \, x \notin T. \end{cases} \tag{7.29}$$

The parameters of $\tilde{t}_i$ are chosen such that the marginal posterior is as accurate as possible. Combining the prior and the local likelihood leads to

$$q_{-i}(z_i) \sim \int p(\mathbf{z}|\mathbf{c}) = \prod_{i \neq j} \tilde{t}_j(z_i)\,\mathrm{d}z_i \tag{7.30}$$

This is the combination of the prior and $d-1$ approximations, where the approximate likelihood $i$ is removed from the approximated posterior. Then $q(z_i) = \mathcal{N}(z_i|\mu_i, \sigma_i^2)$ is obtained from eq. (7.27) and eq. (7.28) by applying eq. (5.21). Since this contains one approximate too many,namely $\tilde{t}_i$, this factor has to be removed by dividing it by $\tilde{t}_i$ to get the cavity distribution

$$q_{-i}(z_i) \sim \frac{q(z_i)}{\tilde{t}_i(z_i)} = \mathcal{N}(z_i|\mu_{-i}, \sigma_{-i}^2) \tag{7.31}$$

with mean and variance

$$\begin{aligned} \mu_{-i} &= \sigma_{-i}^2 \left( \frac{\mu_i}{\sigma_i^2} - \frac{\tilde{\mu}_i}{\tilde{\sigma}_i^2} \right) \\ \sigma_{-i}^2 &= \frac{1}{\frac{1}{\sigma^2} - \frac{1}{\tilde{\sigma}_i^2}} \end{aligned} \tag{7.32}$$

Next, the new un-normalized Gaussian which approximates the product of the cavity distribution and the exact likelihood has to be found

$$\hat{q}(z_i) \sim q_{-i}(z_i) t_i(z_i) \tag{7.33}$$

If $q(x)$ is a Gaussian distribution, then from eq. (5.76) follows that the distribution $q(x)$ which minimizes KL$(p(x)||q(x))$ needs to match the first and second moment of $p(x)$.

To calculate the moments of eq. (7.33) the normalization constant

$$\hat{Z}_i = \int q_{-i}(z_i) t_(z_i)\,\mathrm{d}z_i = \int \mathcal{N}(z_i|\mu_{-i}, \sigma_{-i}^2) t_i(z_i)\,\mathrm{d}z_i \tag{7.34}$$

can be used. There are two cases for $i$, namely $i = 1$ and $i \neq 1$ and thus two calculations have to be performed to obtain the first two moments. However, they are similar in concept and differ only slightly. Therefore, only the first case $i = 1$ is handled in detail. The changes, which have to be made, to obtain the moments for the second case, are mentioned after the derivations of the moments for the first case.

## Calculation of The First and Second Moment

In the following, $i$ is equal to one, i.e. $i = 1$. First, the normalization constant is calculated directly

$$\hat{Z}_i = \int_{-\infty}^{\infty} \frac{\exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma_{-i}^2}\right)}{\sqrt{2\pi\sigma_{-i}^2}} \chi(z_i < 0)\,\mathrm{d}z_i = \frac{1}{\sqrt{2\pi\sigma_{-i}^2}} \int_{-\infty}^{0} \exp\left(-\frac{(z_i-\mu_{-i})}{2\sigma_{-i}^2}\right)\mathrm{d}z_i \tag{7.35}$$

Substituting $x = \frac{z_i-\mu_{-i}}{\sigma_{-i}}$ and setting $\alpha = -\frac{\mu_{-i}}{\sigma_{-i}}$ leads then to

$$\hat{Z}_i = \int_{-\infty}^{\alpha} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)\mathrm{d}x = \Phi(\alpha) \tag{7.36}$$

A similar result is obtained for the case $i \neq 1$, just setting $\alpha = \frac{\mu_{-i}}{\sigma_{-i}}$, and $\hat{Z}_i = \Phi(\alpha)$ has the same form.

To obtain an expression for the first moment, the derivative of $Z_{-i}$ with respect to $\mu_{-i}$ is calculated as

$$\begin{aligned}
\frac{\partial \hat{Z}_i}{\partial \mu_{-i}} &= \frac{1}{\sqrt{2\pi\sigma_{-i}^2}} \int_{-\infty}^{0} \frac{\partial}{\partial \mu_{-i}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma_{-1}^2}\right)\mathrm{d}z_i \\
&= \frac{1}{\sqrt{2\pi\sigma_{-1}^2}} \int_{-\infty}^{0} \frac{z_i-\mu_{-i}}{\sigma_{-1}^2} \exp\left(-\frac{(z_i-\mu_{-1})^2}{2\sigma_{-1}^2}\right)\mathrm{d}z_i \\
&= \underbrace{\int_{-\infty}^{0} \frac{z_i\sigma_{-i}^{-2}}{\sqrt{2\pi\sigma_{-i}^2}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma_{-i}^2}\right)\mathrm{dz_i}}_{=\frac{\hat{Z}_{-i}}{\sigma_{-i}^2}\mathbb{E}_{\hat{q}}[z_i]} - \underbrace{\int_{-\infty}^{0} \frac{\mu_{-i}\sigma_{-1}^{-2}}{\sqrt{2\pi\sigma_{-i}^2}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma_{-i}^2}\right)\mathrm{dz_i}}_{=\frac{\mu_{-i}}{\sigma_{-i}^2}\hat{Z}_{-i}} \\
&= \frac{\hat{Z}_{-i}\mathbb{E}_{\hat{q}}[z_i]}{\sigma_{-i}^2} - \frac{\mu_{-i}}{\sigma_{-i}^2}\hat{Z}_{-i} = \frac{\hat{Z}_{-i}}{\sigma_{-i}^2}\hat{\mu}_i - \frac{\mu_{-i}}{\sigma_{-i}^2}\hat{Z}_{-i}.
\end{aligned} \tag{7.37}$$

This leads to the expression for the first moment

$$\mathbb{E}_{\hat{q}}[z_i] = \hat{\mu}_i = \mu_{-i} + \sigma_{-i}^2 \frac{1}{Z_{-i}} \frac{\partial \hat{Z}_{-i}}{\partial \mu_{-i}}. \tag{7.38}$$

Further, the derivative of $\hat{Z}_{-i}$ from eq. (7.37) is calculated to obtain the mean of eq. (7.38):

$$\begin{aligned}\frac{\partial \hat{Z}_i}{\partial \mu_{-i}} &= \frac{1}{\sqrt{2\pi\sigma_{-i}^2}} \int\limits_{-\infty}^{0} \frac{z_i - \mu_{-i}}{\sigma_{-1}^2} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma_{-i}^2}\right) \mathrm{d}z_i \\ &= \frac{1}{\sigma_{-i}} \int\limits_{-\infty}^{\alpha} \frac{x}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x = -\frac{\varphi(\alpha)}{\sigma_{-i}}\end{aligned} \tag{7.39}$$

Inserting eq. (7.39) into eq. (7.38) results in

$$\mu_i = \mu_{-i} + \sigma_{-i}\frac{\varphi(\alpha)}{\Phi(\alpha)}. \tag{7.40}$$

For the second moment the derivative of $\hat{Z}_{-i}$ with respect to $\sigma^2_{-i}$ has to be calculated

$$\begin{aligned}
\frac{\partial \hat{Z}_{-i}}{\partial \sigma^2_{-i}} &= \frac{\partial}{\partial \sigma^2_{-i}} \frac{1}{\sqrt{2\pi\sigma^2_{-i}}} \int_{-\infty}^{0} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma^2_{-i}}\right) \mathrm{d}z_i \\
&+ \frac{1}{\sqrt{2\pi\sigma^2_{-i}}} \int_{-\infty}^{0} \frac{\partial}{\partial \sigma^2_{-i}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma^2_{-i}}\right) \mathrm{d}z_i \\
&= -\frac{1}{2\sigma^2_{-i}} \underbrace{\int_{-\infty}^{0} \frac{1}{\sqrt{2\pi\sigma^2_{-i}}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma^2_{-i}}\right) \mathrm{d}z_i}_{=\hat{Z}_{-i}} \\
&+ \frac{1}{2\sigma^4_{-i}} \int_{-\infty}^{0} \frac{(z_i-\mu_{-i})^2}{\sqrt{2\pi\sigma^2_{-i}}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma^2_{-i}}\right) \mathrm{d}z_i \\
&= -\frac{\hat{Z}_{-i}}{2\sigma^2_{-i}} + \frac{1}{2\sigma^4_{-i}} \int_{-\infty}^{0} \frac{z_i^2}{\sqrt{2\pi\sigma^2_{-i}}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma^2_{-i}}\right) \mathrm{d}z_i \\
&+ \frac{1}{2\sigma^4_{-i}} \int_{-\infty}^{0} \frac{2\mu_{-i}z_i}{\sqrt{2\pi\sigma^2_{-i}}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma^2_{-i}}\right) \mathrm{d}z_i \\
&+ \frac{1}{2\sigma^4_{-i}} \int_{-\infty}^{0} \frac{\mu^2_{-i}}{\sqrt{2\pi\sigma^2_{-i}}} \exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma^2_{-i}}\right) \mathrm{d}z_i \\
&= -\frac{\hat{Z}_{-i}}{2\sigma^2_{-i}} + \frac{\hat{Z}_{-i}}{2\sigma^4_{-i}} \mathbb{E}_{\hat{q}}[z_i^2] - \frac{\mu_{-i}\hat{Z}_{-i}}{\sigma^4_{-i}} \mathbb{E}_{\hat{q}}[z_i] + \frac{\mu^2_{-i}\hat{Z}_{-i}}{2\sigma^4_{-i}}
\end{aligned} \tag{7.41}$$

Reshaping for the second moment and inserting eq. (7.38) for the first moment leads to

$$\mathbb{E}_{\hat{q}}[z_i^2] = \sigma^2_{-i} + \mu^2_{-i} - \frac{2\sigma^2_{-i}\mu_{-i}}{\hat{Z}_{-i}} \frac{\partial \hat{Z}_{-i}}{\partial \mu_{-i}} + \frac{2\sigma^4_{-i}}{\hat{Z}_{-i}} \frac{\partial \hat{Z}_{-i}}{\partial \sigma^2_{-i}} \tag{7.42}$$

This can be used, again with eq. (7.38) for the first moment, to calculate the variance

$$\begin{aligned}\mathbb{V}_{\hat{q}}[z_i] = \mathbb{E}_{\hat{q}}[z_i^2] - \mathbb{E}_{\hat{q}}[z_i]^2 &= \sigma_{-i}^2 + \mu_{-i}^2 - \frac{2\sigma_{-i}^2\mu_{-i}}{\hat{Z}_{-i}}\frac{\partial \hat{Z}_{-i}}{\partial \mu_{-i}} + \frac{2\sigma_{-i}^4}{\hat{Z}_{-i}}\frac{\partial \hat{Z}_{-i}}{\partial \sigma_{-i}^2} \\ &- \mu_{-i}^2 + \frac{2\sigma_{-i}^2\mu_{-i}}{\hat{Z}_{-i}}\frac{\partial \hat{Z}_{-i}}{\partial \mu_{-i}} - \frac{\sigma_{-i}^4}{\hat{Z}_{-i}^2}\left(\frac{\partial \hat{Z}_{-i}}{\partial \mu_{-i}}\right)^2 \\ &= \sigma_{-i}^2 - \sigma_{-i}^4\left[\frac{1}{\hat{Z}_{-i}^2}\left(\frac{\partial \hat{Z}_{-i}}{\partial \mu_{-i}}\right)^2 - \frac{2}{\hat{Z}_{-i}}\frac{\partial \hat{Z}_{-i}}{\partial \sigma_{-i}^2}\right]\end{aligned} \tag{7.43}$$

Having an expression for the variance, the derivative of $\hat{Z}_{-i}$ with respect to $\sigma_{-i}^2$ needs to be calculated

$$\begin{aligned}\frac{\partial \hat{Z}_{-i}}{\partial \sigma_{-i}^2} &= -\frac{1}{2\sigma_{-i}^2}\Phi(\alpha) + \frac{1}{2\sigma_{-i}^4}\int\limits_{-\infty}^{0}\frac{(z_i-\mu_{-i})^2}{\sqrt{2\pi\sigma_{-i}^2}}\exp\left(-\frac{(z_i-\mu_{-i})^2}{2\sigma_{-i}^2}\right)\mathrm{d}z_i \\ &= -\frac{1}{2\sigma_{-i}^2}\Phi(\alpha) + \frac{1}{2\sigma_{-i}^2}\int\limits_{-\infty}^{\alpha}\frac{x^2}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)\mathrm{d}x \\ &= -\frac{1}{2\sigma_{-1}^2}\Phi(\alpha) + \frac{1}{2\sigma_{-i}^2}\left[\frac{1}{2}\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right) - \frac{x}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)\right]_{-\infty}^{\alpha} \\ &= -\frac{1}{2\sigma_{-i}^2}\Phi(\alpha) + \frac{1}{2\sigma_{-i}^2}\left(\frac{1}{2}\mathrm{erf}\left(\frac{\alpha}{\sqrt{2}}\right) - \alpha\varphi(\alpha) + \frac{1}{2}\right) \\ &= -\frac{1}{2\sigma_{-i}^2}\Phi(\alpha) + \frac{1}{2\sigma_{-i}^2}\Phi(\alpha) - \frac{\alpha}{2\sigma_{-i}^2}\varphi(\alpha) = -\frac{\alpha}{2\sigma_{-i}^2}\varphi(\alpha)\end{aligned} \tag{7.44}$$

This is now inserted into eq. (7.38):

$$\hat{\mu}_i = \mu_{-i} - \sigma_{-i}\frac{\varphi(\alpha)}{\Phi(\alpha)} \tag{7.45}$$

and into eq. (7.43)

$$\hat{\sigma}_i^2 = \sigma_{-i}^2\left(1 - \frac{\varphi(\alpha)}{\Phi(\alpha)}\left[\frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha\right]\right) \tag{7.46}$$

With these calculated moments, the contribution of the cavity distribution can be removed and the parameters of the approximation $\tilde{t}_i$ can be calculated with the following equation

$$\tilde{\mu}_i = \tilde{\sigma}_i^2 \left( \frac{\hat{\mu}_i}{\hat{\sigma}_i^2} - \frac{\mu_{-i}}{\sigma_{-i}^2} \right) \tag{7.47}$$

$$\tilde{\sigma_i}^2 = \frac{1}{\frac{1}{\hat{\sigma}_i^2} - \frac{1}{\sigma_{-i}^2}} \tag{7.48}$$

First, the variance of $\tilde{t}_i$ is calculated

$$\begin{aligned}\tilde{\sigma}_i^2 &= \left[ \frac{1}{\mathbb{V}_{\hat{q}}} - \frac{1}{\sigma_{-i}^2} \right]^{-1} = \left[ \frac{1}{\sigma_{-i}^2 \left( 1 - \frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right] \right)} - \frac{1}{\sigma_{-i}^2} \right]^{-1} \\ &= \sigma_{-i}^2 \frac{1}{\frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right]} = \frac{\sigma_{-i}^2}{\frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right]} - \sigma_{-i}^2 = a^{-1} - \sigma_{-i}^2\end{aligned} \tag{7.49}$$

where $a^{-1} = \frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right] \sigma_{-i}^{-2}$. Then same result is obtaind for $i \neq 1$, but with $\alpha = \frac{\mu_{-i}}{\sigma_{-i}}$. Then the mean can be calculated

$$\begin{aligned}\tilde{\mu}_i &= \tilde{\sigma}_i^2 \left( \frac{\hat{\mu}_i}{\hat{\sigma}_i^2} - \frac{\mu_{-i}}{\sigma_{-i}^2} \right) \\ &= \frac{\sigma_{-i}^2 \left( 1 - \frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right] \right)}{\frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right]} \frac{\mu_{-i} - \sigma_{-i} \frac{\varphi(\alpha)}{\Phi(\alpha)} - \mu_{-i} \left( 1 - \frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right] \right)}{\sigma_{-i}^2 \left( 1 - \frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right] \right)} \\ &= \frac{\frac{\varphi(\alpha)}{\Phi(\alpha)} \left( -\mu_{-i} + \sigma_{-i} \left( \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right) \right)}{\frac{\varphi(\alpha)}{\Phi(\alpha)} \left[ \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right]} = \mu_{-i} - \frac{\sigma_{-i}}{\frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha} = \mu_{-i} + b^{-1}\end{aligned} \tag{7.50}$$

with $b^{-1} = -\left( \frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha \right) \sigma_{-i}^{-1}$. A similar result can be obtained for $i \neq 1$. The only difference is

$$b^{-1} = \frac{\frac{\varphi(\alpha)}{\Phi(\alpha)} + \alpha}{\sigma_{-i}} \tag{7.51}$$

and $\alpha = \frac{\mu_{-i}}{\sigma_{-i}}$. The rest remains the same.

**Making Predictions Based on C1 and C2**

To make prediction about $f(\mathbf{x})$ with some input $\mathbf{x}$ the vector $\mathbf{f} = \begin{pmatrix} f(\mathbf{x}) \\ f(\mathbf{x}_\star) \end{pmatrix}$ is introduced. The distribution for this vector given the constraints C1 and C2 is expressed as

$$p(\mathbf{f}|\mathcal{D}, \mathrm{C1}, \mathrm{C2}) \approx \int p(\mathbf{f}|\mathbf{z}, \mathbf{c}) q(\mathbf{z})\, \mathrm{d}\mathbf{z} = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}_\mathbf{f} \boldsymbol{\Sigma}_\mathbf{f}) \tag{7.52}$$

Here, $\mathbf{f}$ is assumed to be independent of C2 given $\mathbf{z}$. Therefore, the integral is over a product of two Gaussians. By introducing the following covariances

$$\mathbf{K}_\dagger = \operatorname{cov}\left(\mathbf{f}, \begin{pmatrix} \mathbf{z} \\ \mathbf{c} \end{pmatrix}\right) \tag{7.53}$$

$$\mathbf{K}_\mathbf{f} = \operatorname{cov}\left(\mathbf{f}, \mathbf{f}\right) \tag{7.54}$$

the posterior of the above is Gaussian distribution with mean

$$\boldsymbol{\mu}_\mathbf{f} = \mathbf{K}_\dagger \left(\mathbf{K}_{\mathrm{PES}} + \tilde{\mathbf{W}}\right)^{-1} \begin{pmatrix} \mathbf{c} \\ \tilde{\boldsymbol{\mu}} \end{pmatrix} \tag{7.55}$$

and covariance

$$\boldsymbol{\Sigma}_\mathbf{f} = \mathbf{K}_\mathbf{f} - \mathbf{K}_\dagger \left(\mathbf{K}_{\mathrm{PES}} + \tilde{\mathbf{W}}\right)^{-1} \mathbf{K}_\dagger^\top \tag{7.56}$$

where $\tilde{\mathbf{W}}$ is a diagonal matrix

$$\tilde{\mathbf{W}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{\Sigma}} \end{pmatrix} \tag{7.57}$$

because $\tilde{\boldsymbol{\Sigma}}$ is a diagonal matrix.

The first entry of eq. (7.56) corresponds to the marginal variance of $f(\mathbf{x})$ and is used as variance in the Gaussian approximation of $p(f(\mathbf{x})|\mathcal{D}, \mathbf{x}_\star)$. Only the variance is needed because the entropy of a Gaussian is not dependent on the first moment. Therefore the following notation is introduced:

$$\mathbb{V}[\mathbf{x}|\mathbf{x}_\star] = \left[\boldsymbol{\Sigma}_\mathbf{f}\right]_{11}. \tag{7.58}$$

### The Full Acquisition Function

With eq. (7.58) the left entropy term in eq. (7.10) can be calculated with eq. (7.58), which is the variance of a Gaussian

$$\mathbb{S}[p(y|\mathcal{D},\mathbf{x},\mathbf{x}_\star)] = \frac{1}{2}\log\left[2\pi\mathrm{e}\left(\mathbb{V}[\mathbf{x}|\mathbf{x}_\star] + \sigma_e^2\right)\right] \tag{7.59}$$

Now the complete acquisition function from eq. (7.10) is

$$\alpha(\mathbf{x}) = \frac{1}{2}\log\left(\mathbb{V}[\mathbf{x}] + \sigma_e^2\right) - \frac{1}{2}\log\left(\mathbb{V}[\mathbf{x}|\mathbf{x}_\star] + \sigma_e^2\right) = \frac{1}{2}\log\left(\frac{\mathbb{V}[\mathbf{x}] + \sigma_e^2}{\mathbb{V}[\mathbf{x}|\mathbf{x}_\star] + \sigma_e^2}\right) \tag{7.60}$$

### The Implementation of the Acquisition Function

Due to numerical stability, care has to be taken when implementing the EP factors and calculating the predictive variance in eq. (7.58). Although $\tilde{\sigma}_i^2 \to \infty$ is possible, everything remains well defined, which corresponds to ignoring the corresponding likelihood. Therefore, new parameters for numerical stability have to be introduced.

$$\tilde{\tau}_i = \tilde{\sigma}_i^{-2}, \quad \tilde{\mathbf{S}} = \mathrm{diag}(\tilde{\tau}), \quad \tilde{\nu} = \tilde{\mathbf{S}}\tilde{\mu}, \quad \tau_{-i} = \sigma_{-i}^{-2}, \quad \nu_{-i} = \tau_{-i}\mu_i \tag{7.61}$$

rather than $\tilde{\sigma}_i, \tilde{\mu}, \sigma_{-i}^2, \mu_{-i}$. The important matrix is now

$$\mathbf{A} = \mathbf{I} + \tilde{\mathbf{S}}^{1/2}\mathbf{V}_0\tilde{\mathbf{S}}^{1/2} \tag{7.62}$$

Every expression involving $\mathbf{A}^{-1}$ can be efficiently calculated via Cholesky decomposition. Then the parameters for the eq. (7.26) can be computed as

$$\begin{aligned}
\mathbf{\Sigma} &= \left(\tilde{\boldsymbol{S}} + \mathbf{V}_0\right)^{-1} = \mathbf{V}_0 - \mathbf{V}_0\left(\tilde{\mathbf{S}} + \mathbf{V}_0\right)^{-1}\mathbf{V}_0 \\
&= \mathbf{V}_0 - \mathbf{V}_0\left(\tilde{\mathbf{S}} + \tilde{\mathbf{S}}^{-1/2}\left(\mathbf{A} - \mathbf{I}\right)\tilde{\mathbf{S}}^{-1/2}\right)^{-1}\mathbf{V}_0 \\
&= \mathbf{V}_0 - \mathbf{V}_0\left(\tilde{\mathbf{S}}^{-1/2}\mathbf{A}\tilde{\mathbf{S}}^{-1/2}\right)^{-1}\mathbf{V}_0 \\
&= \mathbf{V}_0 - \mathbf{V}_0\tilde{\mathbf{S}}^{1/2}\mathbf{A}^{-1}\tilde{\mathbf{S}}^{1/2}\mathbf{V}_0
\end{aligned} \tag{7.63}$$

After calculating the update for the side parameters, these new parameters have to be taken into account for eq. (7.25). Thus, the inverse new covariance matrix is

$$\mathbf{\Sigma}_{\text{new}}^{-1} = \mathbf{V}_0^{-1} + \tilde{\mathbf{S}}_{\text{old}} + \left(\tilde{\tau}_i^{\text{new}} - \tilde{\tau}_i^{\text{old}}\right) \mathbf{e}_i \mathbf{e}_i^\top \tag{7.64}$$

Using the Woodburry identity [86] the new covariance is computed as

$$\mathbf{\Sigma}^{\text{new}} = \mathbf{\Sigma}^{\text{old}} - \frac{\tilde{\tau}_i^{\text{new}} - \tilde{\tau}_i^{\text{old}}}{1 + \left(\tilde{\tau}_i^{\text{new}} - \tilde{\tau}_i^{\text{old}}\right) \Sigma_{ii}^{\text{old}}} \mathbf{s}_i \mathbf{s}_i^\top \tag{7.65}$$

where $\mathrm{s}_i$ is the $i$-th column vector of $\mathbf{\Sigma}^{\text{old}}$. The new mean can then be calculated from eq. (7.27). The pseudo-code can be shown in listing 7.1.

**Listing 7.1:** Pseudocode for EP factors.

```
do while (not converged):
  do i=1,d
     t_bar(i) = 1/v[i][i] - tilde_t[i]
     v_bar[i] = m/v[i][i] - tilde_v[i]
     call subroutine calculate_moments(Var_q, E_q)
     delta_tau = 1/Var_q-t_bar[i]-tilde_t[i]
     tilde_t[i] = tilde_t[i]+delta_tau
     tilde_v[i] = E_q/Var_q - v_bar[i]
     v = v-1/(1/delta_tau-v[i][i])*s[i]*s_t[i]
     m= V*(tilde_t[i]/tilde_v[i])+V_0_inverse*m_0
  enddo
  L=Cholesky(A)    !A:=I_n +S_half*K*S_half
  R= L_transpose\S_half*K
  V = V_0-R_transpose*R
  m = V*tilde_t + V*V_0_inverse*m_0
enddo
```

Before continuing, the Woodburry-identity [86] has to be introduced:

$$\left[Z + UWV^\top\right]^{-1} = Z^{-1} - Z^{-1}U\left(W^{-1} + V^\top Z^{-1}U\right)^{-1} V^\top Z^{-1} \tag{7.66}$$

This looks similar to $\left[\mathbf{K}+\tilde{\mathbf{W}}\right]^{-1}$. In order to calculate $\tilde{\mathbf{W}}$, the vector

$$\mathbf{U}=\begin{pmatrix}\mathbf{0}\\ \mathbf{I}_d\end{pmatrix} \tag{7.67}$$

is introduced, where $d+d(d-1)/2+nk$ zeros are included. Then with the help of eq. (7.66) and eq. (7.61) can be written as

$$\left[\mathbf{K}_{\text{PES}}+\mathbf{U}\tilde{\mathbf{S}}^{-1}\mathbf{U}^\top\right]^{-1}=\mathbf{K}_{\text{PES}}^{-1}-\mathbf{K}_{\text{PES}}^{-1}\mathbf{U}\left[\tilde{\mathbf{S}}+\mathbf{U}^\top\mathbf{K}_{\text{PES}}^{-1}\mathbf{U}\right]^{-1}\mathbf{U}^\top\mathbf{K}_{\text{PES}}^{-1} \tag{7.68}$$

With this equation, the problem $\tilde{\sigma}_i^2\to\infty$ can be avoided. Therefore, eq. (7.56) can be rewritten as

$$\mathbf{\Sigma_f}=\mathbf{k_f}-\mathbf{k}_\dagger\left(\mathbf{K}_{\text{PES}}^{-1}\mathbf{k}_\dagger^\top-\mathbf{C}\mathbf{K}_{\text{PES}}^{-1}\mathbf{k}_\dagger^\top\right) \tag{7.69}$$

where $\mathbf{C}$ is calculated as

$$\mathbf{C}=\mathbf{K}_{\text{PES}}^{-1}\mathbf{U}\left(\tilde{\mathbf{S}}+\mathbf{U}^\top\mathbf{K}_{\text{PES}}^{-1}\mathbf{U}\right)^{-1}\mathbf{U}^\top \tag{7.70}$$

Every occurrence of the type $\mathbf{a}^{-1}\mathbf{b}$ can be efficiently solved by Cholesky decomposition. That is the reason why the matrix $\mathbf{C}$ is introduced to avoid the direct inversion of $\mathbf{K}_{\text{PES}}$ in eq. (7.68).

### 7.1.3 The GPRTS Algorithm

Starting with an initial geometry, probably close to a real transition state, the energy and gradient of this configuration are calculated. The gradient and the initial geometry are then transformed into internal coordinates, which are used to build the GPR surrogate surface. Since only one training point, when trained on energy and gradients, gives a poor prediction for the Hessian, a second point has to be selected. The Hessian for the initial geometry is zero and thus no a transition state search starting from this point is possible on the surrogate surface. Selecting the next point is done randomly. Therefore, $d=N_i$ random values are drawn from a standard Gaussian distribution. The resulting vector is then scaled up to the dimer length, for both internal and Gaussian process regression for transition state search based on predictive entropy search (GPRTS-PES). This random point is transformed back to Cartesian coordinates and new energy and gradient are calculated. Then, the GPR surface is built with two points.

To describe the Hessian around the initial geometry as accurately as possible, new points are selected according to a dimer rotation (GPR internal) or maximize the acquisition function eq. (7.60). For GPR internal, where the dimer is initially built from the initial geometry as the midpoint and the random point as the endpoint, the dimer rotation from section 3.2.1 is performed. GPRTS-PES uses the maximum of the acquisition function, which is then scaled up (or down) to the dimer length. The new geometry is again back transformed to get new energy and gradient. The dimer rotation for GPR internal is converged if the cosine of the angle from eq. (7.6) is larger than $1 - 10^{-3}$. While for GPRTS-PES the Hessian is accurate enough if the change in the eigenvalues of the Hessian is smaller than $10^{-3}$. If both point selection methods are converged, a translation on the surrogate surface to find a saddle point is performed. This is done by the P-RFO algorithm from section 3.2.2. The P-RFO is sopped on the surrogate either if two times the maximum step is reached, the variance of starting point plus step is smaller than $10^{-4}$ or if P-RFO is converged with eigenvalue smaller than zero and the gradient equal or below $10^{-2} \cdot \eta$, where $\eta$ is from eq. (6.2). This is then the prediction for the new geometry and is transformed back to again get a new energy and gradient.

The translation via P-RFO is repeated either if the global convergence criteria are fulfilled or $\lceil 5\sqrt{g}/(6d) \rceil$, where $g$ is the covariance matrix size, translations occurred. In the last case again either the dimer is rotated or the acquisition function eq. (7.60) is maximized, this time with respect to the latest point after the translation. The whole procedure is then repeated until the global convergence criteria are fulfilled. A schematic representation of the described algorithm for GPRTS-PES is shown in fig. 7.1.

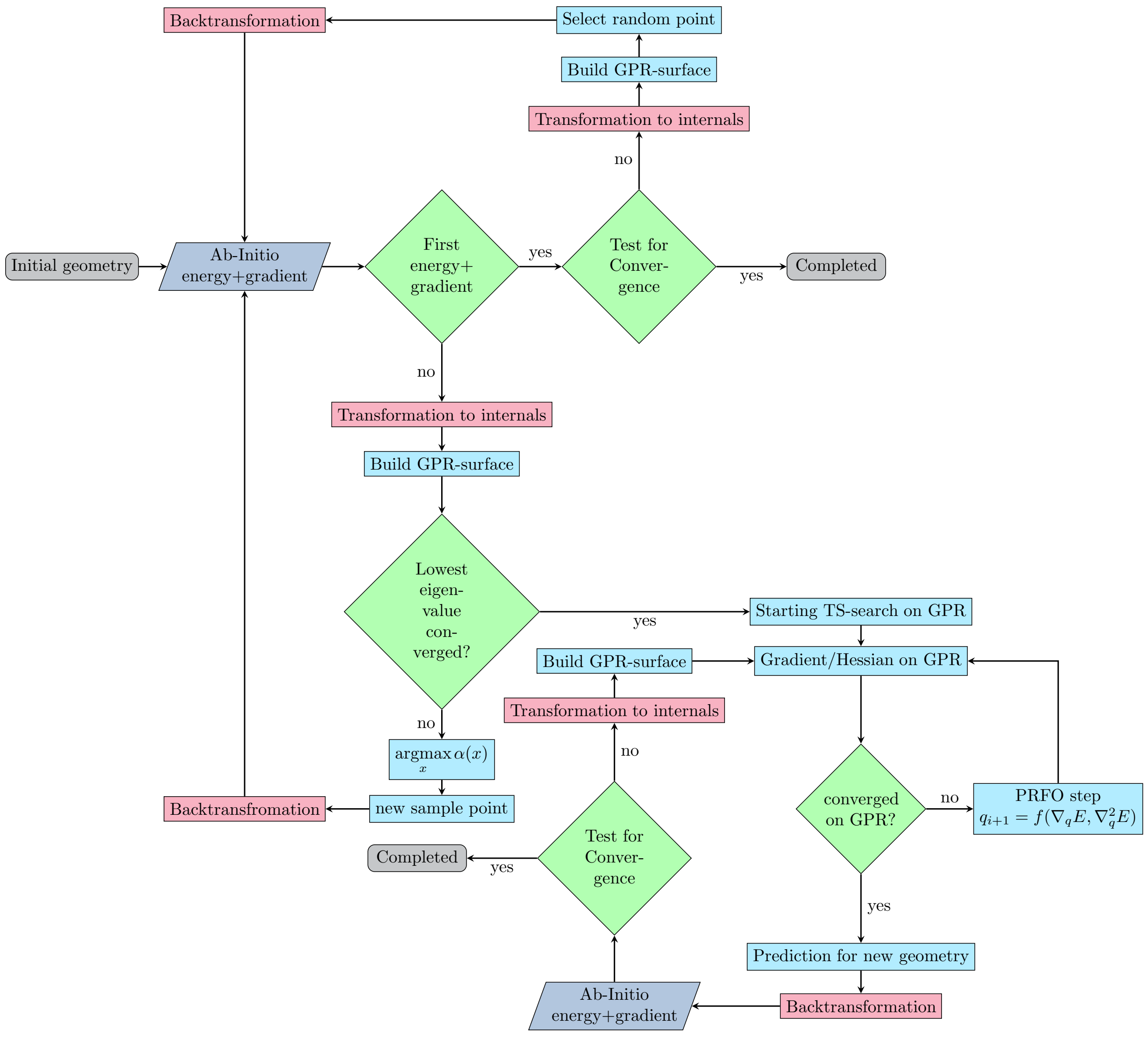


**Figure 7.1:** Schematic representation of the algorithm for transition-state search using GPR and predictive entropy search. This scheme can be also used for GPRTS internal, just replace the maximization of the acquisition function $\alpha(\mathbf{x})$ by a dimer rotation on the GPR surface and adapt the convergence criterion for the rotation.

## 7.2 Results

Before using a real-world example, a two-dimensional toy problem to demonstrate how the GPRTS-PES algorithm works is presented in section 7.2.1.

Afterwards, the transition state search algorithms presented beforehand, or search for a saddle point of first order, are tested for 24 molecules and compared to the dimer method. The energy and gradients were evaluated on the DFT level with BP86 [68] [69] as functional and def2-SVP as basis set [70]. The dispersion correction was used [87]. As the underlying DFT program TURBOMOLE [71] was used interfaced to DL-FIND via ChemShell [72]. The default convergence criteria from DL-FIND were used, as for the GPR minimizer, see section 6.2. Additionally, an energy convergence criterion is used from eq. (7.1). The results are shown in section 7.2.2.

Finally, the hyperparameters are investigated in section 7.2.3. There, the independence of the marginal likelihood from length scale $\ell$ and the gradient noise $\sigma_g$ is shown. Thus, the two hyperparameters are investigated separately.

### 7.2.1 Two-Dimensional Test-System

To demonstrate the GPRTS-PES algorithm a two-dimensional test system was chosen. The functional form is

$$f(x,y) = -20\exp\left[-\frac{x^2+y^2}{5\sqrt{2}}\right] - \exp\left[\frac{1}{2}\left(\cos(2\pi x)+\cos(2\pi y)\right)\right] + e + 20 \qquad (7.71)$$

This is the so-called Ackley function [88] [89]. In the square $[-1,1]^2$ the function is plotted in fig. 7.2. The Ackley function has a lot of stationary points, i.e. $\nabla f(x,y) = 0$. Thus it is a well suited test function to find the nearest saddle point.

As a starting point $\mathbf{x}_{\text{start}} = (0.66666666, 0.2)$ was chosen indicated with a bold $\star$ in fig. 7.2, which lies close to the nearest saddle point $\mathbf{x}_{\text{ts}} = (0.62641, 0.0)$. GPRTS-PES requires 11 total function evaluations, where 7 are for the translation. This means with the starting point and the randomly selected point, two maximizations of the acquisition function $\alpha(x,y)$ are required.

The plot of $\alpha(x,y)$ for two training points can be seen in fig. 7.3, where for better visibility the area was zoomed in between the interval $[0.5, 0.7] \times [0.15, 0.25]$. There are two maxima, both pointing to two near saddle points. The nearest one, $\mathbf{x}_{\text{ts}}$, has a slightly larger value of $\alpha(x,y)$. If optimized properly, this point is selected as the maximum of the acquisition function. Then this direction is upscaled to the maximum dimer length.

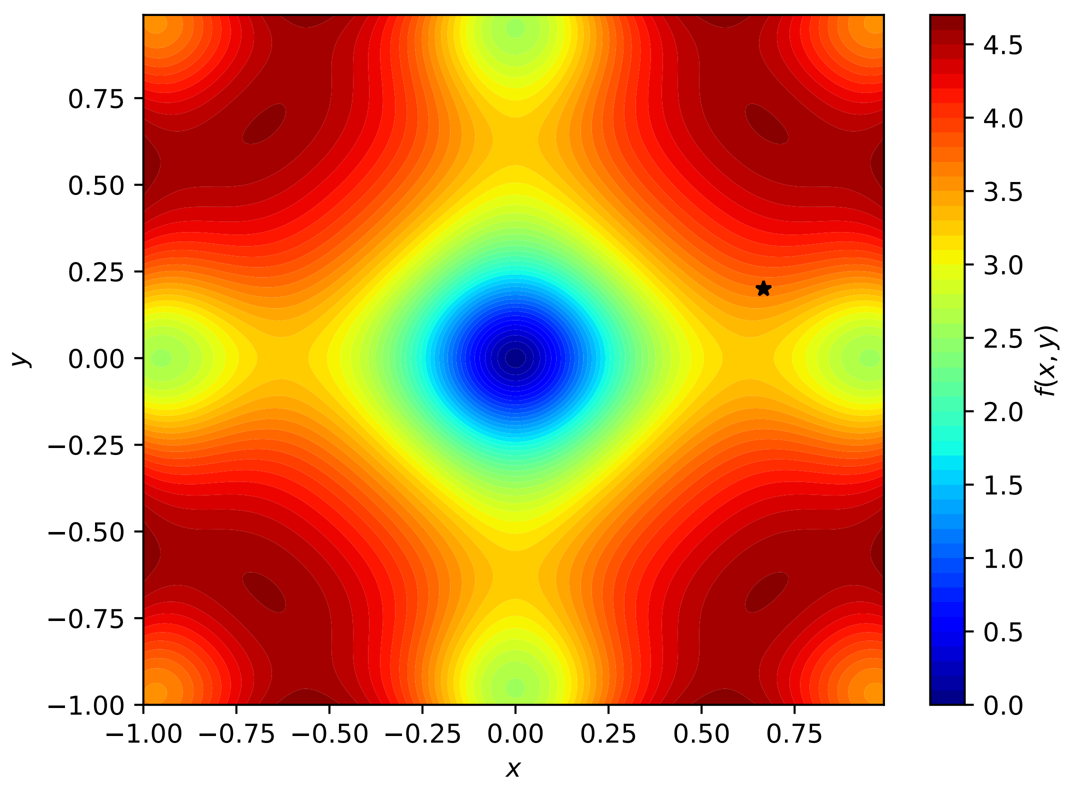


**Figure 7.2:** Two-dimensional test system, where the $\star$ indicates the starting point for a transition state search.

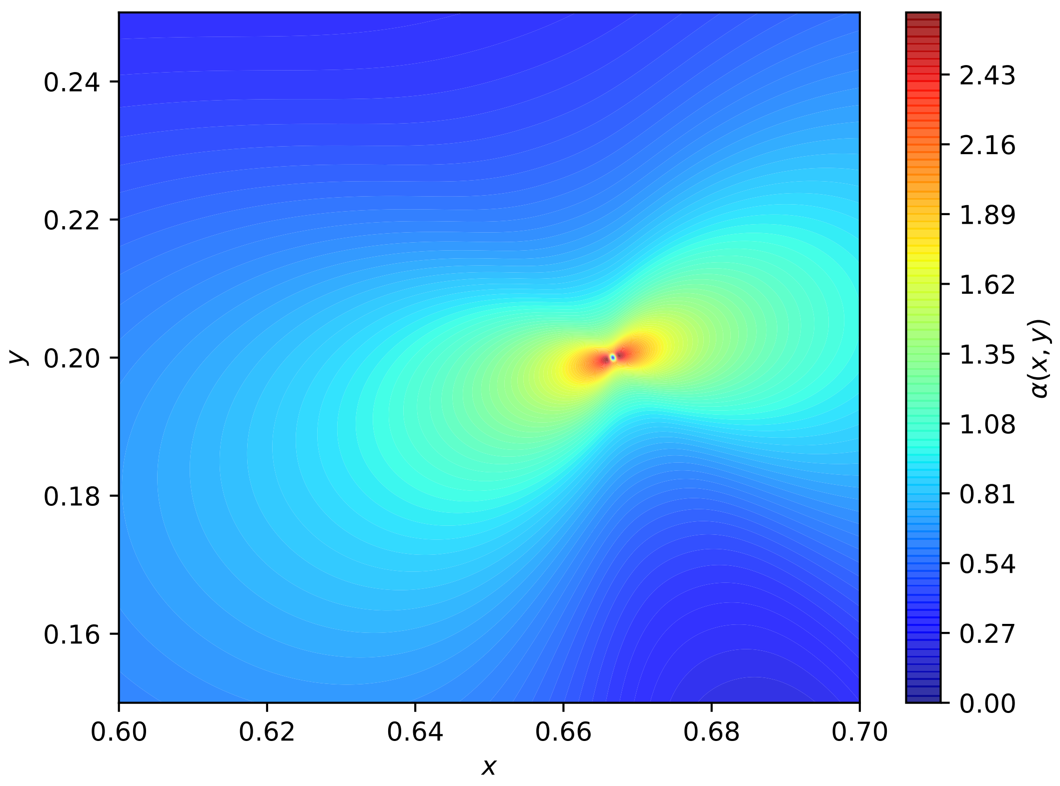


**Figure 7.3:** The acquisition function $\alpha(x, y)$ plotted with two training points for the Ackley function plotted in fig. 7.2. For better visibility zoomed in.

GPRTS in contrast required in total 18 function evaluations to reach the saddle point, from which 12 are for translation. Thus, three dimer rotations on the GPR surrogate are necessary. Therefore 12 translations are necessary to converge to the saddle point. This leads to the following conclusion:

- GPRTS-PES requires fewer sample points, here one, to describe the Hessian at least in the direction of the saddle point better than GPRTS.
- The Hessian is more accurately described by GPRTS-PES, resulting in fewer function evaluations compared to GPRTS.

Overall the two dimensional test case indicates a slightly better preformance of GPRTS-PES compared to GPRTS.

### 7.2.2 Efficiency

The baker test set [8] has been used for testing the transition state search algorithm. However, system 15 from that test set is removed, because the dlc code [9] in dl-find [45] can not create internal coordinates for this molecule. The starting geometries of the molecules used are shown in fig. 7.4. The GPR transition state search is compared to the dimer method implemented in dl-find [39]. As for the GPR minimizer, a box plot to represent the distribution of required energy and gradient evaluation is used and is shown in fig. 7.5. Since the GPR in internal coordinates and GPRTS-PES perform significantly better than the other optimizer/coordinate system combinations, a separate box plot is shown in fig. 7.6 to allow a better comparison between GPRTS internal and GPRTS-PES. In contrast to the minimization, the number of energy and gradient evaluations instead of the number of steps is compared. The reason lies within the way the dimer method handles steps and energy/gradient evaluation. For the dimer method, only translations are considered as a step. The number of energy/gradient evaluations is listed in table 7.2.

The statistical data for fig. 7.5 and fig. 7.6 are shown in table 7.1. In the box plots fig. 7.5 and fig. 7.6 the orange bar is the median. Outliers above the third quartile are identified if the amount of energy and gradient evaluations is above $Q_3 + 1.5 \cdot \mathrm{IQR}$ and are represented as blue circles. The red diamond is the (arithmetic) mean. The quartiles, outliers, and the mean are defined as in the GPR minimization case from section 6.2.1.

Both saddle point search algorithms, GPR and dimer reduce the amount of energy and gradient calculations if the Cartesian coordinates are replaced by internal coordinates. However,

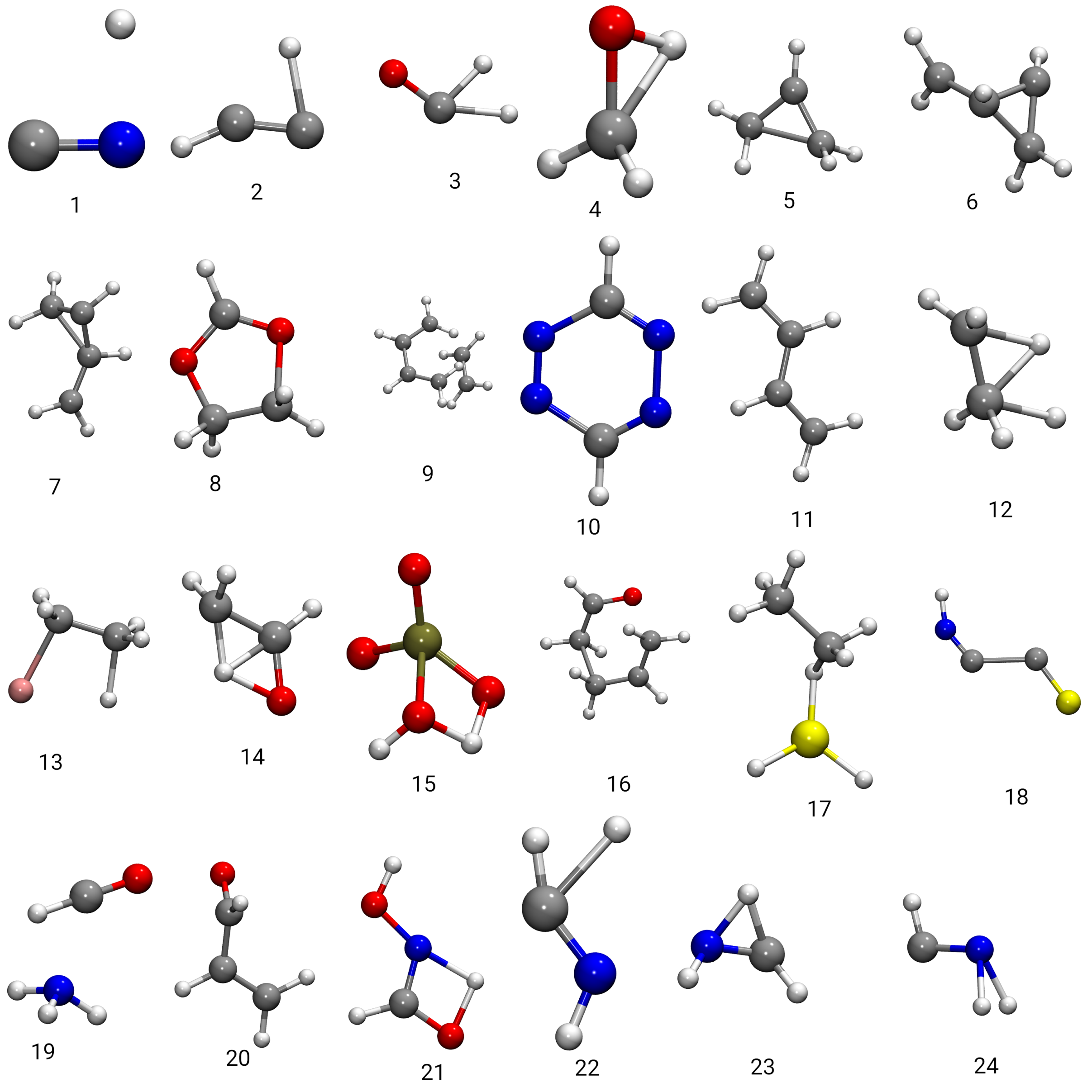


**Figure 7.4:** The 24 molecules used as the test set. Oxygen is red, nitrogen blue, hydrogen white, sulfur yellow, fluorine light red and carbon on grey.

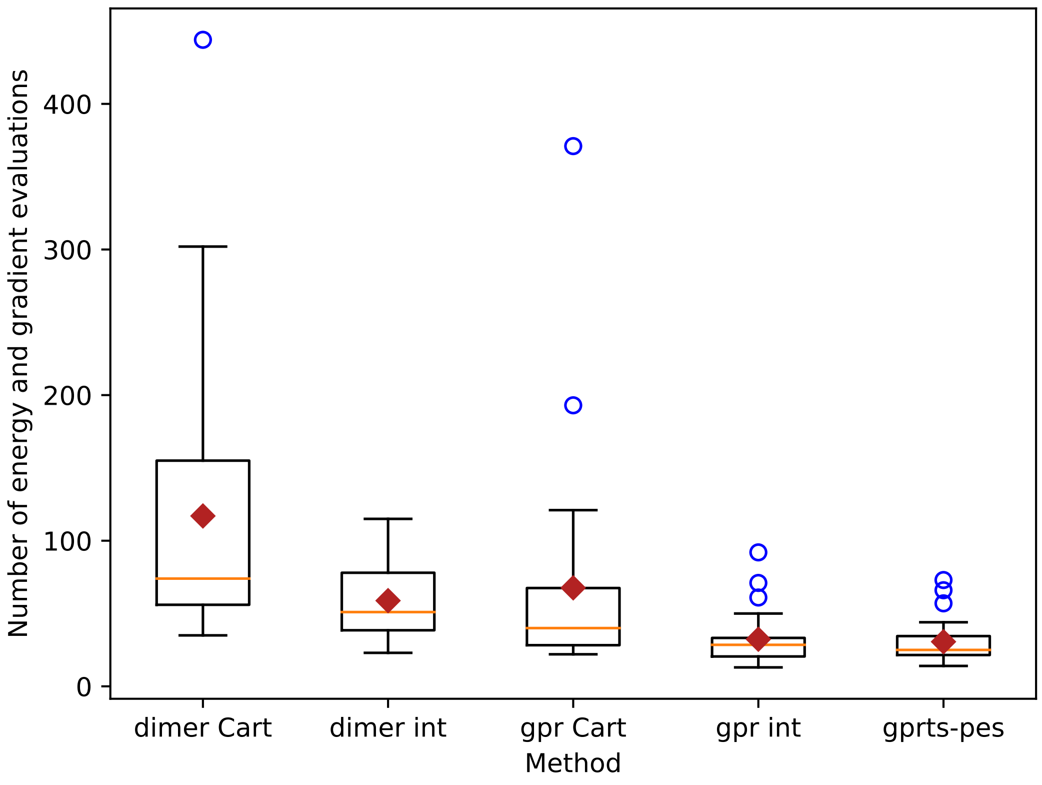


**Figure 7.5:** Performance of the dimer method and the GPR for transition state search in Cartesian (Cart) and internal coordinates.

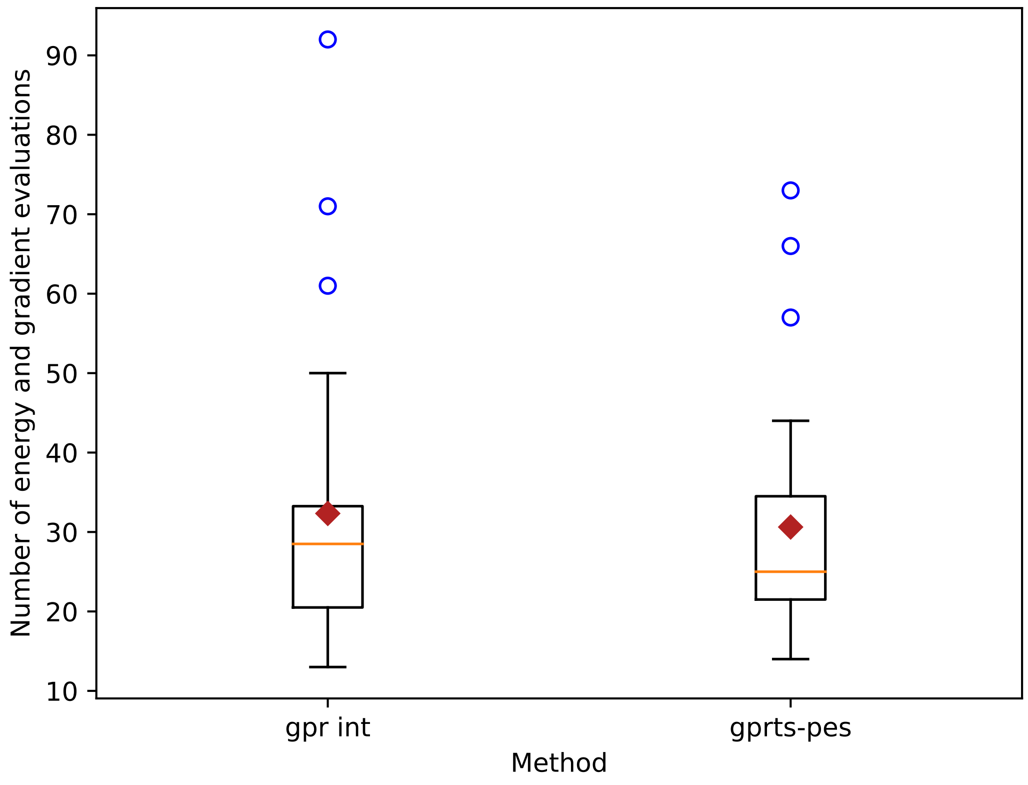


**Figure 7.6:** Performance of GPRTS internal and GPRTS-PES, zoomed in from fig. 7.5.

this time, in contrast to geometry optimization, the internal coordinates introduce problems. Especially the back transformation is more prone to errors. Therefore, an asterisk in table 7.2 indicates the alternative use of the total connection scheme instead of the connection scheme, which uses the bond distance for creation. The total connection scheme is less prone to errors because they do not have dihedrals, which are the main cause of back-transformation failure. However, planar molecules cannot be used within the total connection framework. Having the total connection scheme as a backup eliminates the problems of the back transformation for GPR internal and GPRTS-PES. However, the dimer method for system 22, and system 23 is still not possible to bypass the back transformation failure. Additionally, the dimer method in internal coordinates does not converge for systems 7, 9, and 11. The GPR transition state search in Cartesian coordinates does not converge for systems 7 and 15. The SCF cycle for system 20 of the dimer method in Cartesian coordinates does not converge. Thus, a comparison of the total amount of energy evaluations, in contrast to geometry optimization is not possible. However, when these systems are taken out, a statistical comparison via box plot is still possible.

The median for the dimer method when using internal coordinates instead of Cartesian coordinates is reduced from 74 to 51, this is by $31.1\,\%$ and the mean is reduced by $49.7\,\%$ from 117 down to 58.8. Again, as for geometry optimization, the median is taken for comparison. The first quartile is reduced from 57 to 38.5, $31.1\,\%$, and the third quartile is reduced by $49.7\,\%$ from 155 to 78. For the GPR transition state algorithms, GPR internal and GPRTS-PES, the median is reduced to 28.5 respectively 25, or by $61.5\,\%$ respectively by $66.2\,\%$. The first quartile for GPR internal is reduced by $63.4\,\%$ from 56 to 20.5 and for GPRTS-PES to 21.5 by $61.5\,\%$. For GPR internal the third quartile is reduced from 155 to 33.3 by $78.5\,\%$. GPRTS-PES reduces the third quartile to 34.5 by $77.7\,\%$. The mean is reduced from 117 to 32.3 for GPR internal and to 34.5 for GPRTS-PES, which is by $72.4\,\%$ respectively by $73.8\,\%$.

Comparing GPR internal and GPRTS-PES, the first quartile is $4.6\,\%$ lower for GPR internal, and the third quartile is $3.5\,\%$ lower. However, the median and the mean are $12.3\,\%$ respectively $5.3\,\%$ lower for GPRTS-PES than GPR internal. Since all systems for these two methods converged, a comparison of the total amount of energy and gradient evaluations is possible. GPR internal requires in total 776 energy and gradient evaluations, while GPRT-PES requires 735 energy and gradient calculations, which is a difference of $5.3\%$.

While the first and third quartile is lower for GPR internal, the mean and median are better for GPRTS-PES. This indicates GPRTS-PES has smaller outliers compared to GRP internal. The number of energy evaluations is close together for GPRTS-PES, while GPR internal has

**Table 7.1:** Statistical data for the box plot in fig. 7.5 with different optimization coordinate combinations.

| Optimizer | dimer | | GPR | | |
|---|---|---|---|---|---|
| Coordinate System | Cartesian | internal | Cartesian | internal | GPRTS-PES |
| first quartile ($Q_1$) | 56 | 38.5 | 28.3 | 20.5 | 21.5 |
| median | 74 | 51 | 40 | 28.5 | 25 |
| third quartile ($Q_3$) | 155 | 78 | 67.5 | 33.3 | 34.5 |
| mean | 117 | 58.8 | 67.6 | 32.3 | 30.6 |

a more spread distribution. In contrast to geometry optimization, where a lower dimension needs usually fewer steps to converge, transition state search is more complicated. Therefore, low dimensional systems like system 23, which has $d = 11$, require 73 energy calculations for GPRTS-PES. In contrast system 9 with $d = 42$ requires 57 energy evaluations. In addition, the systems with symmetries, like the planar system 10, lead to difficulties for both GPR algorithms in internal coordinates. The less symmetry a molecule has, the better the internal coordinates work. Especially the total connection scheme can be used in these cases. The total connection scheme is more stable towards the back transformation and thus preferable over the normal dlc code and is also more useful when a bond is destroyed or formed, which is likely in transition state search.

If the median is compared between all methods, GPRTS-PES performs best. However, the computational cost is increased, due to the additional covariance matrix calculations. Since the Cholesky decomposition scales with $\mathcal{O}(d^3)$ and the covariance matrix $\mathbf{K}_{\text{PES}}$ scales with dimension $d^2$ the overall additional cost is in the order of $\mathcal{O}(d^6)$. However, only in the beginning points are sampled. Thus, the number of training points can be neglected, when the dimension is high compared to the number of training points. It is feasible to sample the points, because only a few points are required, even for higher dimensions. This leads to a good Hessian, which describes at least the first eigenvalue good enough.

Additionally, the energy difference compared to the dimer method in Cartesian coordinates is calculated, to indicate whether the same transition state has been found. However, for system 20, where the scf cycle for the dimer method did not converge, the energy is compared to the GPR in Cartesian coordinates. These differences are listed in table 7.3 together with the lowest eigenvalue of the Hessian at the found transition state. Most systems have the same lowest eigenvalue, up to a difference of $10^{-4}\ E_\text{h}/a_0^2$. In this case the system has also the energy difference of $10^{-6}\ E_\text{h}$ or lower. However,some systems have different lowest eigenvalues and higher energy differences. They are listed in the following

**Table 7.2:** Number of energy evaluations for each system shown in fig. 7.4. Additionally, the total amount of energy evaluations for GPR internal and GPRTS-PES was calculated. The asterisk $^\star$ indicates the use of the total connection scheme. If the system did not converge after 500 energy evaluations, this is indicated by nc. scf means, the underlying ab intio program did not converge. dlc refers to a failure of the internal coordinates code.

| Optimizer | Dimer | | GPR | | |
|---|---|---|---|---|---|
| Coordinate System | Cartesian | internal | Cartesian | internal | GPRTS-PES |
| 1 | 53 | 34 | 26 | 13$^\star$ | 14$^\star$ |
| 2 | 74 | 35 | 23 | 21 | 16 |
| 3 | 35 | 38 | 42 | 16 | 20 |
| 4 | 199 | 115$^\star$ | 26 | 14 | 17 |
| 5 | 69 | 69$^\star$ | 42 | 28 | 22$^\star$ |
| 6 | 96 | 79 | 77 | 32 | 26$^\star$ |
| 7 | 221 | nc | nc | 50 | 36 |
| 8 | 79 | 48$^\star$ | 41 | 29$^\star$ | 24$^\star$ |
| 9 | 187 | nc | 94 | 71$^\star$ | 57$^\star$ |
| 10 | 302 | 51 | 193 | 92 | 66 |
| 11 | 444 | nc | 94 | 35 | 32 |
| 12 | 50 | 40$^\star$ | 29 | 24$^\star$ | 20$^\star$ |
| 13 | 63 | 39 | 54 | 22 | 23$^\star$ |
| 14 | 79 | 82 | 39 | 34$^\star$ | 25$^\star$ |
| 15 | 71 | 77 | nc | 33 | 25$^\star$ |
| 16 | 111 | 76 | 104 | 33 | 34 |
| 17 | 123 | 87 | 71 | 33 | 29 |
| 18 | 190 | 47 | 27 | 19 | 22 |
| 19 | 65 | 67 | 29 | 61 | 40$^\star$ |
| 20 | scf | 79 | 121 | 25 | 22 |
| 21 | 45 | 23 | 22 | 25 | 28 |
| 22 | 41 | dlc | 28 | 16 | 44 |
| 23 | 59 | dlc | 47 | 31 | 73 |
| 24 | 35 | 32 | 37 | 19 | 20 |
| total number of steps | – | – | – | 776 | 735 |

**Table 7.3:** The lowest eigenvalue of the Hessian in $E_\mathrm{h}/a_0^2$ at the found transition state and energy difference in Hartree compared to dimer in Cartesian coordinates.

| | Dimer | | GPR | | |
|---|---|---|---|---|---|
| | Cartesian | internal | Cartesian | internal | GPRTS-PES |
| 1 | -0.04393 | -0.04390 $9.2 \cdot 10^{-8}$ | -0.04394 $4.5 \cdot 10^{-8}$ | -0.04393 $2.5 \cdot 10^{-7}$ | -0.04393 $4.5 \cdot 10^{-7}$ |
| 2 | -0.01811 | -0.01806 $1.3 \cdot 10^{-7}$ | -0.01814 $3.0 \cdot 10^{-7}$ | -0.01791 $7.0 \cdot 10^{-6}$ | -0.01813 $1.4 \cdot 10^{-7}$ |
| 3 | -0.12069 | -0.12064 $2.0 \cdot 10^{-7}$ | -0.12066 $7.8 \cdot 10^{-7}$ | -0.12066 $6.1 \cdot 10^{-7}$ | -0.12065 $3.8 \cdot 10^{-7}$ |
| 4 | -0.01321 | -0.01320 $2.8 \cdot 10^{-7}$ | -0.13344 $5.1 \cdot 10^{-2}$ | -0.13342 $5.1 \cdot 10^{-2}$ | -0.13341 $5.1 \cdot 10^{-2}$ |
| 5 | -0.3114 | -0.03123 $9.1 \cdot 10^{-7}$ | -0.3123 $1.2 \cdot 10^{-6}$ | -0.3121 $1.2 \cdot 10^{-6}$ | -0.3121 $1.0 \cdot 10^{-6}$ |
| 6 | -0.00322 | -0.00325 $8.8 \cdot 10^{-7}$ | -0.00324 $1.0 \cdot 10^{-6}$ | -0.00326 $5.4 \cdot 10^{-7}$ | -0.00325 $2.6 \cdot 10^{-7}$ |
| 7 | -0.01309 | – | – | -0.00957 $9.5 \cdot 10^{-3}$ | -0.01301 $1.0 \cdot 10^{-6}$ |
| 8 | -0.02066 | -0.02061 $1.6 \cdot 10^{-6}$ | -0.02067 $1.0 \cdot 10^{-7}$ | -0.02067 $1.2 \cdot 10^{-8}$ | -0.02067 $4.4 \cdot 10^{-8}$ |
| 9 | -0.00623 | – | -0.00624 $3.8 \cdot 10^{-6}$ | -0.00623 $3.0 \cdot 10^{-6}$ | -0.00623 $4.0 \cdot 10^{-6}$ |
| 10 | -0.01064 | -0.01064 $8.2 \cdot 10^{-7}$ | -0.01064 $1.0 \cdot 10^{-6}$ | -0.01077 $1.0 \cdot 10^{-4}$ | -0.00257 $5.6 \cdot 10^{-3}$ |
| 11 | -0.04196 | – | 0.00000 $1.2 \cdot 10^{-1}$ | -0.00145 $1.1 \cdot 10^{-1}$ | -0.00145 $1.0 \cdot 10^{-1}$ |
| 12 | -0.13308 | -0.13244 $2.6 \cdot 10^{-8}$ | -0.13297 $9.1 \cdot 10^{-7}$ | -0.13298 $8.2 \cdot 10^{-7}$ | -0.13303 $7.0 \cdot 10^{-7}$ |
| 13 | -0.10651 | -0.10652 $7.6 \cdot 10^{-7}$ | -0.10647 $5.8 \cdot 10^{-8}$ | -0.10647 $1.3 \cdot 10^{-7}$ | -0.10650 $2.3 \cdot 10^{-7}$ |
| 14 | -0.14452 | -0.14466 $8.2 \cdot 10^{-7}$ | -0.14433 $3.3 \cdot 10^{-6}$ | -0.14435 $1.3 \cdot 10^{-6}$ | -0.14436 $1.2 \cdot 10^{-6}$ |
| 15 | -0.03975 | -0.03972 $3.3 \cdot 10^{-7}$ | – | -0.03974 $7.0 \cdot 10^{-7}$ | -0.03979 $2.8 \cdot 10^{-7}$ |
| 16 | -0.00633 | -0.00642 $1.2 \cdot 10^{-6}$ | -0.00636 $2.0 \cdot 10^{-6}$ | -0.00633 $7.5 \cdot 10^{-7}$ | -0.00636 $1.3 \cdot 10^{-6}$ |
| 17 | -0.04024 | -0.04025 $2.5 \cdot 10^{-7}$ | -0.04025 $4.4 \cdot 10^{-6}$ | -0.04020 $1.4 \cdot 10^{-6}$ | -0.04026 $2.0 \cdot 10^{-6}$ |
| 18 | -0.00199 | -0.00199 $1.3 \cdot 10^{-7}$ | -0.00199 $1.6 \cdot 10^{-8}$ | -0.00199 $1.2 \cdot 10^{-7}$ | -0.00199 $2.2 \cdot 10^{-8}$ |
| 19 | $\star\star$ | -0.03411 $2.6 \cdot 10^{-6}$ | -0.03411 $2.7 \cdot 10^{-6}$ | -0.03411 $7.1 \cdot 10^{-5}$ | -0.03411 $2.9 \cdot 10^{-6}$ |
| 20 | – | -0.00213 $8.0 \cdot 10^{-7}$ | -0.00212 | -0.00213 $7.9 \cdot 10^{-7}$ | -0.00213 $8.1 \cdot 10^{-7}$ |
| 21 | $\star\star$ | $\star\star$ $5.4 \cdot 10^{-8}$ | $\star\star$ $5.2 \cdot 10^{-7}$ | $1.2 \cdot 10^{-3}$ | $1.2 \cdot 10^{-3}$ |
| 22 | -0.08485 | – | -0.08484 $2.9 \cdot 10^{-7}$ | -0.08485 $1.5 \cdot 10^{-8}$ | -0.08485 $6.2 \cdot 10^{-8}$ |
| 23 | -0.13585 | – | -0.13586 $1.7 \cdot 10^{-7}$ | -0.13585 $3.5 \cdot 10^{-9}$ | -0.13585 $1.3 \cdot 10^{-7}$ |
| 24 | -0.12665 | -0.12666 $6.9 \cdot 10^{-7}$ | -0.08361 $1.4 \cdot 10^{-2}$ | -0.12663 $8.9 \cdot 10^{-7}$ | -0.12663 $8.6 \cdot 10^{-7}$ |

- System 4: The dimer method in Cartesian coordinates found a different transition state compared to all other methods. This explains the differences in the energy of about $10^{-2}$ $E_\mathrm{h}$ and the different lowest eigenvalue.
- System 7: GPR internal has a different transition state compared to the Cartesian dimer and GPRTS-PES.
- System 10: GPR internal and GPRTS-PES found a different transition state than the other methods.
- System 11: GPR internal and GPRTS-PES found a different transition state. GPR in Cartesian coordinates converged to a minimum instead of a transition state.
- System 21: Only GPR internal and GPRTS-PES found a saddle of first order, while the dimer method in both coordinate systems and GPR in Cartesian converged to a saddle point of second order. They have two negative eigenvalues of the Hessian.
- System 24: GPR in Cartesian converged to a different transition state.

### 7.2.3 Hyperparameters

As for the GPR for geometry optimization, two hyperparameters can influence on the performance, namely the length scale $\ell$ and the gradient noise parameter $\sigma_g$. In fig. 7.7 the dependence of $\log\, p$ on the length scale $\ell$ and the gradient noise $\sigma_g$ is shown for system 6 for GPR internal. On the top, it is shown for the total connection scheme, and on the bottom for the normal dlc connection scheme. In fig. 7.8 the same properties and the same system are plotted for GPRTS-PES. The number of training points included for these plots is six. As can be seen on these plots, the length scale and the gradient noise are somewhat decoupled and can thus be treated separately.

Comparing the logarithm of the marginal likelihood for the total connection scheme and the normal dlc connectivity for GPR internal in fig. 7.7, the difference between them is a slight shift of the maximum to a higher length scale for dlc, roughly $2\,a_0$. The qualitative likelihood remains similar. The most significant change is the negative $\log\, p$ values for length scales above $17.5\,a_0$.

Qualitatively the same result is achieved for $\log\, p$ for GPRTS-PES, where again a slight shift to higher length scales is preferred for the length scale. In the following first the length scale dependence of the log marginal likelihood will be discussed, followed by the discussion of the gradient noise $\sigma_g$.

**Length Scale**

The length scale dependence $\log\, p(\ell)$ is shown in fig. 7.9 for GPR internal, again for tc on the top and dlc on the bottom. System 6 was used again. The maximum for dlc is more peaked and lies at $\ell \approx 11\,a_0$, while the maximum for tc is at $\ell \approx 12.5\,a_0$. Increasing the number of training points, in this case from 6 to 8 to 10, does not significantly change the maximum position, only the $\log\, p$ value increases for both connection schemes. Thus, a constant length scale for the GPR internal is justified. Since the other systems showed a similar result a constant $\ell = 13\,a_0$ is chosen, and, in contrast to GPR geometry optimization, is not changed throughout the transition state search. The length scale for the total connection scheme is usually larger, because this scheme uses all atoms to connect, resulting in larger coordinates that are recognized by the marginal likelihood.

Looking at the length scale dependence for GPRTS-PES a similar result is found and is shown in fig. 7.10. For the tc case, qualitatively the same result is achieved as for GPR internal. The maximum is more shallow compared to dlc and is at $\ell \approx 12\,a_0$. Again, the position of the maximum does not change significantly with the number of training points, just like before. However, for GPRTS-PES with the dlc scheme, there are some differences. First, the maximum is at a slightly lower position, namely at $\ell \approx 10\,a_0$, and remains constant for a different number of training points. In contrast to GPR internal, there is a steep decrease in the

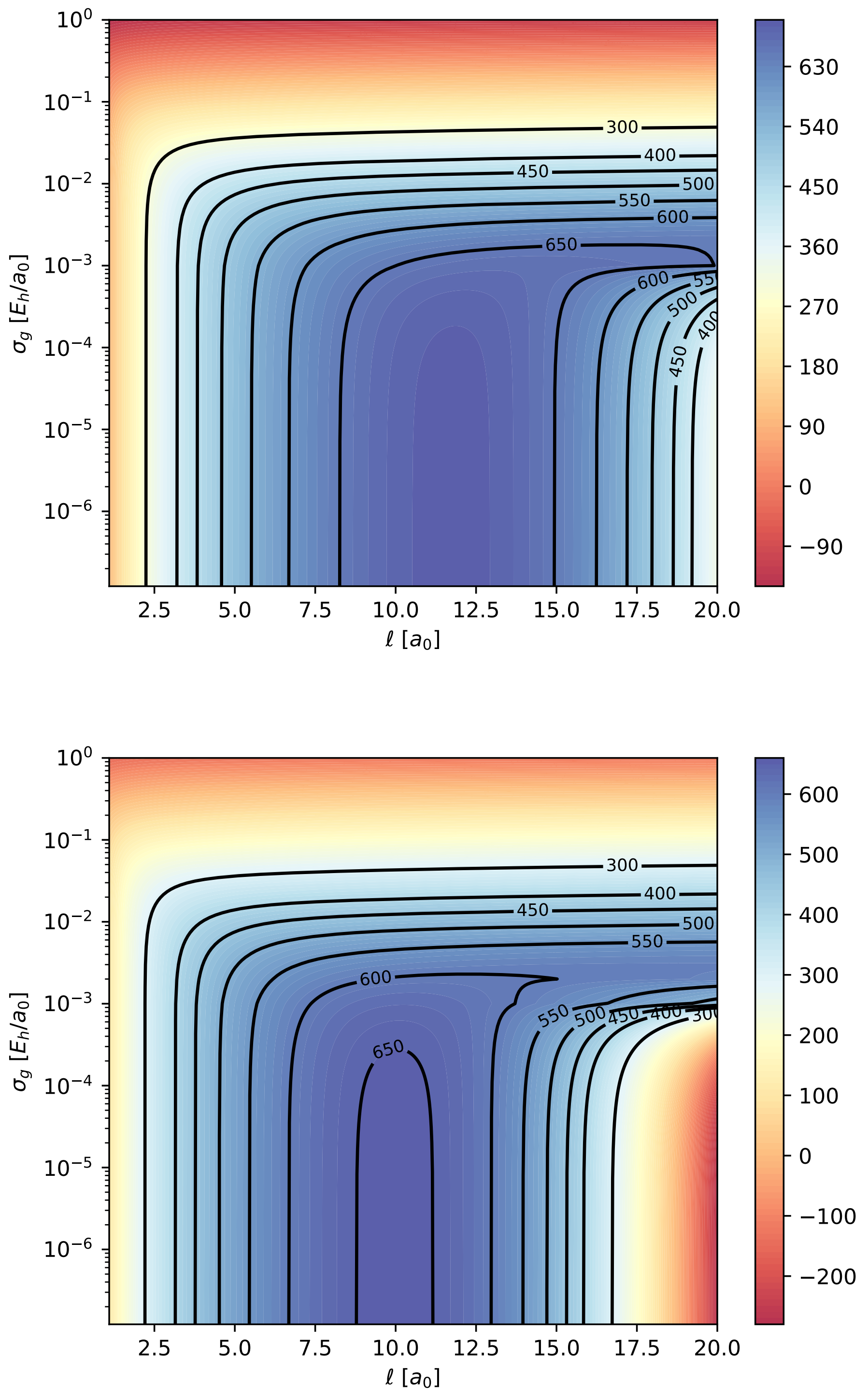
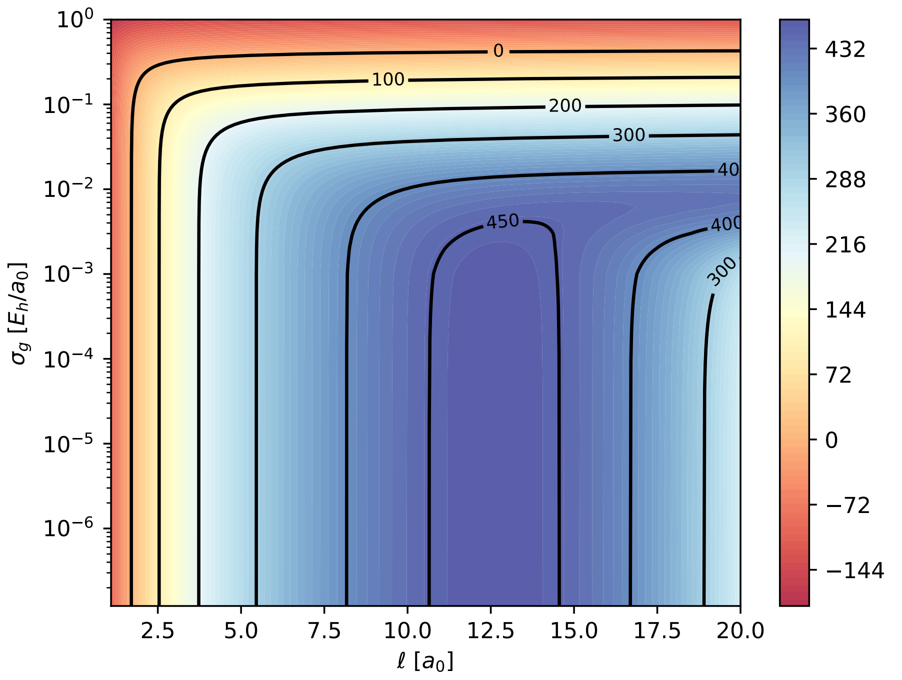


**Figure 7.7:** Heat map of the logarithm of the marginal likelihood $\log p$ depending on the length scale $\ell$ and the gradient noise $\sigma_g$ for GPR internal for total connection scheme (tc) on the top and dlc on the bottom.



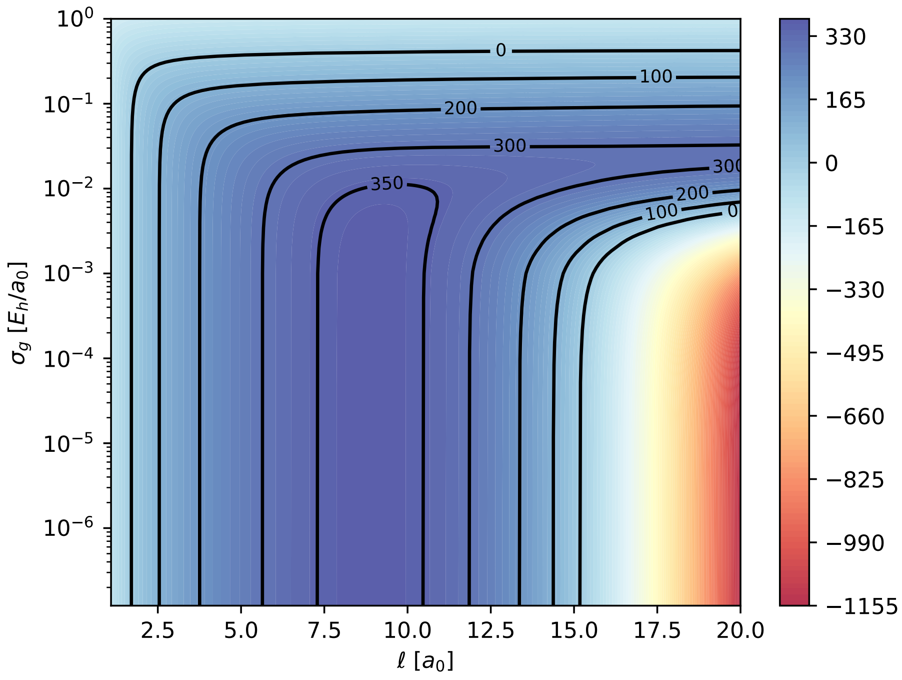
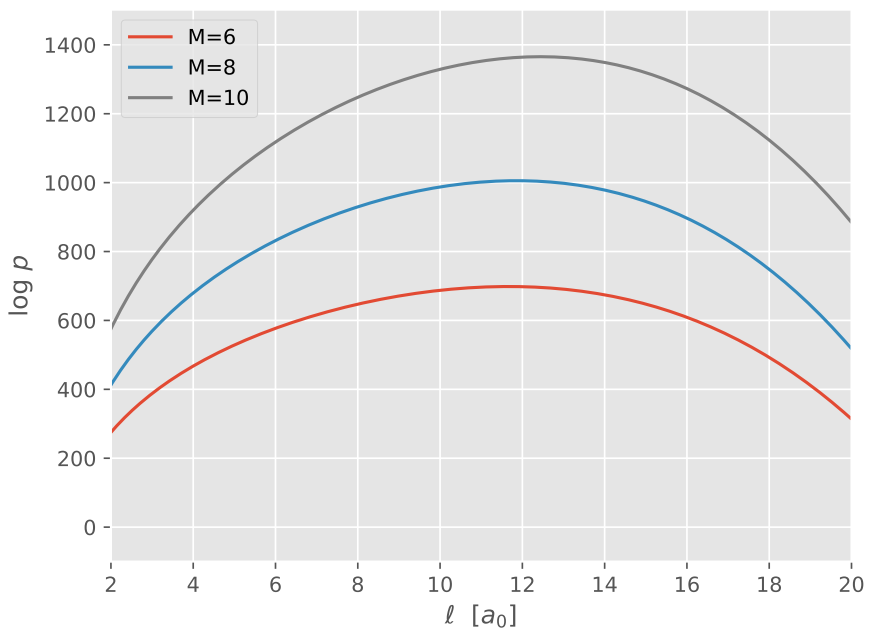


**Figure 7.8:** Heat map of the logarithm of the marginal likelihood $\log p$ depending on the length scale $\ell$ and the gradient noise $\sigma_g$ for GPRTS-PES, for tc on the top and dlc on the bottom.



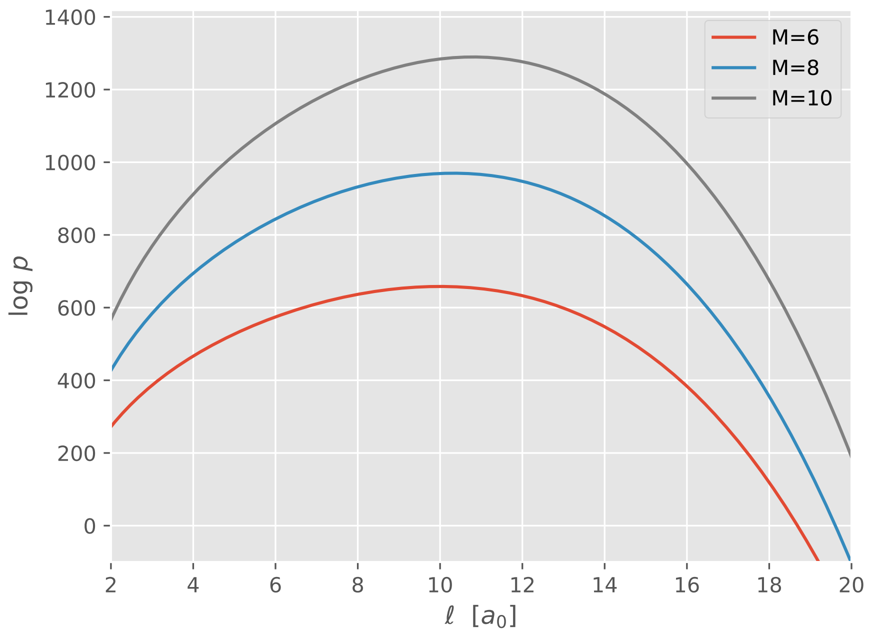


**Figure 7.9:** The logarithm of marginal likelihood $\log p$ depending on the length scale $\ell$ for GPRTS internal, on the top tc and on the bottom dlc. The number of training points is indicated with $M$.

likelihood for larger length scales, indicating that smaller length scales are preferred. Thus, the length scale is fixed at $\ell = 13\,a_0$ for GPRTS-PES (both tc and dlc) too.

An alternative way to select the length scale for the dlc case for GPR internal and GPRTS-PES is to calculate the distances between all dlc coordinates and use the maximum distance between these coordinates as the length scale. When using this approach the results do not defer significantly from the constant length scale $\ell = 13\,a_0$. However, this approach is not feasible for tc, because the distances between the internal coordinates are much larger due to every atom being connected to every atom and thus leads to too large length scales, which can worsen the result by increasing the number of energy evaluations.

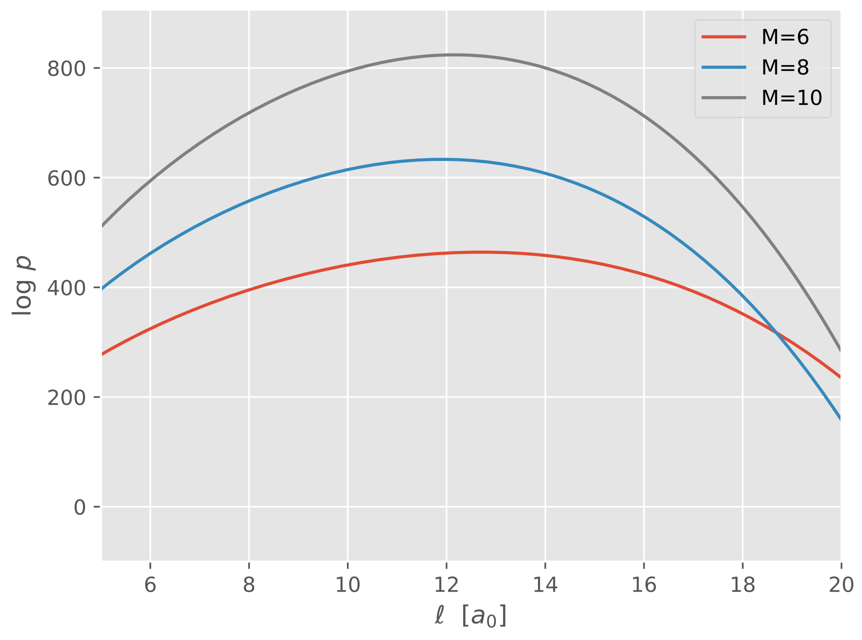
**Gradient Noise Parameter**

Just like for geometry optimization, the noise parameter $\sigma$ can be split into one for the energy $\sigma_e$ and one for the gradient $\sigma_g$. The noise parameter for the energy is set to $\sigma_e = 10^{-7}\,E_\mathrm{h}$, like for geometry optimization. Therefore, only the gradient noise is investigated. For GPR internal $\log p(\sigma_g)$ is plotted in fig. 7.11 for tc on top and dlc on the bottom. In fig. 7.12 $\log p(\sigma_g)$ is shown for GPRTS-PES for tc on the top and dlc on the bottom.

For GPR internal, both tc and dlc, the marginal likelihood is independent of the gradient noise for $\sigma_g < 10^{-4}\ E_\mathrm{h}/a_0$, which is the same result as for geometry optimization. One difference is, that for a smaller number of training points, i.e. $M = 6$ the marginal likelihood is already independent of the gradient noise, if $\sigma_g < 10^{-3}\ E_\mathrm{h}/a_0$.

When looking at the gradient noise for GPRTS-PES in fig. 7.12, there are some small differences between tc and dlc. While the marginal likelihood for tc is already independent of the gradient noise, for $\sigma_g < 5 \cdot 10^{-2}\ E_\mathrm{h}/a_0$, for dlc the independence is if $\sigma_g < 5 \cdot 10^{-3}\ E_\mathrm{h}/a_0$. Therefore, the marginal likelihood is independent over a larger area of the gradient noise.

Qualitatively for small noise parameters, the marginal likelihood for both methods does not depend on the gradient noise. Thus, the gradient noise for both methods was set to $\sigma_g = 10^{-7}$ $E_\mathrm{h}/a_0$, just like for GPR internal 1 for geometry optimization.



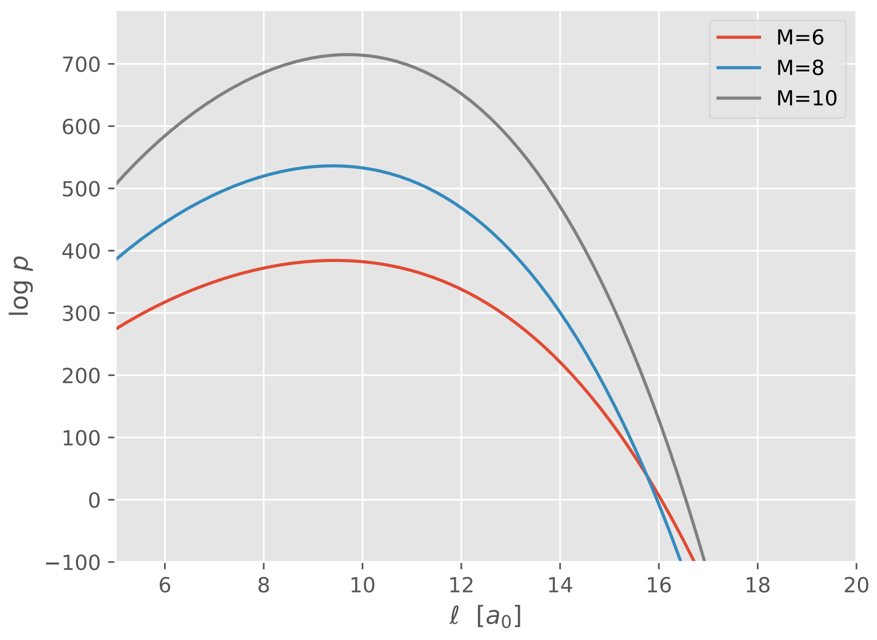
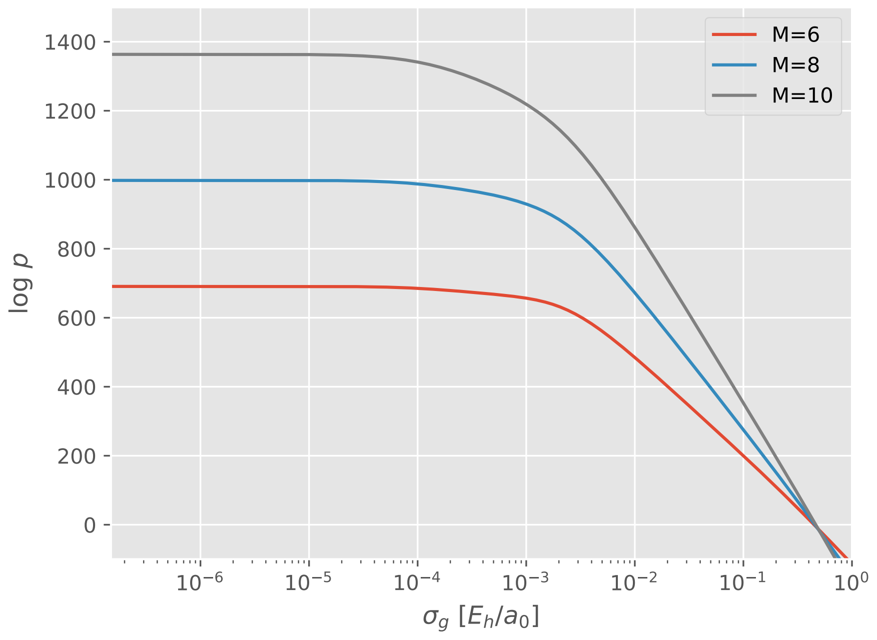


**Figure 7.10:** The logarithm of the marginal likelihood $\log p$ depending on the length scale $\ell$ for GPRTS-PES. tc is shown on the top, dlc on the bottom. The number of training points is indicated with $M$.



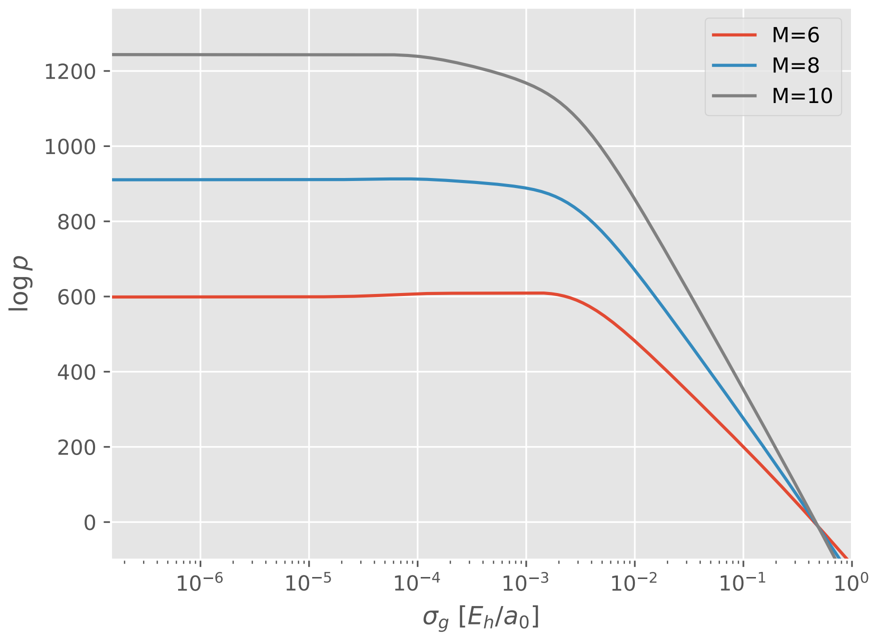
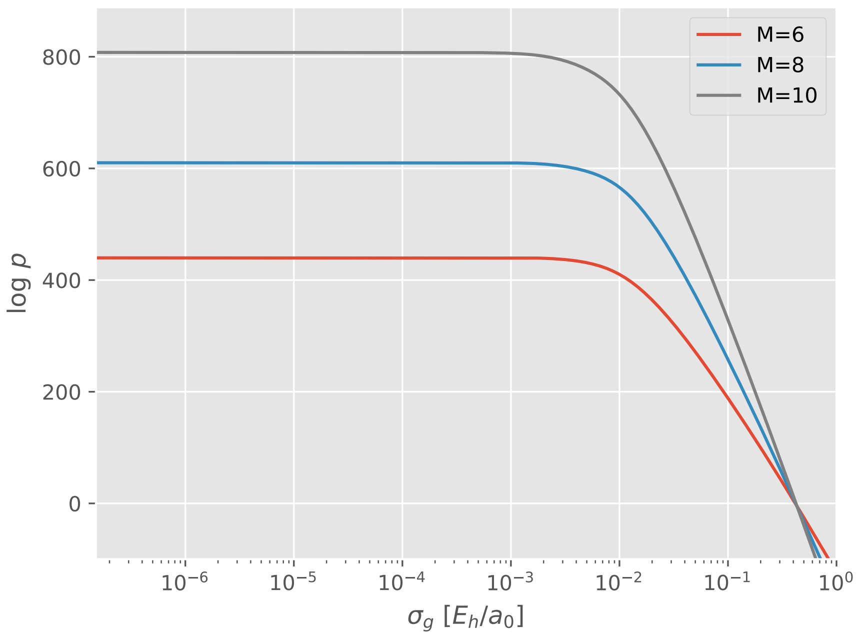


**Figure 7.11:** The logarithm of the marginal likelihood $\log\, p$ depending on the gradient noise $\sigma_g$ for GPRTS internal. The total connection scheme is on the top, dlc on the bottom.



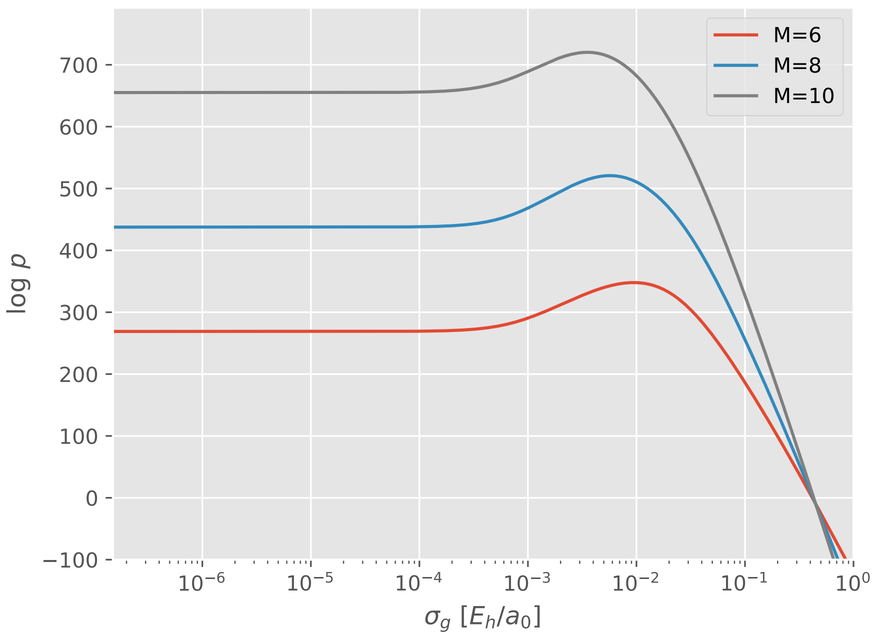


**Figure 7.12:** The marginal likelihood $\log p$ depending on the gradient noise $\sigma_g$ for GPRTS PES. On the top, the total connection scheme is shown, on the bottom dlc.

## 7.3 Conclusion

The transition state search in internal coordinates based on Gaussian process regression reduces the total amount of energy and gradient evaluations, for both methods GPRTS internal and GPRTS-PES, compared to Cartesian coordinates. GPRTS-PES performs slightly better than GPRTS internal. In contrast to internal 2 for geometry optimization, this approach to avoid the iterative back transformation is not feasible, because the additional matrix-matrix multiplications for the Hessian evaluation on the GPR surrogate surface introduce too much overhead. Therefore both methods introduced here require a back transformation of the prediction in internal coordinates to Cartesian coordinates. However, the back transformation is prone to failure especially for transition state search, in contrast to geometry optimization.

Most problems arise for systems, where bond breaking and formation play a role, which is more likely for transition state search than geometry optimization. Sometimes more than one molecule is involved in a transition state search, which also introduces problems to the dlc code. In these cases the total connection scheme is more stable, because there are no dihedrals included, which is usually the reason for a back transformation failure. However, just using the total connection scheme does not solve this problem entirely. First, the dimer method, for example, in internal coordinates has still problems with the back transformation for two systems. Second, the total connection scheme cannot be applied to planar and linear molecules. Both GPRTS methods seem to solve this problem when using the total connection scheme as an alternative to dlc.

GPRTS internal is computationally less expensive compared to GPRTS-PES because the sampling with the acquisition function requires more computational cost than the dimer rotation on the GPR surrogate surface for GPRTS internal. Comparing the total computational time of all GPR calculations, GPRTS internal requires 34.9 s on an Intel Xeon Silver 4214R CPU, while GPRTS-PES requires 29.2 s. However, GPRTS-PES uses $22.5\,\%$ of CPU time in total, and GPRTS internal only $9\,\%$. The advantage of GPRTS-PES is, that it reduces the total amount of energy evaluation, for this system, to a lower number compared to GPRTS internal. GPRTS-PES requires in general fewer samples to correctly describe the Hessian for a translation step than GPRTS internal. Thus, it is justified to use the increased computational cost introduced by maximizing the acquisition function. Since the implementation uses no parallelization of the code, a certain speed-up can be expected for the maximization of the acquisition function for GPRTS-PES.

There are possible improvements to overall improve the performance of GPRTS-PES. One option would be to draw samples from the distribution $p(\mathbf{x}_\star|\mathcal{D})$ to cover a larger variety of $\mathbf{x}_\star$ as transition states to sample, like in [81]. This would allow not only to detect the correct direction of the transition state on the real potential energy surface but rather have a sufficient length of the next point to improve the overall behavior of the algorithm. However, this would introduce additional computational cost. Another possibility would be, to use the acquisition function after each translation step to select a possible better guess than the P-RFO on the GPR surrogate surface. This again would introduce additional computational cost.

Additionally, there is room for improvement for the internal coordinates. The total connection scheme could be expanded by not only calculating all distances but also calculating all angles between three atoms. This would allow more flexibility to the internal coordinates and not introduce the susceptibility to errors caused by the dihedral angles. It would not require recalculating the primitive internal coordinates, after bond breaking and/or bond formation.

The length scale $\ell$ and the noise parameter for the gradient $\sigma_g$ are the relevant hyperparameters for a transition state search. As could be shown, they can be treated separately, because they are independent of each other. The length scale is fixed to $\ell = 13\,a_0$ and is kept constant, in contrast to geometry optimization based on GPR. All noise parameters with $\sigma_g < 10^{-4}$ $E_\mathrm{h}/a_0$ are well suited for GPRTS internal and GPRTS-PES. Thus $\sigma_g = 10^{-7}$ $E_\mathrm{h}/a_0$ is chosen as the noise parameter, which also acts as regularization for the Cholesky decomposition for numerical stability.

In summary, there are two ways presented, how to select training points to describe the Hessian as accurately as possible for the transition state search on the GPR surface in internal coordinates. The first way, GPR internal, selects the points by simply aligning a dimer to the eigenvector corresponding to the lowest eigenvalue. The other way, GPRTS-PES, uses an acquisition function, which is maximized to sample the next points. Introducing also the total connection scheme, both ways seem to handle the susceptibility of errors of the back transformation well. GPRTS-PES seems to perform a bit better than GPRTS internal but requires additional CPU time. All in all, GPR with the use of the total connection scheme is advantageous for a transition state search, compared to the standard dimer method. However, if the system is planar or linear, the dlc can still be used, because in those cases no total connection scheme can be calculated.

# 8 Final Discussion and Outlook

In this thesis, two algorithms were presented to combine GPR and internal coordinates, one for geometry optimization, and the other for TS search. For geometry optimization, two approaches were presented. Both of them evaluate the covariance function in internal coordinates. However, GPR internal 1 builds the GPR surface in internal coordinates, while GPR internal 2 uses the chain rule to build the GPR surface in Cartesian coordinates. GPR internal 2 is not extended to TS search due to the high computational overhead, instead, another approach of using an acquisition function was presented, see section 7.1.2. In both algorithms, the combination of GPR and internal coordinates performs better than the standard algorithms. They significantly reduce the total amount of energy evaluations.

The main drawback in internal coordinates is the iterative back-transformation. While it did not show any problems for geometry optimization, it is prone to errors for TS search. In addition, for geometry optimization, an alternative was presented, namely GPR internal 2. To a certain degree, this problem can be solved by using the total connection scheme to construct the primitive internal coordinates, see section 4.1.

The hyperparameters were obtained heuristically. The marginal likelihood was calculated for a test system and a nearby maximum of for this test system was chosen. Since the underlying ab initio program causes only little noise, the lowest possible value for the gradient noise was chosen, that still allows a stable Cholesky decomposition. The length scale for the geometry optimization is chosen fixed at the beginning and dynamically reduces towards convergence. However, in the TS search, the length scale over the whole optimization procedure is kept constant. These ways to choose the length scale performed best. In addition, there is no expensive maximization of the marginal likelihood required.

GPR internal 2 can be compared to Meyer et al. [74], at least those molecules with the same starting geometry. The implementation here seems to perform better compared to their work. The other approach, GPR internal 1 can be compared to Raggi et al. [18]. Their implementation of Kriging, which is essentially GPR, seems to perform better than this work. The main reason might be, that they use $d$ length scale parameters, where $d$ is the dimension of the system. Their length scales are determined by a surrogate model Hessian, where they use the

$i$-th eigenvalue of that Hessian to obtain the $i$-th length scale parameter. Systems 8, 23, 24, and 26 were excluded, they need 218 steps in total, while this work required 314 steps in total. Using correctly adapted length scales for each dimension, seem to perform better compared to a single length scale parameter, especially for larger molecules.

A comparison of GPRTS internal and GPRTS-PES can be done with the work of Galván et al. [25]. They use again a surrogate model Hessian to obtain length scale parameters for each dimension. They outperform this work. Using different length scales for each dimension seems to work significantly better compared to one length scale parameter. The reason might be, that for TS search in two different subspaces, an optimization is performed, i.e a maximization in one dimension and a minimization in all other dimensions. Thus implementing the Matérn covariance function for all dimensions could decrease further the number of energy evaluations.

The recent usage of automatic differentiation in ML [90] can be used to accelerate the calculation of the derivatives of the GPR and thus reduce the computational time of a single GPR step. This is especially useful for GPR internal 2, here a lot of derivatives have to be calculated due to the chain rule. Applying automatic differentiation would also allow internal 2 to be useful for TS search since the computational overhead would become negligible. It would also speed up the optimization of the acquisition function for GPRTS-PES because the number of expensive derivatives would be replaced by automatic derivatives.

Overall the GPR optimizers in internal coordinates seem to perform better than any other optimizer/coordinate system combination. Therefore, the GPR optimizers in internal coordinates can be used to achieve faster search for stationary points and save time for other calculations. In addition, the GPR seems to perform more stable in internal coordinates, than other optimizers.

# Bibliography

[1] D. Born and J. Kästner: *Geometry Optimization in Internal Coordinates Based on Gaussian Process Regression: Comparison of Two Approaches*. Journal of Chemical Theory and Computation **17** (9), 5955–5967 (2021)

[2] E. Schrödinger: *An Undulatory Theory of the Mechanics of Atoms and Molecules*. Physical Review **28** (6), 1049–1070 (1926)

[3] O. Burrau: *Berechnung des Energiewertes des Wasserstoffmolekel-Ions ($H_2^+$) im Normalzustand*. Naturwissenschaften **15** (1), 16–17 (1927)

[4] M. Born and R. Oppenheimer: *Zur Quantentheorie der Molekeln*. Annalen der Physik **389** (20), 457–484 (1927)

[5] K. J. Laidler and M. C. King: *Development of transition-state theory*. The Journal of Physical Chemistry **87** (15), 2657–2664 (1983)

[6] P. Pulay and G. Fogarasi: *Geometry optimization in redundant internal coordinates*. The Journal of Chemical Physics **96** (4), 28562860 (1992)

[7] C. Peng, P. Y. Ayala, H. B. Schlegel, and M. J. Frisch: *Using redundant internal coordinates to optimize equilibrium geometries and transition states*. Journal of Computational Chemistry **17** (1), 4956 (1996)

[8] J. Baker and F. Chan: *The location of transition states: A comparison of Cartesian, Z-matrix, and natural internal coordinates*. Journal of Computational Chemistry **17** (7), 888–904 (1996)

[9] S. R. Billeter, A. J. Turner, and W. Thiel: *Linear scaling geometry optimisation and transition state search in hybrid delocalised internal coordinates*. Physical Chemistry Chemical Physics **2**, 2177–2186 (2000)

[10] C. E. Rasmussen and C. K. Williams: *Gaussian Processes for Machine Learning*. MIT Press (2006)

[11] J. P. Alborzpour, D. P. Tew, and S. Habershon: *Efficient and accurate evaluation of potential energy matrix elements for quantum dynamics using Gaussian process regression*. The Journal of Chemical Physics **145** (17), 174112 (2016)

[12] A. P. Bartók, R. Kondor, and G. Csányi: *On representing chemical environments*. Physical Review B **87**, 184115 (2013)

[13] M. J. Mills and P. L. Popelier: *Intramolecular polarisable multipolar electrostatics from the machine learning method Kriging*. Computational and Theoretical Chemistry **975** (1), 4251 (2011)

[14] R. Raghunathan and O. A. von Lilienfeld: *Many Molecular Properties from One Kernel in Chemical Space* (2015)

[15] A. Denzel and J. Kästner: *Gaussian process regression for geometry optimization*. The Journal of Chemical Physics **148** (9), 094114 (2018)

[16] G. Schmitz and O. Christiansen: *Gaussian process regression to accelerate geometry optimizations relying on numerical differentiation*. The Journal of Chemical Physics **148** (24), 241704 (2018)

[17] E. Garijo del Río, J. J. Mortensen, and K. W. Jacobsen: *Local Bayesian optimizer for atomic structures*. Physical Review B **100**, 104103 (2019)

[18] G. Raggi, I. F. Galván, C. L. Ritterhoff, M. Vacher, and R. Lindh: *Restricted-Variance Molecular Geometry Optimization Based on Gradient-Enhanced Kriging*. Journal of Chemical Theory and Computation **16** (6), 3989–4001 (2020)

[19] Z. D. Pozun, K. Hansen, D. Sheppard, M. Rupp, K. Müller et al.: *Optimizing transition states via kernel-based machine learning*. The Journal of Chemical Physics **136** (17), 174101 (2012)

[20] A. A. Peterson: *Acceleration of saddle-point searches with machine learning*. The Journal of Chemical Physics **145** (7), 074106 (2016)

[21] O. Koistinen, F. B. Dagbjartsdóttir, V. Ásgeirsson, A. Vehtari, and H. Jónsson: *Nudged elastic band calculations accelerated with Gaussian process regression*. The Journal of Chemical Physics **147** (15), 152720 (2017)

[22] A. Denzel and J. Kästner: *Gaussian Process Regression for Transition State Search*. Journal of Chemical Theory and Computation **14**, 57775786 (2018)

[23] R. Meyer, K. S. Schmuck, and A. W. Hauser: *Machine Learning in Computational Chemistry: An Evaluation of Method Performance for Nudged Elastic Band Calculations*. Journal of Chemical Theory and Computation **15** (11), 65136523 (2019)

[24] O. Koistinen, V. Ásgeirsson, A. Vehtari, and H. Jónsson: *Nudged Elastic Band Calculations Accelerated with Gaussian Process Regression Based on Inverse Interatomic Distances*. Journal of Chemical Theory and Computation **15** (12), 67386751 (2019)

[25] I. F. Galván, G. Raggi, and R. Lindh: *Restricted-Variance Constrained, Reaction Path, and Transition State Molecular Optimizations Using Gradient-Enhanced Kriging*. Journal of Chemical Theory and Computation **17** (1), 571582 (2021)

[26] K. Ahuja, W. H. Green, and Y. Li: *Learning to Optimize Molecular Geometries Using Reinforcement Learning*. Journal of Chemical Theory and Computation **17** (2), 818825 (2021)

[27] F. Jensen: *Introduction to Computational Chemistry*. John Wiley & Sons Ltd (2007)

[28] C. G. Broyden: *The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations*. IMA Journal of Applied Mathematics **6** (1), 76–90 (1970)

[29] R. Fletcher: *A new approach to variable metric algorithms*. The Computer Journal **13** (3), 317–322 (1970)

[30] D. Goldfarb: *A family of variable-metric methods derived by variational means*. Mathematics of Computation **24**, 23–26 (1970)

[31] D. F. Shanno: *Conditioning of quasi-Newton methods for function minimization*. Mathematics of Computation **24**, 647–656 (1970)

[32] D. Liu and J. Nocedal: *On the limited memory BFGS method for large scale optimization*. Mathematical Programming (45), 503528 (1989)

[33] C. Lemaréchal: *Cauchy and the gradient method*. Doc Math Extra **251** (254), 10 (2012)

[34] M. R. Hestenes and E. Stiefel: *Methods of Conjugate Gradients for Solving Linear Systems*. Journal of Research of the National Bureau of Standards **49** (6), 409436 (1952)

[35] R. Fletcher and C. M. Reeves: *Function minimization by conjugate gradients*. The Computer Journal **7** (2), 149–154 (1964)

[36] E. Polak and G. Ribière: *Note sur la convergence de méthodes de directions conjuguées*. Revue Française d'Automatique, Informatique, Recherche Opérationnelle **3** (1), 3543 (1969)

[37] Y. H. Dai and Y. Yuan: *A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property*. SIAM Journal on Optimization **10** (1), 177–182 (1999)

[38] G. Henkelman and H. Jónsson: *A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives*. The Journal of Chemical Physics **111** (15), 7010–7022 (1999)

[39] J. Kästner and P. Sherwood: *Superlinearly converging dimer method for transition state search*. Journal of Chemical Physics **128**, 014106 (2008)

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

[41] C. J. Cerjan and W. H. Miller: *On finding transition states*. The Journal of Chemical Physics **75** (6), 28002806 (1981)

[42] J. Baker: *An algorithm for the location of transition states*. Journal of Computational Chemistry **7** (4), 385–395 (1986)

[43] A. Banerjee, N. Adams, J. Simons, and R. Shepard: *Search for stationary points on surfaces*. The Journal of Physical Chemistry **89** (1), 5257 (1985)

[44] P. Pulay, G. Fogarasi, F. Pang, and J. E. Boggs: *Systematic ab initio gradient calculation of molecular geometries, force constants, and dipole moment derivatives*. Journal of the American Chemical Society **101** (10), 25502560 (1979)

[45] J. Kästner, J. M. Carr, T. W. Keal, W. Thiel, A. Wander et al.: *DL-FIND: An Open-Source Geometry Optimizer for Atomistic Simulations*. J. Phys. Chem. A **113** (43), 11856–11865 (2009)

[46] E. B. Wilson, J. C. Decius, and P. C. Cross: *Molecular Vibrations*. McGraw-Hill, New York (1955)

[47] A. Ben-Israel and T. N. E. Greville: *Generalized Inverses: Theory and Applications*. Springer (2003)

[48] E. H. Moore: *On the reciprocal of the general algebraic matrix*. Bulletin of the American Mathematical Society **9** (26), 385–396 (1920)

[49] R. Penrose: *A generalized inverse for matrices*. Mathematical Proceedings of the Cambridge Philosophical Society **51** (3), 406413 (1955)

[50] J. Baker, A. Kessi, and B. Delley: *The generation and use of delocalized internal coordinates in geometry optimization*. The Journal of Chemical Physics **105** (1), 192212 (1996)

[51] E. Parzen: *Stochastic processes*. SIAM (1999)

[52] C. Hesse: *Angewandte Wahrscheinlichkeitstheorie*. Vieweg (2003)

[53] M. Loève: *Probability theory*. Courier Dover Publications (2017)

[54] K. Hinderer: *Grundbegriffe der Wahrscheinlichkeitstheorie*. Springer (1972)

[55] A. Stuart and K. Ord: *Kendall's Advanced Theory of Statistics Volume I - Distribution Theory*. Edward Arnold (1994)

[56] E. Engvall and P. Perlmann: *Enzyme-Linked Immunosorbent Assay, Elisa: III. Quantitation of Specific Antibodies by Enzyme-Labeled Anti-Immunoglobulin in Antigen-Coated Tubes1*. The Journal of Immunology **109** (1), 129–135 (1972)

[57] P. Billingsley: *Probability and Measure*. John Wiley & Sons (1986)

[58] R. von Mises: *Mathematical Theory of Probability and Statistics*. Academic Press (1964)

[59] M. L. Stein (Editor): *Interpolation of spatial data: some theory for kriging*. Springer series in statistics. Springer, New York ; Berlin ; Heidelberg [u.a.]. Includes bibliographical references and index (1999)

[60] B. Matérn: *Spatial Variation*. SSBM **36** (2013)

[61] M. Abramowitz, I. A. Stegun, and R. H. Romer: *Handbook of mathematical functions with formulas, graphs, and mathematical tables* (1988)

[62] E. Solak, E. Murray-Smith, W. E. Leithead, D. Leith, and C. Rasmussen: *Derivative observations in Gaussian process models of dynamic systems*. Advances in neural information processing systems **15** (2002)

[63] D. J. C. MacKay: *Information Theory, Inference, and Learning Algorithms*. www.inference.org.uk (2003)

[64] S. Kullback and R. A. Leibler: *On Information and Sufficiency*. The Annals of Mathematical Statistics **22** (1), 79 – 86 (1951)

[65] S. Kullback: *Information theory and statistics. john riley and sons*. Inc. New York (1959)

[66] T. Cover and J. Thomas: *Elements of Information Theory*. John Wiley & Sons (1991)

[67] D. Jones: *A Taxonomy of Global Optimization Methods Based on Response Surfaces*. Journal of Global Optimization **21**, 345383 (2001)

[68] A. D. Becke: *Density-functional exchange-energy approximation with correct asymptotic behavior*. Physical Review A **38**, 3098–3100 (1988)

[69] J. P. Perdew: *Density-functional approximation for the correlation energy of the inhomogeneous electron gas*. Physical Review B **33**, 8822–8824 (1986)

[70] F. Weigend and R. Ahlrichs: *Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy*. Physical Chemistry Chemical Physics **7**, 3297–3305 (2005)

[71] *TURBOMOLE V7.3 2018, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from http://www.turbomole.com.*

[72] S. Metz, J. Kästner, A. A. Sokol, T. W. Keal, and P. Sherwood: *ChemShell—a modular software package for QM/MM simulations*. WIREs Computational Molecular Science **4**, 101–110 (2014)

[73] M. Kirchhof, K. Gugeler, F. R. Fischer, M. Nowakowski, A. Bauer et al.: *Experimental and Theoretical Study on the Role of Monomeric vs Dimeric Rhodium Oxazolidinone Norbornadiene Complexes in Catalytic Asymmetric 1,2- and 1,4-Additions*. Organometallics **39**, 3131–3145 (2020)

[74] R. Meyer and A. W. Hauser: *Geometry optimization using Gaussian process regression in internal coordinate systems*. Journal of Chemical Physics **152**, 084112 (2020)

[75] M. Schonlau, W. J. Welch, and D. R. Jones: *Global versus local search in constrained optimization of computer models* (1998)

[76] M. Sasena, P. Y. Papalambros, and P. Goovaerts: *Metamodeling sampling criteria in a global optimization framework*

[77] H. J. Kushner: *A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise*. Journal of Basic Engineering **86** (1), 97–106 (1964)

[78] C. Perttunen: *A computational geometric approach to feasible region division in constrained global optimization*. *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 585–590 vol.1 (1991)

[79] J. Elder: *Global R/sup d/ optimization when probes are expensive: the GROPE algorithm*. *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 577–582 vol.1 (1992)

[80] J. Mockus: *Application of Bayesian approach to numerical methods of global and stochastic optimization*. Journal of Global Optimization **4**, 347365 (1994)

[81] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani: *Predictive Entropy Search for Efficient Global Optimization of Black-box Functions*. *Advances in Neural Information Processing Systems*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, volume 27. Curran Associates, Inc. (2014)

[82] P. Hennig and C. J. Schuler: *Entropy Search for Information-Efficient Global Optimization*. Journal of Machine Learning Research **13** (6) (2012)

[83] D. J. C. MacKay: *Information-Based Objective Functions for Active Data Selection*. Neural Computation **4** (4), 590–604 (1992)

[84] N. Houlsby, F. Huszar, Z. Ghahramani, and J. Hernández-lobato: *Collaborative Gaussian Processes for Preference Learning*. *Advances in Neural Information Processing Systems*, edited by F. Pereira, C. Burges, L. Bottou, and K. Weinberger, volume 25. Curran Associates, Inc. (2012)

[85] T. P. Minka: *A family of algorithms for approximate Bayesian inference*. Ph.D. thesis, Massachusetts Institute of Technology (2001)

[86] W. W. Hager: *Updating the Inverse of a Matrix*. SIAM Review **31** (2), 221–239 (1989)

[87] S. Grimme, J. Antony, S. Ehrlich, and H. Krieg: *A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu*. The Journal of Chemical Physics **132** (15), 154104 (2010)

[88] D. H. Ackley: *A connectionist machine for genetic hillclimbing*. Ph.D. thesis, Carnegie Mellon University (1987)

[89] T. Bäck: *Artificial Landscapes*. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press (1996)

[90] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind: *Automatic Differentiation in Machine Learning: a Survey*. Journal of Machine Learning Research **18** (153), 143 (2018)

## Erklärung über die Eigenständigkeit der Dissertation

Ich versichere, dass ich die vorliegende Arbeit mit dem Titel *Machine-Learning Techniques for Geometry Optimization* 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 *Machine-Learning Techniques for Geometry Optimization* is entirely my own work except where otherwise indicated. Passages and ideas from other sources have been clearly indicated.

Stuttgart, der 27. März 2023

Daniel Born