DIPLOMARBEIT

Evolutionary and Gradient-Based Optimization Methods in Tensor LEED

Ausgeführt am Institut für Angewandte Physik
der Technischen Universität Wien
Wiedner Hauptstraße 8-10 / 134
1040 Wien

unter der Anleitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Michael Schmid,

Michele Riva, PhD

und

Dipl.-Ing. Alexander Michael Imre

durch

Paul HAIDEGGER
Matrikelnummer 11902091

Wien, 10. Dezember 2024

Paul Haidegger

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

LEED $I(V)$ is a quantitative extension to low-energy electron diffraction (LEED). For LEED $I(V)$ the intensity I of the diffracted electrons is measured as a function of the kinetic energy of the incident electron beam (V). The resulting $I(V)$ curves are then compared to calculated $I(V)$ curves using a Reliability factor (R factor).

The calculation of $I(V)$ curves is computationally expensive. Curves for an input structure can be calculated with a so-called full dynamic calculation (i.e., including multiple scattering effects). Based on this full dynamic calculation, the effect of small changes to the input can be evaluated using the perturbative tensor-LEED approach. We developed a new implementation of tensor-LEED in PYTHON and Google JAX. The program makes use of the information produced in the reference calculation by TensErLEED and ViPErLEED. While the original TensErLEED code could only calculate the delta amplitude for displacements on a fixed grid and in one direction at a time, the reworked code allows on-demand computation of the delta amplitudes, and consequently the intensity and the R factor, for arbitrary displacements.

The parameters for the calculated $I(V)$ curves include the real part of the interstitial potential of the crystal, geometric displacements, vibrational amplitudes, and the site occupations. After reducing the number of parameters based on symmetry and user constraints, the remaining parameter vector is normalized such that all parameters are in the range $[0, 1]$. The remaining vector is then used for the optimization of the R factor.

The main focus of this work is on methods for solving this optimization problem and testing the feasibility of various optimization strategies. Since the R factor surface is rather flat when far from the global minimum and forms a smooth basin close to it, the algorithm is split into two parts. A global search focuses on exploration and aims at finding the basin containing the global minimum. Then, a local search focuses on exploitation to reach the minimum within the basin.

For the global part of the optimization, we find an evolutionary-style algorithm to perform well, with partial resampling applied to stay within the bounds. For the local search we find the SLSQP (Sequential Least SQuares Programming) and L-BFGS-B (Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm with Bounds) algorithms to be most suitable. A scaling factor is introduced to address the problem of overshooting in the initial iterations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Zusammenfassung

LEED $I(V)$ ist eine quantitative Erweiterung der Niederenergie-Elektronenbeugung (LEED), bei der die Intensität I der gebeugten Elektronen in Abhängigkeit von der kinetischen Energie des einfallenden Elektronenstrahls (V) gemessen wird. Die resultierenden $I(V)$-Kurven werden anschließend mit berechneten $I(V)$-Kurven mithilfe eines Zuverlässigkeitsfaktors (R-Faktor) verglichen.

Die $I(V)$-Kurven einer gegebenen Kristallstruktur können durch eine sogenannte voll-dynamische Berechnung (d. h. unter Berücksichtigung von Mehrfachstreueffekten) ermittelt werden. Auf Basis dieser voll-dynamischen Berechnung kann der Einfluss kleiner Änderungen an der Eingabestruktur mit dem störungstheoretischen Tensor-LEED-Ansatz berechnet werden. Wir haben eine neue Implementierung von Tensor-LEED in Python und Google Jax entwickelt, die auf den in der Referenzberechnung von TensErLEED und ViPErLEED erzeugten Daten beruht. Während der ursprüngliche TensErLEED-Code Delta-Amplituden nur für Verschiebungen auf einem festen Gitter und entlang jeweils einer Richtung berechnen konnte, ermöglicht der überarbeitete Code die On-Demand-Berechnung der Delta-Amplituden sowie der Intensität und des R-Faktors für beliebige Verschiebungen.

Zu den Parametern für die Berechnung von $I(V)$-Kurven zählen der Realteil des interstitialen Potenzials des Kristalls, geometrische Verschiebungen, Vibrationsamplituden sowie die Besetzungswahrscheinlichkeiten der Gitterplätze. Nach der Reduktion der Parameteranzahl basierend auf Symmetrieüberlegungen und benutzerdefinierten Einschränkungen wird der verbleibende Parametervektor so normiert, dass alle Parameter im Bereich $[0, 1]$ liegen. Ein derart normierter Parametervektor wird anschließend zur Optimierung des R-Faktors verwendet.

Der Schwerpunkt dieser Arbeit liegt auf Methoden zur Lösung dieses Optimierungsproblems sowie auf der Evaluierung verschiedener Optimierungsstrategien. Da die Oberfläche des R-Faktors weit entfernt vom globalen Minimum relativ flach ist, während sie in der Nähe des Minimums ein glattes Tal bildet, wird der Algorithmus in zwei Phasen unterteilt. In der globalen Suche wird der Fokus auf die Exploration gelegt, um die Senke zu identifizieren, die das globale Minimum enthält. Danach folgt eine lokale Suche, die auf die Feinauswertung abzielt, um das Minimum innerhalb dieser Senke zu erreichen.

Für den globalen Optimierungsteil hat sich ein evolutionärer Algorithmus als effektiv er-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

wiesen, bei dem partielles Resampling eingesetzt wird, um den zulässigen Wertebereich der Parameter einzuhalten. Für die lokale Suche erweisen sich die Algorithmen SLSQP (Sequential Least SQuares Programming) und L-BFGS-B (Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm with Bounds) als am geeignetsten. Ein Skalierungsfaktor wird eingeführt, um das Problem des Überschießens in den Anfangsiterationen zu lösen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Contents

1. Introduction and Motivation 1
1.1. Low-Energy Electron Diffraction: LEED 1
1.2. LEED $I(V)$ 2

2. LEED Theory 4
2.1. TensErLEED 4
2.2. ViPErLEED 5
2.3. On-the-Fly Calculation 5
2.4. R factor 5
2.5. Symmetry 7
2.6. Parameter Space 7
2.7. Software Testing 8

3. Optimization Theory 10
3.1. CMA-ES 10
3.1.1. Multivariate Normal Distribution 10
3.1.2. The Algorithm 12
3.1.3. Clinamen2 14
3.2. L-BFGS-B 15
3.2.1. BFGS 15
3.2.2. L-BFGS 16
3.2.3. L-BFGS-B 16
3.3. SLSQP 17
3.3.1. Equality Constraints 17
3.3.2. Inequality Constraints 19
3.4. Optimization Code 20

4. Results and Discussion 21
4.1. Global Search 22
4.1.1. Single-Parameter Optimization Approach 22
4.1.2. Basin Hopping 22
4.1.3. CMA-ES 23
4.2. Bounds for Clinamen2 23
4.2.1. Penalizing Points Outside Bounds 23
4.2.2. Adaptive Penalization Based on Distance 24

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.2.3. Logistic Mapping 25
4.2.4. Resampling Points Outside Bounds 26
4.3. Local Search 28
4.4. Benchmarking 29

5. Conclusion 34

List of Figures 36

References 37

A. Tables 41

B. Copyright Clearances 42

C. Optimization Code 44

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

1. Introduction and Motivation

This work builds on the results and code by the same author discussed in a prior project thesis, Ref [1]. The chapters below on qualitative and quantitative low-energy electron diffraction (LEED) as well as the chapters 2.1 and 2.2 are partly adopted from Ref [1], as their validity is unchanged.

1.1. Low-Energy Electron Diffraction: LEED

LEED is a common technique in surface science employed for the determination of the surface structure of single-crystal samples. A simple diagram of the experimental setup can be seen in Figure 1.1. In LEED, a primary electron beam, originating from an electron gun, is directed onto the sample. The electron beam interacts with the surface atoms, where it is diffracted, and backscattered as a set of diffraction beams according to Bragg's law. After interacting with the surface, the backscattered electrons traverse a set of grids. The suppressor grid filters out the inelastically scattered electrons. The remaining electrons are accelerated onto a phosphor screen, where they interact to produce photons, thereby making the diffraction pattern visible. The pattern is then usually recorded by a CCD camera [2].

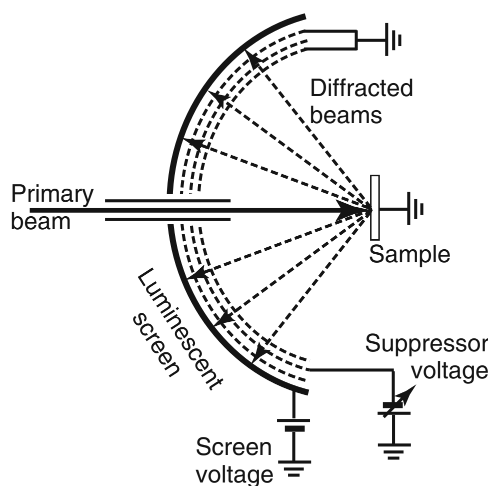

Figure 1.1.: Schematic drawing of the LEED experiment. Reprinted with permission from Ref. [3].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

In LEED, electrons with energies ranging from about approximately 25 eV to 1000 eV are used. Energies in this range are near the minimum of the so-called universal curve of the inelastic mean free path; there the mean free path is approximately 5–10 Å (see Figure 1.2). This makes the process very sensitive to signals from only the topmost atomic layers [3].

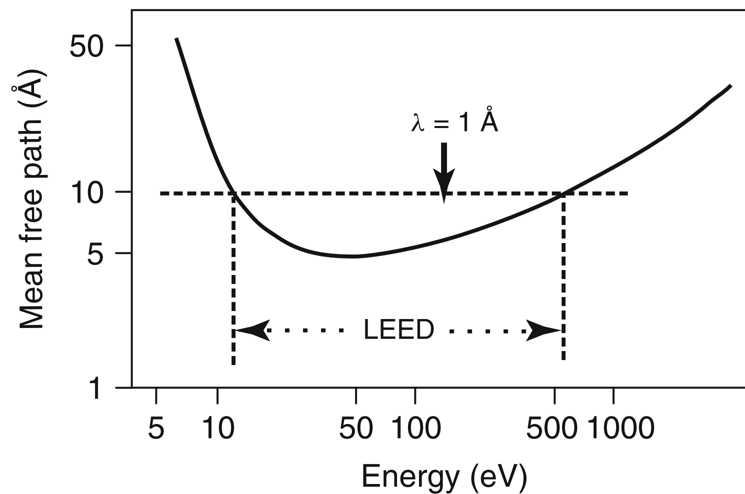

Figure 1.2.: Inelastic mean free path of electrons in solids. The arrow indicates where the de Broglie wavelength is 1 Å. Reprinted with permission from Ref. [3].

1.2. LEED $I(V)$

Qualitatively determining the shape and size of the surface unit cell and, therefore, the superlattice is relatively straightforward with LEED. However, determining the positions of the atoms within the unit cell from quantitative intensities (LEED $I(V)$) is much more challenging due to the significant impact of multiple-scattering effects. Multiple scattering occurs due to the strong interaction between electrons and atoms. LEED $I(V)$ measures the intensities, I, of the diffraction spots as a function of the energy, E, of the electrons, as determined by the acceleration voltage, V.

Because the measured $I(V)$ curves lack the phase information of the diffracted electron wave, there is no universal method to directly determine the surface structure from the $I(V)$ spectra. This limitation is also known as the famous “phase problem,” which affects many measurements in quantum physics. Instead, one is limited to “guessing” a potential

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

surface structure and computing the resulting $I(V)$ curves. When the calculated spectra agree with the experimental data, this indicates that the assumed surface structure is correct. One generally starts from a starting guess and then slightly alters the structural parameters. This leads to a high-dimensional optimization problem, where the complexity increases with the number of parameters. A problem arises when dealing with complex structures, as the computational effort increases drastically with the number of atoms in the unit cell [2, 4].

Using LEED-$I(V)$ has demonstrated great success in surface structure determinations. For example, by using the PYTHON package ViPErLEED (see Section 2.2) with the TensErLEED code (see Section 2.1), the structure of a surface-telluride phase with (10×10) periodicity on Pt(111) was analyzed [5], a so far unknown submonolayer phase of copper telluride with $(5 \times \sqrt{3})_{rect}$ was identified [6], and the structure-property relationship of MnO_x chains on Ir(001) was resolved [7].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2. LEED Theory

2.1. TensErLEED

In general most LEED codes calculate scattering using the so-called muffin-tin approximation, which treats atoms as point-scatterers. To calculate intensity spectra from a surface, a full-dynamic (i.e., including multiple-scattering effects) calculation must be performed. Due to the multiple scattering effects this calculation is computationally expensive. For that reason, it is not favorable to perform a full-dynamic LEED calculation for the $I(V)$ curves for optimization of all the atom positions within the unit cell of a complex surface structure. To reduce the computational effort, the so-called tensor-LEED approximation was introduced by Rous and Pendry [8]. The idea is to perform a full-dynamic LEED calculation for an unperturbed reference surface and to utilize a perturbation theory approach to compute the impact of small changes of the coordinates of the reference surface structure, as well as small changes in the vibration amplitudes, onto the resultant diffraction amplitudes and $I(V)$ curves. Using this approach, the computational effort scales only linearly with the number of perturbed atoms. The Erlangen tensor-LEED package (TensErLEED) is a well-established software package that utilizes the mentioned approximation to compute intensity spectra for a specified surface structure [4]. Additionally, it employs a search algorithm to optimize the surface structure to find the best fit with a set of measured intensity spectra [9]. The drawback of first-order perturbation theory is that it breaks down when encountering large geometric displacements to the atoms (>0.2 to 0.4 Å). Therefore, when facing significant displacements, a new time-consuming reference calculation must be performed [10].

Besides the geometrical and vibrational displacements, TensErLEED also allows changes in the real part of the interstitial potential of the crystal and the ratios of shared occupations [11, 12]. Any site in the crystal lattice may be statistically occupied by multiple chemical species, analogous to random substitutions in a bulk alloy. For each element that may be present in a given site, one occupational parameter is used to represent the fraction of that element occupying the site. The sum of the occupational parameters for one site cannot exceed unity, as it is physically impossible to have more than a fully occupied site. However, the sum of the fractional occupations for one site may be less than one to account for vacancies.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.2. ViPErLEED

ViPErLEED (Vienna package for Erlangen LEED) is a project by the TU Wien surface physics group for improving LEED-$I(V)$ methodology. It comprises both software and hardware components. Custom electronics are employed for conventional LEED measurements to generate LEED-$I(V)$ data, while the software can capture "movies" of the LEED patterns, and extract from them the $I(V)$ curves using the ImageJ based spot-tracker [13]. Additionally, ViPErLEED utilizes the PYTHON package VIPERLEED.CALC as a wrapper and major feature extension to the TensErLEED package for the computation of $I(V)$ spectra and structure optimization [14, 15].

2.3. On-the-Fly Calculation

In the TensErLEED code, amplitude changes due to the perturbation of the reference structure ("delta amplitudes") are computed for displacements along fixed axes and a fixed grid is used for the displacements and all other perturbations. Due to this limitation the part of the TensErLEED code that calculates the delta amplitudes was rewritten in PYTHON as part of this work and the preceding project thesis. The delta amplitudes are calculated by a function that takes arbitrary displacements, vibration amplitudes, occupations and changes to the real part of the interstitial potential as an input. Since the native PYTHON implementation has a significantly lower performance than the TensErLEED code written in FORTRAN, our PYTHON implementation uses performant numerical libraries like NUMPY [16] and JAX [17], which make heavy use of vectorization [1].

Based on these delta-amplitudes, intensity spectra can be calculated from the delta amplitudes and afterwards the R factor can be determined by comparison with the experimental $I(V)$ curves (see Section 2.4). Furthermore, the partial derivatives of the R factor with respect to the perturbations can be computed by using the automatic differentiation tool from JAX.

2.4. R factor

The ultimate objective of a tensor-LEED calculation is to find a combination of structural parameters that result in calculated intensities that closely resemble the experimental spectrum. Comparing theoretical and experimental spectra requires a quantitative measure, ideally a scalar. For this purpose, reliability factors (R factors) are introduced, to quantify the degree of similarity between two sets of spectra. In general, the lower the number the better the agreement. An effective R factor should exhibit high sensitivity to peak

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

positions, be insensitive to absolute intensity, but sensitive to relative intensities [3, 18].

Multiple such R factors were proposed in the literature. The most simple ones are the R_1 and R_2 factors defined as

$$R_1 = A_1 \int |I_\mathrm{e} - cI_\mathrm{t}| dE, \tag{2.1}$$

$$R_2 = A_2 \int (I_\mathrm{e} - cI_\mathrm{t})^2 dE, \tag{2.2}$$

with

$$c = \int I_\mathrm{e} dE / \int I_\mathrm{t} dE, \tag{2.3}$$

$$A_1 = 1/\int I_\mathrm{e} dE, \tag{2.4}$$

$$A_2 = 1/\int I_\mathrm{e}^2 dE, \tag{2.5}$$

where I_e and I_t are the experimental and theoretical Intensities [19]. R_1 and R_2 are primarily of historic interest and are not widely used at the time of writing. Instead, most studies use the Zanazzi-Jona and Pendry R factors

$$R_\mathrm{ZJ} = A_\mathrm{ZJ} \int \frac{|I_\mathrm{e}'' - cI_\mathrm{t}''||I_\mathrm{e}' - cI_\mathrm{t}'|}{|I_\mathrm{e}'| + \max|I_\mathrm{e}'|} dE, \tag{2.6}$$

$$R_\mathrm{P} = \frac{\int (Y_\mathrm{e} - Y_\mathrm{t})^2 dE}{\int (Y_\mathrm{e}^2 + Y_\mathrm{t}^2) dE}, \tag{2.7}$$

with

$$A_\mathrm{ZJ} = 1/(0.027 \int I_\mathrm{e} dE), \tag{2.8}$$

$$Y = \frac{L}{(1 + V_\mathrm{0i}^2 L^2)}, \tag{2.9}$$

where L stands for the logarithmic derivative $L = \mathrm{d}\ln(I)/\mathrm{d}E$ of the intensity and $V_{0\mathrm{i}}$ denotes the imaginary part of the inner potential of the crystal [19]. The Zanazzi-Jona R factor was widely used in the past. It has, however, been replaced by the Pendry R factor, which tends to be the default choice in LEED-$I(V)$ today. The Pendry R factor ranges from 0 to 2, where 0 indicates perfect agreement. It should be noted that the R factor optimization discussed in Chapter 4 exclusively utilizes the Pendry R factor.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.5. Symmetry

The structural symmetries of all periodic structures can be characterized by a finite set of symmetry groups. In three dimensions, there are 230 space groups. Any two-dimensional periodic structure, such as a crystal surface structure can be described by two linearly independent translations and is described by one of the 17 wallpaper groups, which are the 2D equivalents to space groups. The possible symmetries include translation, rotation, reflection, and glide reflection. For rotational symmetry, only 2-, 3-, 4-, and 6-fold rotations are allowed, according to the crystallographic restriction theorem [20]. However, quasi-crystals are an exception of this rule, where for example 5-fold rotations are possible. As shown in Fig. 2.1, each wallpaper group has a fixed number of symmetries, which can be any combination of the aforementioned types. The right panel displays the locations of all symmetry axes and planes, while the left panel illustrates example atoms and how their displacements are influenced by these symmetries.

The ViPErLEED code package is capable of identifying the correct symmetry group as well as determining the symmetry centers and vectors for a surface structure [14]. These symmetries constrain the movements of atoms and groups of atoms, effectively reducing the total number of free parameters for the system. The code automatically adjusts and reduces the parameter space based on the corresponding symmetries.

2.6. Parameter Space

The parameter space for optimization of the R factor by tensor LEED is spanned by the real part of the interstitial potential of the crystal, geometric displacements and vibrational amplitudes, and occupational parameters. Symmetries reduce the parameter space, as any symmetry-linked atoms can only be perturbed in symmetry-conserving manners. Additionally, the user can apply further constraints to reduce the parameter space. For instance, specific atoms can be fixed in place, allowing adjustments only to those atoms that have the greatest influence on the $I(V)$ curves.

Furthermore, the parameters are bound to a specific interval to ensure the algorithm remains in the desired region and to prevent nonphysical behavior, such as negative vibration amplitudes or a total occupation outside the range $[0, 1]$. Additionally, bounds on the magnitude of geometric displacements can ensure that the algorithm stays within the region of first-order perturbation theory.

After reducing the parameter space and setting the intervals, the remaining parameters p_i

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

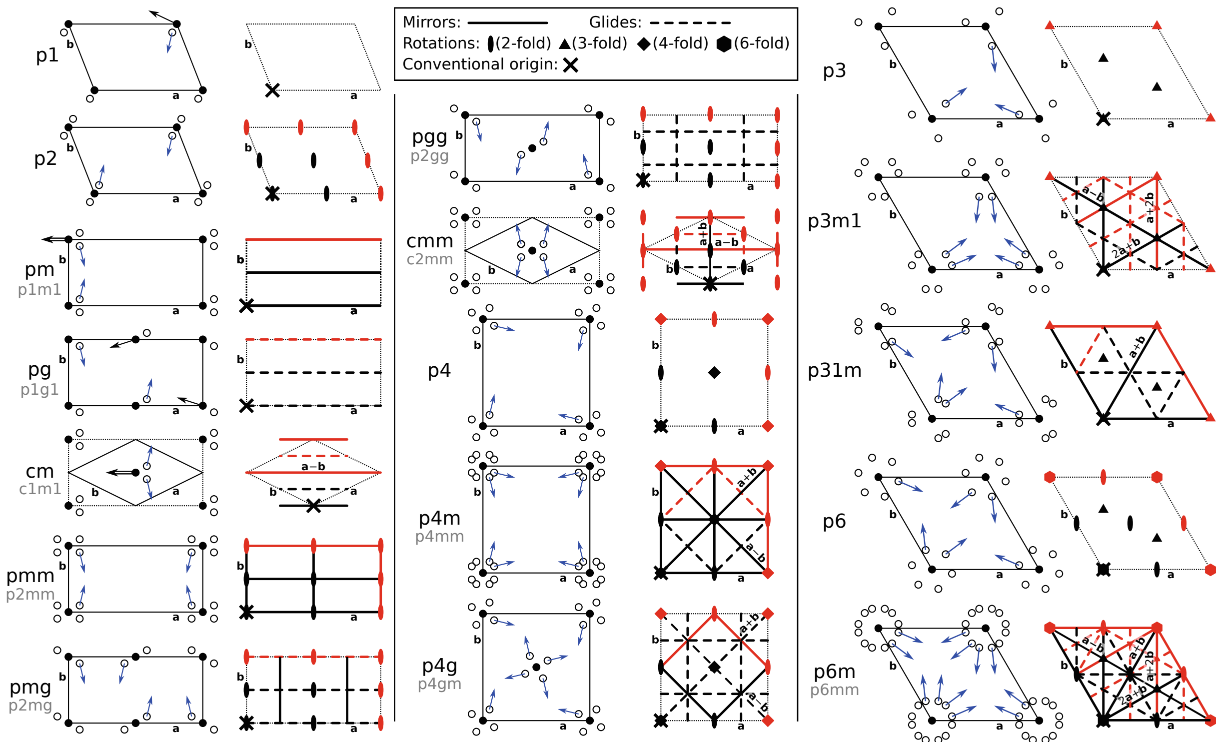

Figure 2.1.: All 17 plane symmetry groups are shown. The right side of each panel illustrates the positions of all symmetry elements, where red symmetry elements correspond to the black ones by a translation of integer multiples of the unit-cell basis vectors. The left panel shows examples of symmetry-equivalent atoms, indicated by filled and open circles. The blue and black arrows represent examples of linked in-plane displacements, while the double arrows indicate displacements restricted to the mirror plane. Figure from Ref. [14], used with permission by the authors.

are normalized to the range $p_i \in [0, 1]$. Parameter normalization improves the optimization process, particularly in gradient-based methods.

2.7. Software Testing

To ensure that any computer program runs as expected, it is generally best practice to write test cases for any piece of code. These tests are called unit tests when they focus on individual functions or small components, and integration tests when they focus on multiple parts at once to test the interaction between them. Software tests help to ver-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

ify that the program works correctly after development and can detect if any changes or refactoring introduce errors. In these tests, the program is run with inputs for which the correct outputs are known, and the program's results are compared against these known outputs. Since it is impossible to test every possible input, it is important to select representative and comprehensive test cases. One key criterion is to ensure sufficient code coverage, meaning that every branch of the program (such as conditional statements or loops) should be executed at least once, to cover the different paths the program might take. However, even if this criterion is met, not all errors will necessarily be found, as some may only occur with specific inputs [21].

It is generally better to test individual functions rather than testing the entire program at once. Testing simpler functions makes it easier to write tests and to isolate errors when they occur. To minimize undetected errors, it is important to follow some best practices when selecting test inputs. If a function accepts different data types, all should be tested. If the input is constrained to a specific interval, tests should include the endpoints for the interval as well as points within it. Certain values, such as zero, one, or very small numbers, are typical cases that can lead to unexpected behavior. However, depending on the specific function, other values may also trigger unusual outcomes. Therefore, it is important to identify and test any inputs that could cause edge cases or special conditions. For functions that take array-valued inputs, it is important to check the dimensions of both input and output arrays. Additionally, it is best practice to test for invalid inputs, such as incorrect data types or values outside the allowed range [22].

It is worth noting that in scientific programming, due to time and resource constraints, it is common to prioritize writing tests for critical components of the program rather than exhaustively testing all possible cases.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3. Optimization Theory

This work focuses on locating the global minimum of the R factor (see Section 2.4). Since the dimension of the problem scales with the number of atoms in the unit cell, the optimization increases in difficulty with the complexity of the unit cell. In the following, different algorithms are discussed which aim to find the minimum of an objective function. It is worth mentioning that calculating the gradient of the R factor is significantly more computationally expensive than calculating the R factor itself. While L-BFGS-B (see Section 3.2) and SLSQP (see Section 3.3) are gradient-based algorithms, CMA-ES does not utilize the gradient.

3.1. CMA-ES

The following brief explanation of CMA-ES is derived from Ref. [23, 24]. CMA-ES (Covariance Matrix Adaptation Evolution Strategy) is a numerical optimization algorithm used to find the global minimum of non-linear, non-convex functions. In this work, Clinamen2 [25] is used as the PYTHON implementation of CMA-ES. For the algorithm, a number of points in the parameter space (also called individuals) are drawn from a multivariate normal distribution. Then, the values of the objective function (the function to be minimized) are evaluated, and based on these values, the multivariate normal distribution is updated. This process is repeated iteratively until the convergence criterion is fulfilled. A set of individuals drawn from the distribution is called a generation and the number of individuals in one generation is called population size λ. Below, the most important properties of the multivariate normal distribution are explained.

3.1.1. Multivariate Normal Distribution

The multivariate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ is determined by its mean, $\mathbf{m} \in \mathbb{R}^n$, and its (positive definite) covariance matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$, with n being the dimension of the parameter space. The covariance between two random variables X_1 and X_2 is a measure of the dependency between these parameters. It is defined as $\mathrm{cov}(X_1, X_2) = \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_2 - \mathrm{E}[X_2])]$, with $\mathrm{E}[X]$ being the expected value of X. The covariance matrix contains the covariances between all variables in the parameter space, where $C_{ij} = \mathrm{cov}(X_i, X_j)$. The diagonal entries correspond to the variances of the individual variables. The eigendecomposition of the covariance matrix is written as $\mathbf{C} = \mathbf{B}(\mathbf{D})^2\mathbf{B}^\mathrm{T}$, where $\mathbf{B}$ is an orthogonal matrix (rotation and reflection) and $\mathbf{D}$ is a diagonal matrix containing the eigenvalues. Thus, the square roots of the eigenvalues can be interpreted

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

as the standard deviation along the eigenvectors. Geometrically, the covariance matrix is represented by a hyperellipsoid, showing a surface of equal density $\{\mathbf{x} \in \mathbb{R}^n | \mathbf{x}^T \mathbf{C}^{-1} \mathbf{x} = 1\}$. Some examples for such ellipsoids are shown in Figure 3.1. The principal axes of the ellipsoids are along the eigenvectors and the squared lengths of the principal axes are given by the corresponding eigenvalues.

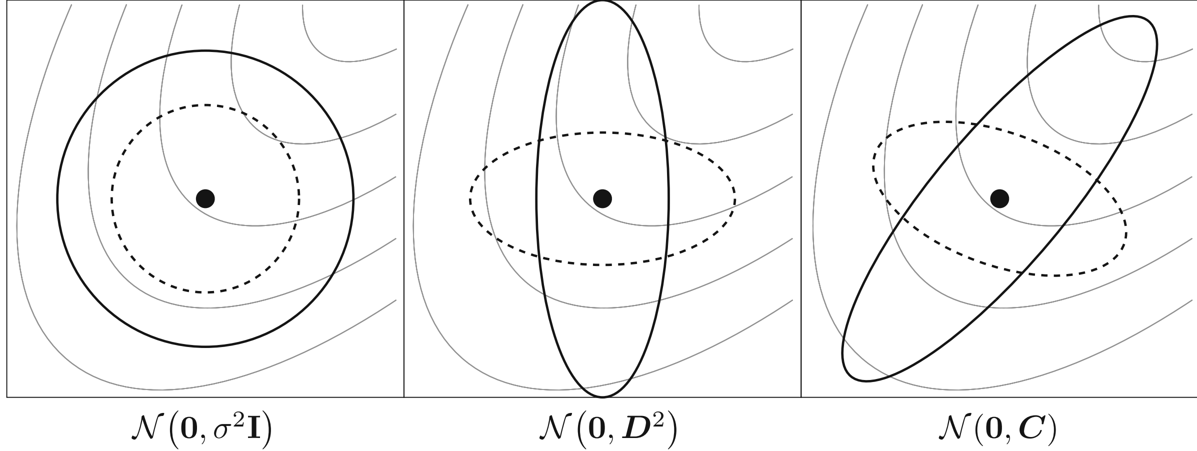

Figure 3.1.: The thick and dashed ellipsoids show the one-σ contour for six different, arbitrary distributions. $\mathbf{D}^2$ is a diagonal matrix and $\mathbf{C}$ is the covariance matrix. The thin lines are contour lines for an objective function. Figure from Ref. [24], used with permission by the author.

The multivariate normal distribution can be generated as

$$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{3.1}$$

with $\mathcal{N}(\mathbf{0}, \mathbf{I})$ being the standard normal distribution in each dimension and $\mathbf{I}$ being the identity matrix. "$\sim$" means equal distributions on both sides. Using $\mathbf{C}^{\frac{1}{2}} = \mathbf{B}\mathbf{D}\mathbf{B}^{\mathrm{T}}$, the fact that an orthogonal transformation does not change an isotropic distribution $\mathbf{B}^{\mathrm{T}}\mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathbf{D}\mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{D}^2)$, Eq. (3.1) can be rewritten as

$$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathbf{B}\mathcal{N}(\mathbf{0}, \mathbf{D}^2), \tag{3.2}$$

showing that the distribution is simply a normal distribution with the standard deviation in each direction corresponding to the length of one principal axis and an orthogonal matrix to align with these principal axes (plus a shift to set the mean at $\mathbf{m}$).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.1.2. The Algorithm

After the objective function has been evaluated for all individuals in one generation, the multivariate normal distribution is updated. Specifically, all the individuals with generation number $g+1$ are drawn from the multivariate normal distribution given by

$$\mathbf{x}_k^{(g+1)} \sim \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)}), \tag{3.3}$$

where $\mathbf{x}_k^{(g+1)}$ is the k-th individual from generation $g+1$. $\sigma^{(g)}$ is the overall standard deviation at generation g and is called step size. Note that the step size is defined differently from other algorithms, as it is a parameter for the width of the distribution and differs from the learning rate, which is introduced later. It should be mentioned that the mean and covariance matrix of the individuals $\mathbf{x}_k^{(g+1)}$ are $\mathbf{m}^{(g)}$ and $\mathbf{C}^{(g)}$. This notation is used to remain consistent with [24]. The drawn individuals are sorted by their function values $f(\mathbf{x}_{1:\lambda}^{(g+1)}) \leq f(\mathbf{x}_{2:\lambda}^{(g+1)}) \leq \cdots \leq f(\mathbf{x}_{\lambda:\lambda}^{(g+1)})$, with $\mathbf{x}_{i:\lambda}^{(g+1)}$ being the i-th best individual from generation $g+1$. The following three sections explain how $\mathbf{m}$, $\mathbf{C}$ and σ are updated.

Updating the Mean

Based on these sorted individuals, the first μ are selected ($\mu \leq \lambda$) to calculate the mean for the next generation

$$\mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + c_\mathrm{m} \sum_{i=1}^{\mu} w_i(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}), \tag{3.4}$$

$$\sum_{i=1}^{\mu} w_i = 1, \qquad w_1 \geq w_2 \geq \cdots \geq w_\mu \geq 0, \tag{3.5}$$

where w_i are the weights and c_m is the learning rate, which is most commonly set to 1. Note that $\mathbf{m}^{(g+1)}$ is the mean of the individuals $\mathbf{x}_k^{(g+2)}$ of the updated generation $g+2$, according to Eq. (3.3). If the weights decay very quickly, the majority of the information is derived from the first few points, while the last ones contribute almost no information. For this reason, an effective sample size $\mu_\mathrm{eff} = 1/\sum_{i=1}^{\mu} w_i^2$ is introduced. The effective sample size fulfills $1 \leq \mu_\mathrm{eff} \leq \mu$, with $\mu_\mathrm{eff} = \mu$ when the weights are equal. Clinamen2 uses the following equations for μ and the weights

$$\mu = \frac{\lambda}{2}, \tag{3.6}$$

$$w_i = \frac{\ln\frac{\lambda+1}{2i}}{\sum_{j=1}^{\mu}\ln\frac{\lambda+1}{2j}}, \tag{3.7}$$

where μ is always rounded down.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Updating the Covariance Matrix

The matrix,

$$\mathbf{C}_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})^{\mathrm{T}}, \tag{3.8}$$

can be used as a covariance matrix for the next generation. Note that this is not the covariance matrix of the best μ individuals of $\mathbf{x}_k^{(g+1)}$, since $\mathbf{m}^{(g)}$ is the mean of all individuals $\mathbf{x}_k^{(g+1)}$. This matrix aims to generate "better" individuals than the last one, by increasing the distribution width along the direction of descent (σ is set to 1 for Eq. (3.8)). Note that $\mathbf{C}^{(g+1)}$ is the covariance matrix for the individuals $\mathbf{x}_k^{(g+2)}$. This update rule can be used when the population is large enough to reliably estimate the covariance matrix. However, for a fast optimization the population size needs to be small, so the rank-μ update is used

$$\mathbf{C}^{(g+1)} = (1 - c_\mu \sum_{i=1}^{\lambda} w_i)\mathbf{C}^{(g)} + c_\mu \sum_{i=1}^{\lambda} w_i \mathbf{y}_{i:\lambda}^{(g+1)} (\mathbf{y}_{i:\lambda}^{(g+1)})^{\mathrm{T}}, \tag{3.9}$$

$$\mathbf{y}_{i:\lambda}^{(g+1)} = (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})/\sigma^{(g)}, \tag{3.10}$$

which uses the information from the previous generations to better estimate the covariance matrix, with $c_\mu \leq 1$ being another learning rate. The sum in Eq. (3.9) runs over all individuals and not just the selected ones. As a result, the weights must satisfy the new conditions $w_1 \geq \cdots \geq w_\mu > 0 \geq w_{\mu+1} \geq \cdots \geq w_\lambda$, which means that negative weights for $i > \mu$ are possible to include more information. The weights are usually chosen to satisfy the conditions $\sum_{i=1}^{\mu} w_i = 1$ and $\sum_{i=1}^{\lambda} w_i \approx 0$. Clinamen2 does not use negative weights. All weights for $i > \mu$ are set to zero.

The evolution path,

$$\mathbf{p}_{\mathrm{c}}^{(g+1)} = (1 - c_{\mathrm{c}})\mathbf{p}_{\mathrm{c}}^{(g)} + \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})\mu_{\mathrm{eff}}} \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{c_{\mathrm{m}}\sigma^{(g)}}, \tag{3.11}$$

provides information on how the mean of the generations move in the parameter space, thus capturing the correlation between successive steps. It uses exponential smoothing, meaning the most recent step has the greatest influence and the influence decays with descending generation number g. The evolution path starts at $\mathbf{p}_{\mathrm{c}}^{(0)} = \mathbf{0}$ and $1/c_{\mathrm{c}}$ is the backward time horizon of the evolution path with $c_{\mathrm{c}} \leq 1$ (for more information, see [24]). The rank-one update of the covariance matrix is given by

$$\mathbf{C}^{(g+1)} = (1 - c_1)\mathbf{C}^{(g)} + c_1 \mathbf{p}_{\mathrm{c}}^{(g+1)} (\mathbf{p}_{\mathrm{c}}^{(g+1)})^{\mathrm{T}}, \tag{3.12}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

where c_1 is yet another learning rate. The CMA update combines the rank-μ update and the rank-one update

$$\mathbf{C}^{(g+1)} = (1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i)\mathbf{C}^{(g)} + c_1 \mathbf{p}_\mathrm{c}^{(g+1)} (\mathbf{p}_\mathrm{c}^{(g+1)})^\mathrm{T} + c_\mu \sum_{i=1}^{\lambda} w_i \mathbf{y}_{i:\lambda}^{(g+1)} (\mathbf{y}_{i:\lambda}^{(g+1)})^\mathrm{T}. \quad (3.13)$$

Updating the Step Size

Eq. (3.13) does not result in an optimal step size. (Remember that the step size determines the width of the distribution.) Therefore, an additional update rule is required, which is based on the evolution path. When the evolution path is shorter than its expected length under random selection, the single steps are anti-correlated, which means the step size should be decreased. When the evolution path is longer than expected, single steps are correlated (meaning they go in the same direction). Thus, the step size should be increased, to reduce number of steps needed.

The limited number of individuals in Eq. (3.4) leads to statistical fluctuations of the mean, which impact the evolution path. These fluctuations are larger in the directions with larger standard deviation. To compensate for this, the formula for the conjugate evolution path can be slightly modified

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_\mathrm{eff}}\,\mathbf{C}^{(g)^{-\frac{1}{2}}} \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{c_\mathrm{m}\sigma^{(g)}}, \quad (3.14)$$

with $1/c_\sigma$ being the backward time horizon of the evolution path with $c_\sigma \leq 1$. $\mathbf{C}^{(g)^{-\frac{1}{2}}}$ is defined as $\mathbf{C}^{(g)^{-\frac{1}{2}}} = \mathbf{B}^{(g)}\mathbf{D}^{(g)^{-1}}\mathbf{B}^{(g)^\mathrm{T}}$. The conjugate evolution path under random selection is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The update rule for the step size is given by

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_\sigma^{(g+1)}||}{\mathrm{E}||\mathcal{N}(\mathbf{0}, \mathbf{I})||} - 1\right)\right), \quad (3.15)$$

where $d_\sigma \approx 1$ is a damping parameter and $\mathrm{E}||\mathcal{N}(\mathbf{0}, \mathbf{I})||$ is the expectation of the Euclidean norm of a vector drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Eq. (3.15) ensures that the step size increases if the conjugate evolution path is larger than the expected fluctuation from a random selection. On the other hand, if the mean is close to the minimum the μ best individuals in Eq. (3.4) scatter less than μ_eff individuals of the full generation. Therefore, the step size decreases.

3.1.3. Clinamen2

Clinamen2 is a Python implementation of CMA-ES used in the code for this thesis. Clinamen2 employs the combination of the rank-μ and rank-one update for the covari-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

ance matrix according to Eq. (3.13). Furthermore, the storage and computation of the covariance matrix is avoided by updating the Cholesky factor $\mathbf{A}$ with $\mathbf{C} = \mathbf{A}\mathbf{A}^{\mathrm{T}}$. Clinamen2 proved to be successful in identifying low-energy cluster configurations of Ag_5, Ag_6, and Ag_7 as well as to identify defects in stoichiometric Si supercells by interfacing to density-functional-theory codes or neural-network force fields [25]. The default values for the parameters c_{m} (3.4), c_μ (3.9), c_{c} (3.11), c_1 (3.12), c_σ (3.14) and d_σ (3.15) are defined as

$$c_{\mathrm{m}} = 1, \tag{3.16}$$

$$c_1 = \frac{2}{(n+1.3)^2 + \mu_{\mathrm{eff}}}, \tag{3.17}$$

$$c_\mu = \min\left(1 - c_1, \frac{2(\mu_{\mathrm{eff}} - 2 + 1/\mu_{\mathrm{eff}})}{(n+2)^2 + \mu_{\mathrm{eff}}}\right), \tag{3.18}$$

$$c_{\mathrm{c}} = \frac{4 + \mu_{\mathrm{eff}}/n}{n + 4 + 2\mu_{\mathrm{eff}}/n}, \tag{3.19}$$

$$c_\sigma = \frac{\mu_{\mathrm{eff}} + 2}{n + \mu_{\mathrm{eff}} + 5}, \tag{3.20}$$

$$d_\sigma = 1 + c_\sigma + 2 \cdot \max\left(0, \sqrt{\frac{\mu_{\mathrm{eff}} - 1}{n+1}} - 1\right) \tag{3.21}$$

with n being the dimension of the parameter space.

3.2. L-BFGS-B

3.2.1. BFGS

The Newton method is a well-known search algorithm that aims to find the minimum of an objective function. In each step, the algorithm approximates the objective function at the current point by a quadratic function and moves toward the minimum of the quadratic function. The vector $\mathbf{p}$ which points from the initial point $\mathbf{x}$ to the minimum of this hyper-parabola is given by

$$\mathbf{p} = -(\mathbf{H}(\mathbf{x}))^{-1}\nabla f(\mathbf{x}), \tag{3.22}$$

where $\mathbf{H}(\mathbf{x})$ is the Hessian matrix at the point $\mathbf{x}$. A problem arises when the Hessian matrix is not positive definite as the vector $\mathbf{p}$ may not point towards a minimum. It should be mentioned that we cannot calculate the Hessian analytically.

The next part is derived from Ref. [26] chapter 6.1. In the quasi-Newton method, the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Hessian matrix is not calculated for every step. Instead, an approximation of the Hessian matrix $\mathbf{B}$ is used, which is updated at each iteration.

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is a modified version of the Newton method. In BFGS, the search direction $\mathbf{p}_k$ is computed by using an approximation of the Hessian in Eq. (3.22). A line search is then performed in the direction of $\mathbf{p}_k$ to calculate the step size α_k that satisfies the so-called Wolfe conditions [27, 28]

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k (\nabla f_k)^{\mathrm{T}} \mathbf{p}_k, \tag{3.23a}$$

$$(\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k))^{\mathrm{T}} \mathbf{p}_k \geq c_2 (\nabla f_k)^{\mathrm{T}} \mathbf{p}_k, \tag{3.23b}$$

with $0 < c_1 < c_2 < 1$. The first condition provides an upper limit for α_k and the second condition a lower limit. After finding the next point $\mathbf{x}_{k+1}$, the approximation of the Hessian is updated

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} \mathbf{B}_k}{\mathbf{s}_k^{\mathrm{T}} \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k}, \tag{3.24}$$

with

$$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_k \mathbf{p}_k, \tag{3.25}$$

$$\mathbf{y}_k = \nabla f_{k+1} - \nabla f_k. \tag{3.26}$$

This leaves the initial approximation $\mathbf{B}_0$ as a free choice. The implementation in SciPy [29] used in this work uses the identity matrix.

3.2.2. L-BFGS

This section is a brief summary of the difference between the L-BFGS (Limited memory BFGS) and the BFGS. For more detailed comparison, see Ref. [30]. In the L-BFGS algorithm, instead of storing the entire approximate Hessian matrix $\mathbf{B}$, which is an $n \times n$ matrix, the vectors $\mathbf{s}_k$ and $\mathbf{y}_k$ from the last m iterations are stored in $n \times m$ matrices. In the literature, this version of the algorithm is motivated by saving storage space for large dimensions n. In practice, the bigger problem is the requirement of probing sufficiently many data points for calculating the $n(n+1)/2$ elements of the Hessian.

3.2.3. L-BFGS-B

This section is based on Refs. [30] and [26] chapter 16.7. For optimization problems with boundary conditions, there exists a further variant of the L-BFGS algorithm, namely L-BFGS-B (L-BFGS with bounds). The problem can be defined as

$$\min f(\mathbf{x}), \tag{3.27a}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

$$\text{subject to } \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \tag{3.27b}$$

with $\mathbf{l}$ being the lower and $\mathbf{u}$ being the upper bound. Every component x_i of $\mathbf{x}$ can have different bounds l_i and u_i.

In the first step, the search direction is defined by the negative gradient $\mathbf{g}$ at the starting point $\mathbf{x}_k$. Each time a component reaches its bound, it is fixed to that bound. As a result, the path is defined as a piecewise-linear path

$$\mathbf{x}(t) = \mathbf{P}(\mathbf{x} - t\mathbf{g}, \mathbf{l}, \mathbf{u}), \tag{3.28}$$

with the components of $\mathbf{P}$ defined as

$$P(x_i, l_i, u_i)_i = \begin{cases} l_i & \text{if } x_i < l_i \\ x_i & \text{if } x_i \in [l_i, u_i] \\ u_i & \text{if } x_i > u_i. \end{cases} \tag{3.29}$$

The Cauchy point is the first local minimum along this path $\mathbf{x}^c$. After finding the Cauchy point, all the variables that are at the bounds are held fixed and the quadratic problem over the subspace of the remaining variables, called free variables, is solved. All the free variables, which are outside of the bounds, are set to the bounds to get an approximate solution $\overline{\mathbf{x}}_{k+1}$. Then, a line search along $\mathbf{d}_k = \overline{\mathbf{x}}_{k+1} - \mathbf{x}_k$ is performed, which has to satisfy the Wolfe conditions [27, 28] and remain within the bounds, to find the new starting point for the next iteration, $\mathbf{x}_{k+1}$.

3.3. SLSQP

In this section, the motivation behind the SLSQP (Sequential Least SQuares Programming) algorithm is explained, which is derived from Ref. [31] and [26] chapter 18.1.

3.3.1. Equality Constraints

First, we consider the problem with additional equality constraints

$$\min f(\mathbf{x}), \tag{3.30a}$$

$$\text{subject to } \mathbf{c}(\mathbf{x}) = \mathbf{0}, \tag{3.30b}$$

where $\mathbf{c}(\mathbf{x})$ is a vector describing the constraints. In principle, the optimization in this work involves inequality constraints only. Equality constraints are used for parameters

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

that have reached a bound (the active set, see the next section). To solve the problem with the additional constraints the Lagrangian function $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{c}(\mathbf{x})$, where $\boldsymbol{\lambda}$ are the Lagrange multipliers, is used. An extremum of the Lagrangian in the extended space corresponds to an extremum of f, subject to the equality constraints. The gradient of the Lagrangian with respect to a vector containing the components of both, $\mathbf{x}$ and $\boldsymbol{\lambda}$ is given by

$$\nabla\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\mathbf{x}) - \mathbf{A}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\lambda} \\ \mathbf{c}(\mathbf{x}) \end{bmatrix}, \tag{3.31}$$

where $\mathbf{A}(\mathbf{x})$ denotes the Jacobian matrix of the constraints

$$\mathbf{A}(\mathbf{x})^{\mathrm{T}} = [\nabla c_1(\mathbf{x}), \nabla c_2(\mathbf{x}), ..., \nabla c_m(\mathbf{x})]. \tag{3.32}$$

Note that only $\mathcal{L}$ has a gradient with respect to $\boldsymbol{\lambda}$. The gradient of the other functions, such as f, is with respect to $\mathbf{x}$ only. By using the second derivative of the Lagrangian

$$\nabla^2\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla^2_{xx}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) & -\mathbf{A}(\mathbf{x})^{\mathrm{T}} \\ \mathbf{A}(\mathbf{x}) & \mathbf{0} \end{bmatrix}, \tag{3.33}$$

the iteration step

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \boldsymbol{\lambda}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{k} \\ \boldsymbol{\lambda}_{k} \end{bmatrix} + \begin{bmatrix} \Delta\mathbf{x}_{k} \\ \Delta\boldsymbol{\lambda}_{k} \end{bmatrix}, \tag{3.34}$$

can be computed using Newton's method (see Eq. (3.22))

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k & -\mathbf{A}_k^{\mathrm{T}} \\ \mathbf{A}_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_{k} \\ \Delta\boldsymbol{\lambda}_{k} \end{bmatrix} = \begin{bmatrix} \nabla f_k - \mathbf{A}_k^{\mathrm{T}}\boldsymbol{\lambda}_k \\ \mathbf{c}_k \end{bmatrix}. \tag{3.35}$$

In summary, a quadratic approximation of the Lagrangian in the extended space is used to find $\Delta\mathbf{x}_k$ and $\Delta\boldsymbol{\lambda}_k$.

It can be shown [26] that solving the problem for the iteration step $\mathbf{p} = \Delta\mathbf{x}$

$$\min_{\mathbf{p}}(f_k + \nabla f_k^{\mathrm{T}}\mathbf{p} + \frac{1}{2}\mathbf{p}^{\mathrm{T}}\nabla_{xx}\mathcal{L}_k\mathbf{p}) \tag{3.36a}$$

$$\text{subject to } \mathbf{A}_k\mathbf{p} + \mathbf{c}_k = \mathbf{0} \tag{3.36b}$$

yields the same result as Eq. (3.35). Eq. (3.36a) is a rearranged form of the quadratic approximation to the Lagrangian, and Eq. (3.36b) is a linear approximation to the constraints. For the current work, the linear approximation is not necessary since only linear constraints are present.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.3.2. Inequality Constraints

With additional inequality constraints we consider the problem with a set $\mathcal{E}$ of equality constraints and a set $\mathcal{I}$ of inequality constraints as

$$\min f(\mathbf{x}), \tag{3.37a}$$

$$\text{subject to } c_i(\mathbf{x}) = 0,\ i \in \mathcal{E}, \tag{3.37b}$$

$$c_i(\mathbf{x}) \geq 0,\ i \in \mathcal{I}, \tag{3.37c}$$

which reduces to Eq. (3.27) with no equality constraints and the most simple form of inequality constraints, limits for the components of $\mathbf{x}$. To solve Eq. (3.37) the quadratic problem is solved iteratively using a linear approximation of the constraints

$$\min_{\mathbf{p}} f_k + \nabla f_k^{\mathrm{T}}\mathbf{p} + \frac{1}{2}\mathbf{p}^{\mathrm{T}}\nabla_{xx}\mathcal{L}_k\mathbf{p}, \tag{3.38a}$$

$$\text{subject to } \nabla c_i(\mathbf{x}_k)^{\mathrm{T}}\mathbf{p} + c_i(\mathbf{x}_k) = 0,\ i \in \mathcal{E}, \tag{3.38b}$$

$$\nabla c_i(\mathbf{x}_k)^{\mathrm{T}}\mathbf{p} + c_i(\mathbf{x}_k) \geq 0,\ i \in \mathcal{I}. \tag{3.38c}$$

Initially, a point $\mathbf{x}_0$ must be provided that satisfies all the constraints. From this point an active set, consisting of all the indices of the active constraints, is calculated as

$$\mathcal{A}_0 = \{i \in \mathcal{E}\} \cup \{i \in \mathcal{I} | c_i(\mathbf{x}_0) = 0\}. \tag{3.39}$$

In the next step the inactive constraints are ignored and the problem is solved with the active constraints only (see Eq. (3.35)). These active constraints are equality constraints, therefore Eq. (3.35) in Section 3.3.1 can be used to get $\mathbf{p}_k = \Delta\mathbf{x}_k$ and $\boldsymbol{\lambda}_{k+1}$ (by rearranging Eq. (3.35), $\boldsymbol{\lambda}_{k+1}$ can be calculated without prior knowledge of $\boldsymbol{\lambda}_k$). Then $\mathbf{p}_k$ is used as a search direction with the step size α_k, which must satisfy the following condition to ensure that the inequality conditions are not violated

$$\alpha \leq \hat{\alpha} = \begin{cases} \min \frac{c_i(\mathbf{x}_k) + \nabla c_i(\mathbf{x}_k)^{\mathrm{T}}\mathbf{x}_k}{\nabla c_i(\mathbf{x}_k)^{\mathrm{T}}\mathbf{p}_k}, & \text{if } \nabla c_i(\mathbf{x}_k)^{\mathrm{T}}\mathbf{p}_k < 0 \text{ for some } i \notin \mathcal{A}_k \\ +\infty, & \text{if } \nabla c_i(\mathbf{x}_k)^{\mathrm{T}}\mathbf{p}_k \geq 0 \text{ for all } i \notin \mathcal{A}_k. \end{cases} \tag{3.40}$$

After performing a line search to find a suitable α_k the new search point $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$ is computed and the active set $\mathcal{A}_k$ is updated. If $\alpha = \hat{\alpha}$ the additional constraints are added to the active set. If some of the Lagrange multipliers $\lambda_i(\mathbf{x}_k)$ associated with inequality conditions in the active set are negative, the smallest of them is removed from the active set.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

To enhance performance, SLSQP uses the same update rule for the Hessian as L-BFGS-B (see Eq. (3.24)). Note that SLSQP and L-BFGS-B use different line searches. One important difference is that the step size in the L-BFGS-B algorithm needs to fulfill the Wolfe conditions (see Eq. (3.23)), which requires calculating the gradient for each trial point in the line search. The SLSQP algorithm only requires the objective function to decrease in each step.

3.4. Optimization Code

The Python code for the search algorithm is presented in the Appendix C. The code is designed using multiple inheritance layers with an abstract base class (`Optimizer`), which defines a basic structure for all optimization algorithms. The second abstraction layer differentiates between gradient-based optimizers (`GradOptimizer`) and non-gradient optimizers (`NonGradOptimizer`). The concrete implementation consists of the different optimizers: `LBFGSBOptimizer` and `SLSQPOptimizer` from the `GradOptimizer` subclass represent local search methods (see Section 4.3), while the `CMAESOptimizer` from the `NonGradOptimizer` subclass implements the CMA-ES algorithm, used for the global search (see Section 4.1). The class `SequentialOptimizer` combines global and local optimizer to perform a full search.

`CMAESResult` is a class for the output of the `CMAESOptimizer` class to match the output of the local optimizer classes. The function `create_resample_and_evaluate` is used to implement partial resampling for the `CMAESOptimizer` to satisfy the boundary conditions (see Section 4.2).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4. Results and Discussion

The aim of this work is to identify the surface structure from a set of measured intensity spectra. This is achieved by locating the global minimum of the R factor (see Section 2.4). Finding this minimum is not an easy task, since the dimension of the problem scales with the number of atoms in the unit cell. Furthermore, the R factor surface exhibits a rough landscape with many local minima. The surface structure in two dimensions is shown in Fig. 4.1, with the rest of the parameters at the global minimum (left) and not minimized (right).

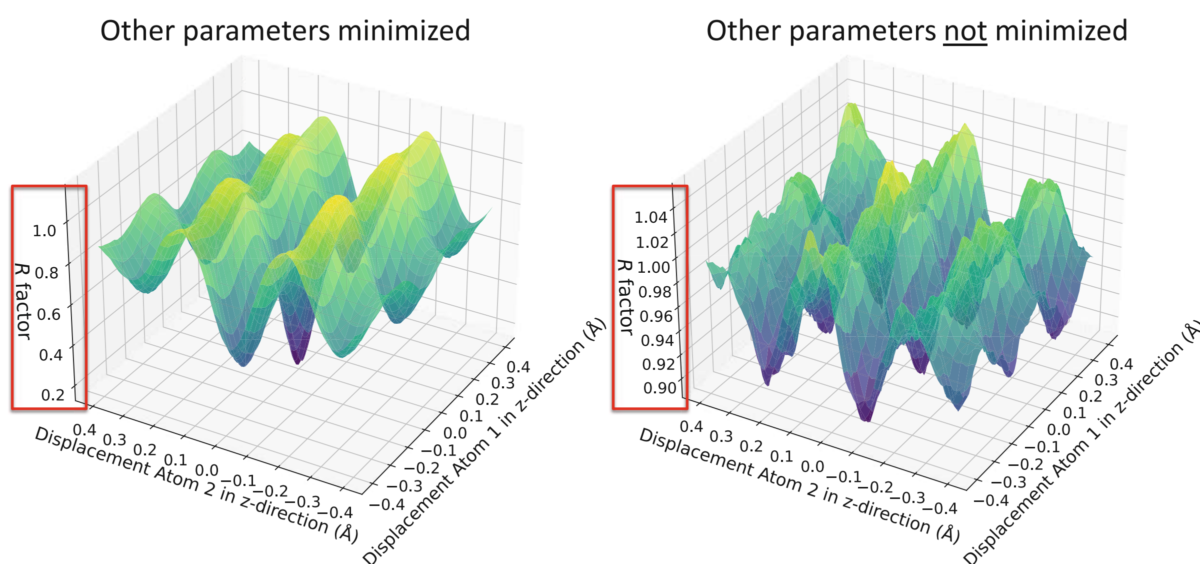

Figure 4.1.: Two-parameter R factor surface for the $Fe_2O_3(012) - (1 \times 1)$ structure. In the left plot, the remaining parameters are at the global minimum position, whereas in the right plot they are not. Notice the different scales of the vertical axes.

As Fig. 4.1 shows, a smooth basin can be identified near the global minimum, indicating that a local minimization algorithm started in that basin should be sufficient to locate the global minimum. Therefore, it is advantageous to split the search into two parts. A first optimization method aims to identify the correct basin, with the main focus on exploring the parameter space. Once the correct basin is located, a second algorithm focuses on exploiting the basin to converge toward the global minimum.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.1. Global Search

In the following, the algorithms tested for basin identification are discussed. Each algorithm was initially tested for a $Fe_2O_3(012)-(1\times1)$ system [14] with ten variable parameters (one for the potential, four vibrational parameters, and five geometric parameters), while the remaining parameters are fixed. If the algorithm proves successful, the number of variable parameters is increased.

4.1.1. Single-Parameter Optimization Approach

This algorithm is a simple approach and is based on the idea that most of the parameters may not be strongly correlated, thus making it possible to optimize one parameter at a time. Prior work [15] has already shown that there is a correlation between many of the optimization parameters. Nonetheless, we implemented this approach for comparison purposes, as it gives a suitable baseline and is relatively fast and simple to implement.

In the vast majority of runs, the algorithm was unable to identify the correct basin, indicating that it is not suitable for this problem.

4.1.2. Basin Hopping

Basin hopping is an established optimization method that consists of three parts. It assumes that the objective function is made up of multiple basins of various depth each of which can be fully exploited with a local optimization. First, a random perturbation of the coordinates is applied, then, a local minimization to determine the depth of the basin is performed, and finally, the new coordinates are either accepted or rejected based on the function value in the minimum. The SciPy version of basin hopping used in this work is implemented as described in Ref. [32].

This algorithm was tested because it has proven useful for objective functions with basin-like features, as shown in Fig. 4.1. Since the local minimization must be performed in every iteration, it is crucial for this process to be very fast. However, we found this approach to be infeasibly slow for the problem at hand. As a result, even after several hours and only ten variable parameters, the algorithm did not move very far in the parameter space and was therefore unable to find the global minimum in a reasonable time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.1.3. CMA-ES

The CMA-ES (see Section 3.1) algorithm has numerous hyperparameters that can be changed to refine the algorithm. The three most important ones for the optimization of the R factor surface are the number of generations, the population size, and the initial width of the distribution (named step size in CMA-ES). A higher number of generations does not necessarily make the algorithm better, as the algorithm explores in the beginning but shifts to exploiting in the last generations, when the step size becomes smaller. However, if the number of generations is too small the exploration is cut off before the correct basin is identified. The population size has to scale with the number of parameters. The default population size in Clinamen2 is $\lambda = 4 + 3\log(n)$, with n being the dimension of the problem [25]. Experience shows that for our problem a higher population size is needed. The step size equals the standard deviation in each direction during the first generation and gives a measure of the deviation of the distributions in the later generations. In other words, the initial step size can be seen as a measure of the exploration. In our tests, we found that a very large step size (0.5 to 0.6 for the normalized parameters) is crucial to find the global minimum.

After implementing bounds, which will be explained in the following section, this algorithm was able to reliably find the correct basin in a reasonable time. The stopping condition for the algorithm was defined such that the standard deviation of the minimum points from the last five generations must be smaller than 10^{-4}. However, since the aim of this algorithm is to focus on exploration, this condition is only satisfied if the number of generations is too high.

4.2. Bounds for Clinamen2

CMA-ES as implemented in Clinamen2, does not implement boundary conditions as they are required for the physically constrained optimization of the R factor. It is thus necessary to adapt the sampling method in a way that takes these boundary conditions into account. For all these methods, minimizing the number of function evaluations is essential, as they are the main factor in the runtime.

4.2.1. Penalizing Points Outside Bounds

A first simple approach to implementing boundary conditions is to penalize all the points outside the bounds by setting the R factor to a fixed high value (see Fig. 4.2(a)). The bounded function is augmented and a fixed high value is returned for any argument vector $\mathbf{x}$ outside the n-dimensional boundary box. To save execution time, the real function is

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

only evaluated if $\mathbf{x}$ is within the bounds. This version works if the starting point, as well as the minimum is located near the center of the box and the number of parameters and the initial step size are sufficiently small. However, if these conditions are not met, most of the sampled points lie outside of the box, which provides little useful information for the algorithm for the updates, and the mean drifts in seemingly random directions and outside of the box.

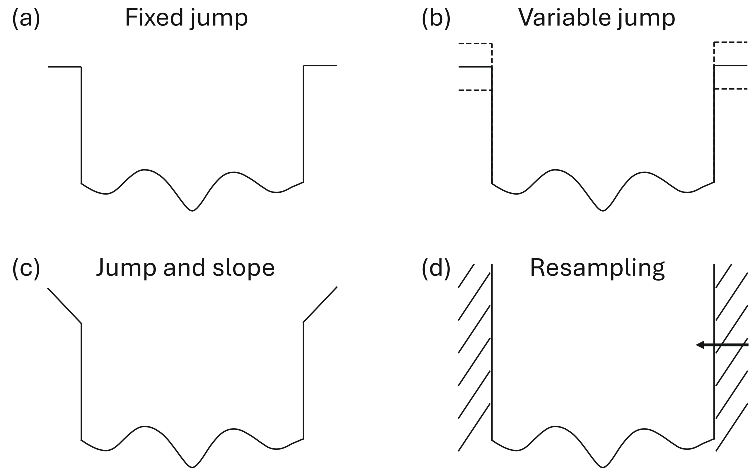

Figure 4.2.: Schematic drawing of the attempted methods to include boundaries. The R factor outside the bounds is set to a fixed high value (a), set to a high value proportional to the number of parameters outside of the box (b), or set to a high value plus a value proportional to the distance from the bounds (c). The fourth version was to resample all points outside the box, thus not allowing any points outside.

4.2.2. Adaptive Penalization Based on Distance

An improvement could be observed if the R factor, for points outside the box, is not set to a fixed value but rather to a value dependent on the point. One approach was to use a high value that is proportional to the number of variables outside the bounds (see Fig. 4.2(b)). Another possibility is to add a term proportional to the distance from the box to a fixed high value (see Fig. 4.2(c)). In these cases, the CMA-ES naturally drives the mean back to the inside of the hyper-box.

A common issue with these approaches is that for the individuals that are outside of the box, the CMA-ES cannot adapt to the objective function. Since the evaluation time

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

of these individuals is significantly shorter compared to that of individuals requiring an evaluation of the objective function, its impact can be considered negligible.

The problem arises when considering the initial step size of the algorithm. When starting with a large initial step size for the first few generations, almost all the individuals include at least one variable outside of the box. This leads to the algorithm significantly decreasing the step size to stay inside the box. During these first generations, the algorithm does not learn much about the objective function and the mean of the generations drifts around, but generally tends toward the center of the box. Proper sampling of the objective function only starts when the step size is small enough. In other words, this version gives an implicit upper limit for the step size, that gets smaller with more parameters. This, unfortunately, conflicts with the finding that a bigger initial step size is crucial for the algorithm to focus on exploring at the beginning to find the right basin.

One thing that should be mentioned here is that the adaptive penalization approach has been shown to be successful for most of the runs. Only for systems with the global minimum very close to the bounds, this approach tends to become unreliable.

4.2.3. Logistic Mapping

Another approach is to map all values to the range $(0, 1)$. This way, the algorithm does not require explicit bounds. For the mapping, the logistic function $f(x) = \frac{1}{1+e^{-x}}$ is used.

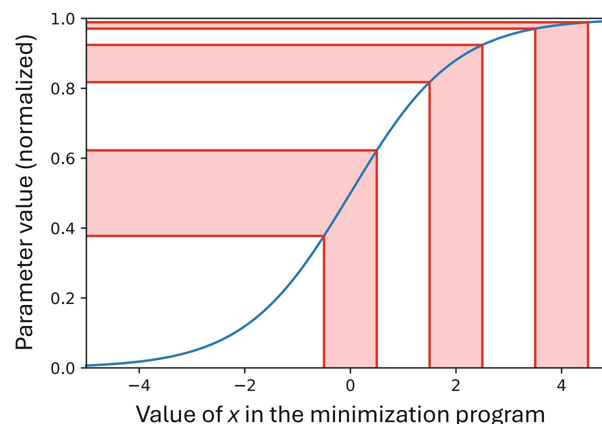

Figure 4.3.: Logistic function. The red areas show intervals of equal size on the x-axis and how the mapped intervals on the y-axis differ in length.

In Fig. 4.3, the logistic function and the mapping of three intervals are shown. As seen in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the figure, intervals of equal lengths on the x-axis correspond to different lengths on the y-axis. Since the algorithm starts in the middle, the step size (interpreted as intervals) on the y-axis decreases as the mean moves further from the starting point. This is a significant problem because, with a reasonable step size, the outer areas are not properly explored, and with larger step sizes, exploration in the center worsens. Therefore, this approach performed poorly in the tested cases.

4.2.4. Resampling Points Outside Bounds

A way to circumvent this issue is to re-sample any points that are drawn outside the boundaries (see Fig. 4.2(d)). Resampling means that if a point lies outside the bounds, a new point is drawn from the same multivariate normal distribution (see Eq. (3.3)). This process is repeated until the point lies within the bounds. While this ensures that all used points remain within the bounds and thus prevents interference with exploration, a downside of this method is that resampling alters the multivariate normal distribution (see Section 3.1.1), causing the mean to shift away from the bounds and the standard deviation to decrease. The difference between a distribution with and without resampling is shown in Fig. 4.4 in one dimension. The problem with this version is that even though the resampling is very fast, for large dimensions, a lot of resampling needs to take place, since the the number of resamples scales exponentially with the number of dimensions. For example, if the algorithm starts at the center of the box with a step size of 0.5 (which has been shown to be a very good starting condition) each component has a probability of approximately 0.68 of being sampled within the box. For 30 dimensions the probability of the entire point lying inside the box is 0.68^{30}. Using the geometric distribution, the expected number of resamples can be calculated as $1/0.68^{30} \approx 10^5$. This makes this version less suitable for our problem.

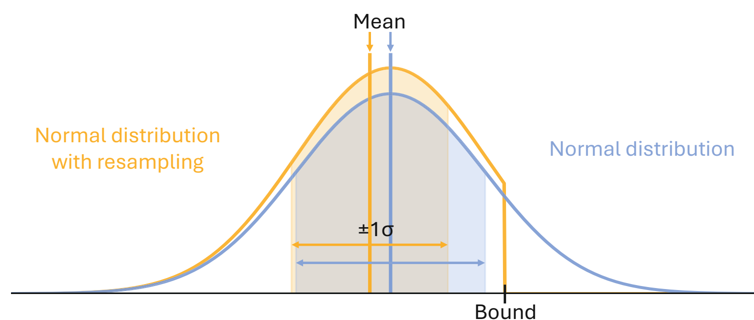

Figure 4.4.: Change in the normal distribution when a part is cut off, which happens when resampling values above a bound. The mean shifts and the standard deviation decreases.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The version used in this work uses partial resampling, which means it resamples if the point is outside of the box but only replaces the parameters that are outside. With partial resampling, the number of resamples scales only linearly with the number of dimensions. As long as the eigenvectors of the covariance matrix are not aligned with the unit vectors of the system, correlations exist between the parameters. Resampling only some components modifies the correlations for the components that are resampled.

Fig. 4.5 illustrates full and partial resampling. The eigenvectors of the partially resampled distributions correspond to the principal axes of the ellipse (see Section 3.1.1). For the partially resampled distribution, the ellipse is substantially rotated compared to the other two ellipsoids. This rotation results from the loss of correlation between the parameters. Furthermore, the ellipse of the partially resampled distribution is more circular due to an increase in the minor principal axis. This occurs because the major principal axis is projected onto the x-axis, leading to a larger standard deviation along the minor principal axis.

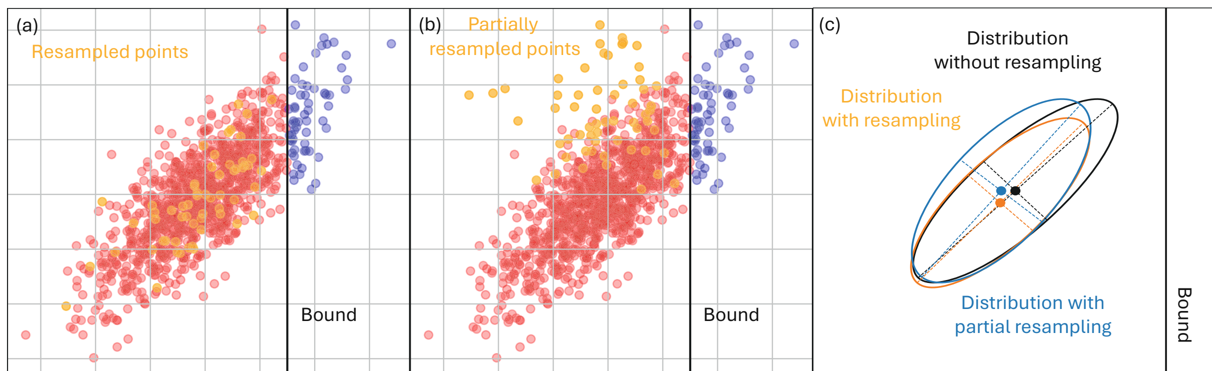

Figure 4.5.: Points are drawn from a multivariate normal distribution. The points outside the bounds (blue) are fully resampled (a) or partially resampled (b). The resulting effect on the distribution is depicted in the schematic drawing (c), where the ellipsoids show the one-σ contour for a multivariate distribution without resampling, one with full resampling, and one with partial resampling.

All methods to include bounds discussed above may encounter difficulties when the minimum is near the bounds. For that reason, we output a warning if a parameter is close to the bound, so that the user is made aware and can choose to change the respective bound(s) if necessary. Furthermore, this algorithm is primarily used to identify the right basin, so slight inaccuracies in the exact location of the minimum are tolerable. If the minimum found is outside the applicable range of the first-order perturbation theory approach, a new reference calculation should be performed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

For the majority of parameters the range can be adjusted by the user to avoid a global minimum near the bounds. Special care needs to be taken with regard to occupational parameters, as a value less than 0 or greater than 1 is unphysical. If the expected optimum value is close to 0 or 1, to prevent the aforementioned problem with a minimum very close to the bound, the user should consider using a fixed occupation in the initial search. Alternatively, one may consider bounds outside the 0–1 range in the initial search [33].

4.3. Local Search

In the following, the algorithms tested for local optimization are discussed. All implementations of the various algorithms are from the scipy.minimize library [29]. For the comparison, each algorithm is tested with six different starting points, all within the global minimum basin. One of these points is already positioned at the minimum. Note that Powell moved away from the global minimum even when starting at the global minimum.

Name	Mean R Factor	Avg. Time (s)	Success Count
COBYLA [34]	0.1665	65	1/6
Nelder-Mead [35]	0.1637	596	2/6
Powell [36]	N/A	N/A	0/6
L-BFGS-B	0.1574	2218	6/6
TNC [37]	0.1570	2429	6/6
Trust-Constr [38]	0.1585	2678	5/6
SLSQP	0.1574	233	5/6
Scaled SLSQP	0.1570	228	6/6

Table 4.1.: Comparison of different local minimization algorithms. The first three do not use a gradient, while the remaining four do. The R factor as well as the average time are only calculated for successful runs. The success count indicates the number of runs in which the algorithms successfully located the correct minimum.

Table 4.1 summarizes the performance of the different algorithms (see Appendix A for all data). The first three algorithms, which do not utilize the gradient, did not locate the minimum for a sufficient number of starting points. Among the algorithms using the gradient, the Trust-Constr algorithm performed worse in every category than the other gradient-based local optimization algorithms. L-BFGS-B (see Section 3.2) and TNC produced very similar outcomes. Since the L-BFGS-B algorithm is widely used and more common than TNC, the current work focuses on L-BFGS-B as the second-best algorithm, rather than

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TNC.

The SLSQP algorithm (see Section 3.3) is significantly faster than L-BFGS-B, due to its reduced use of the computationally expensive gradient and faster convergence. However, the SLSQP algorithm tends to overshoot in the initial steps. This is also the reason why the SLSQP algorithm failed to find the minimum in one instance in Table 4.1, as the overshooting moved it into a neighbouring basin, resulting in the minimum of the wrong basin being found.

Both the SLSQP and the L-BFGS-B algorithms employ the same approximation of the Hessian and both algorithms start with the identity matrix. Thus, the initial step of neither algorithm accounts for the curvature. After some iterations, the approximation improves and the algorithms converge. Both algorithms carry the risk of overshooting into a different basin during the initial iterations. However, due to differences in the line search and the absence of the Wolfe conditions, the SLSQP algorithm is more likely to move to a different basin.

To address the issue of overshooting in the SLSQP algorithm, the step size could be reduced, but this is currently not possible in the SciPy version. A more elegant solution would be to use a different starting matrix for the Hessian approximation instead of the identity. This could be achieved by running the L-BFGS-B algorithm for a few iterations and using the BFGS approximation as a starting point. However, the SciPy implementation does not, at the time of writing, allow an input for the starting matrix.

Instead, the current version uses a scaling factor that is applied to the entire function $f_{\text{scaled}} = d \cdot f$, and therefore also to the gradient $\nabla f_{\text{scaled}} = d \cdot \nabla f$, with d being the scaling factor. This effectively reduces the step size in the initial iterations, and once the Hessian approximation becomes sufficiently accurate, the algorithm converges to the minimum of the scaled function, which is also the minimum of the original function.

Altogether, we find the SLSQP and L-BFGS-B algorithms to perform best for the problem at hand. The scaled version of SLSQP generally appears to be the most efficient option for local minimization, with L-BFGS-B being a slower but somewhat more reliable alternative.

4.4. Benchmarking

For the benchmarking, our implementation was executed on the Vienna scientific cluster (VSC-5 [39]) using an A100 GPU. An interval for the geometric displacements is defined

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

such that the global minimum is not at the center of the box, simulating a practical case where the global minimum is within the box but not at the center. This setup allows both global and local search options to be tested. For this run, the real part of the interstitial potential, two vibrational, and 30 geometrical parameters are optimized. For this number of parameters a population size of around 30 has been shown to be adequate. As already mentioned, a large initial step size, set to 0.5 here, is important. For this step size, the number of generations is set to 100, which is a little on the lower side. However, as it can be seen in Fig. 4.6, this is still sufficient for this problem, as the evolutionary algorithm reaches a final R factor of around 0.2, indicating it is quite deep in the basin.

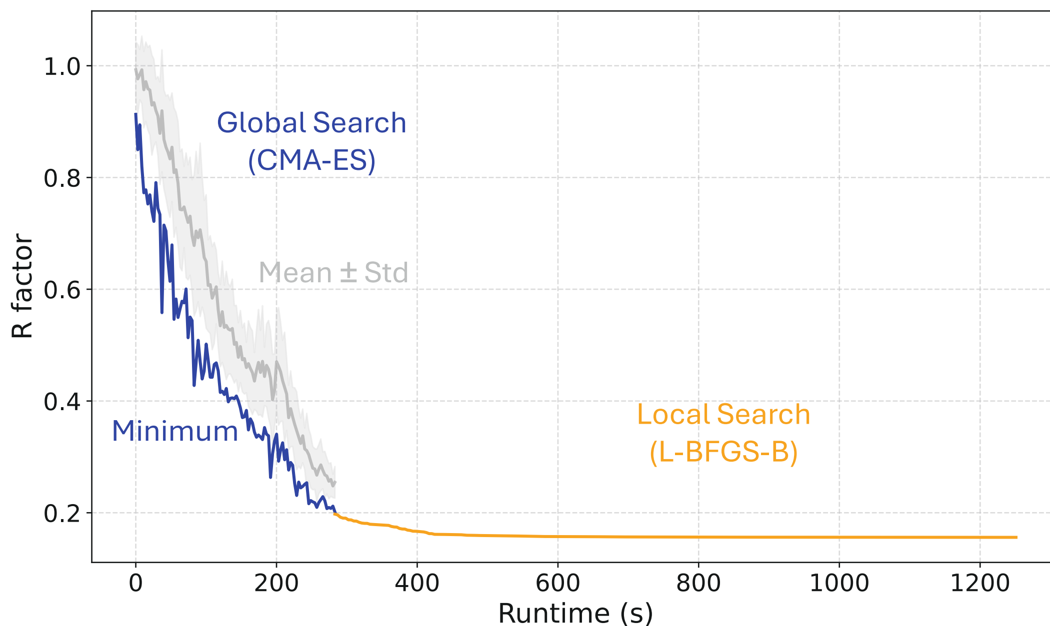

Figure 4.6.: Benchmarking for an entire run. The gray line represents the mean of the R factor of each CMA-ES generation and the gray area the range within $\pm$ one standard deviation. The blue line indicates the minimum R factor of each generation. The orange line depicts the R factor during the L-BFGS-B algorithm.

In Fig. 4.6, the progression of the R factor over time is depicted. In the first part, the CMA-ES algorithm is used for the global search, and in the second part, the L-BFGS-B algorithm is used for the local search. As can be seen, the final part of the algorithm takes the longest. For this reason, the convergence condition for the difference in function value between two iterations was set slightly higher for this run, so it would finish earlier.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Even though the decrease in the final part of the algorithm appears exponential, it is more linear towards end, leading to some improvement even after a long time. In other words, stopping the algorithm even earlier resulted in a small but significant increase in the R factor.

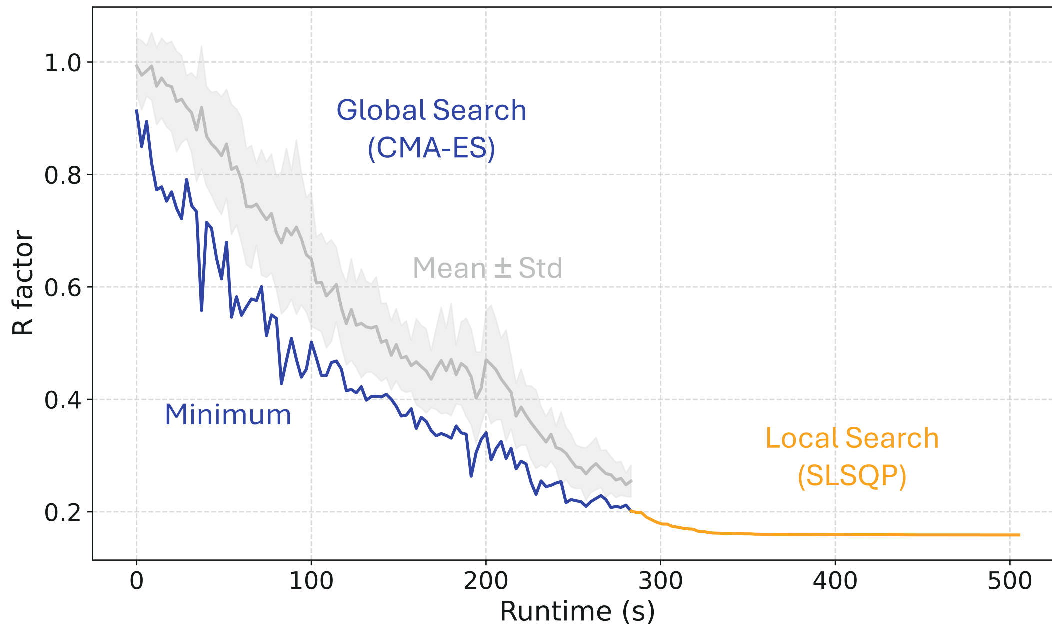

Figure 4.7.: Benchmarking for an entire run. The gray line represents the mean of the R factor of each CMA-ES generation and the gray area the range within $\pm$ one standard deviation. The blue line indicates the minimum R factor of each generation. The orange line depicts the R factor during the SLSQP algorithm.

Fig. 4.7 shows the same global search as Fig. 4.6, but the local search is performed by the SLSQP algorithm. Although the overshooting in this run is not large enough to move to a neighbouring basin, this might be the case for a different starting point. This run is much faster than the one depicted in Fig. 4.6, due to the faster convergence of the SLSQP algorithm.

In Fig. 4.8, the progression of the R factor is shown, with the second part now being performed by the scaled SLSQP algorithm. Overshooting is smaller in this run due to the scaled gradient, which reduces the risk of jumping into a neighbouring basin. For this run, the scaled SLSQP is faster than the standard SLSQP, though this is not always the case.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

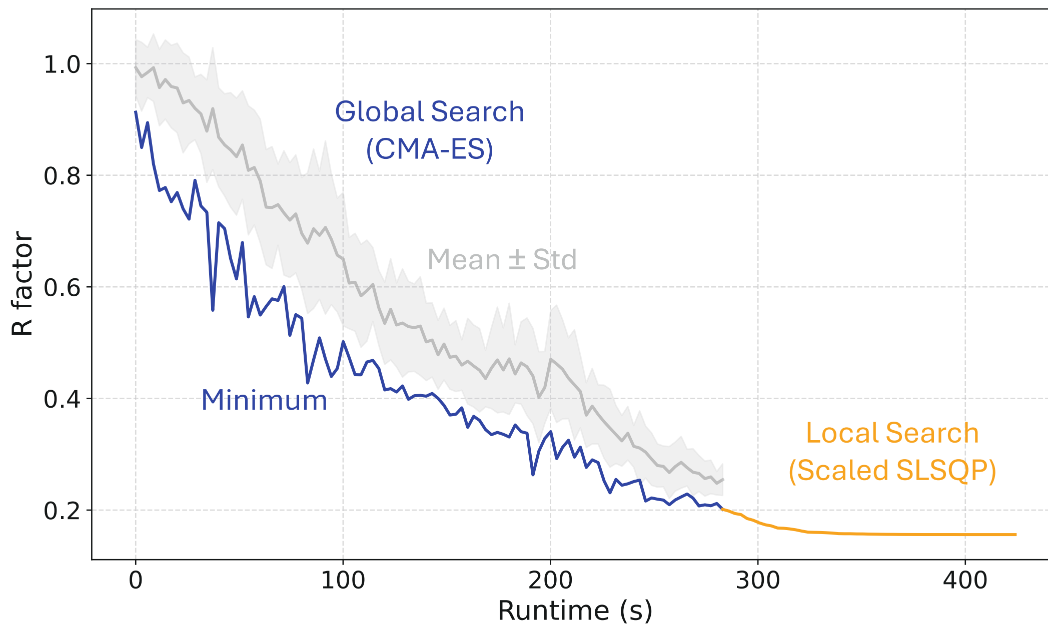

Figure 4.8.: Benchmarking for an entire run with the scaled function. The gray line represents the mean of the R factor of each CMA-ES generation and the gray area the standard deviation. The blue line indicates the minimum R factor of each generation. The orange line depicts the R factor during the scaled SLSQP algorithm.

Until now, only a converged system of hematite has been used. Here, converged means that the reference calculation is performed close to the global minimum and the global minimum of the R factor in the tensor-LEED approximation is within the set bounds. This ensures that the global minimum can be found without repeating the reference calculation. In practice, the starting point usually corresponds to an unrelaxed system. After finding the minimum of this system, the reference calculation and the search are repeated until the global minimum is located.

Finding the exact global minimum using the structure search as implemented in TensErLEED and ViPErLEED is a lengthy procedure, due to the grid-based approach. Even for an almost converged system, multiple iterations of refining the grid and re-calculating delta-amplitudes are necessary to reach the final convergence. For this reason, it is hard to compare the new search with the TensErLEED search on the converged system. Therefore, the two search algorithms are compared on an unrelaxed system.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

In the following, the TensErLEED search and the new search are compared with the same 33 parameters as above, based on the reference calculation of an unrelaxed system. The population size for the CMA-ES was 20, and for the TensErLEED search 48. Since TensErLEED uses several optimization runs with different search ranges, for CMA-ES a bigger search area was used than for TensErLEED.

The TensErLEED search required approximately 7 hours and 30 minutes to locate the minimum R factor of 0.548 and was performed on the Vienna scientific cluster (VSC-4 [40]) on a node with 96 logical cores and 96 GB memory. Note that the TensErLEED code cannot be run on a GPU. On the same setup, our implementation, at the time of writing, would take approximately 12 hours of compile time and 3 hours and 40 minutes of runtime. This is because the new approach using the JAX library is aimed at GPU-like architectures and has not yet been optimized for usage on CPU-only systems. When performed on the VSC-5 with an A100 GPU, the new code takes 6 minutes and 42 seconds of compile time and an additional 6 minutes and 40 seconds of runtime to calculate the minimum R factor of 0.546, representing a significant time improvement compared to the TensErLEED code. Note that all these times are subject to change, as optimizations and refinements to the algorithm are under way. In particular, the memory usage and calculation time of R factor gradients are under active development.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5. Conclusion

Due to the reworked version of the tensor-LEED calculation, the R factor and gradient can be calculated on-demand for arbitrary parameter values such as displacements. This enables the use of various, standardized optimization algorithms. Furthermore, the parameter space is reduced by symmetries and user constraints, and the remaining parameters are normalized.

Because of varying properties of the R factor surface depending on the distance to the global minimum, the search algorithm is split into two parts. The first part focuses on exploring the parameter space to identify the basin containing the global minimum. The second part focuses on exploiting the basin to locate the global minimum.

For the global search, the CMA-ES algorithm is used. Since the used PYTHON implementation (Clinamen2 [25]) does not support bounds, different approaches were tested to include bounds. The partial resampling method led to the most successful runs, especially for systems with the basin close to the bounds. The local search showed the best performance when using the SLSQP algorithm and introducing a scaling factor to address the problem of overshooting in the initial iterations.

It is important to note that the new implementation so far has only been tested on $Fe_2O_3(012) - (1 \times 1)$ (and Cu(111), as a very simple system). For future systems, the hyperparameters of Clinamen2 or the scaling factor might require some adjustments.

We found that the SLSQP algorithm tends to overshoot in the initial iterations, which can make it jump into the wrong basin. To avoid this problem, an initial Hessian approximation is likely the best approach. However, the SCIPY library does not currently support specifying an initial Hessian. To address this issue, a different library could be used. Another possibility is to use only the diagonal elements of the Hessian (assuming the off-diagonal elements are small) and scale the parameters accordingly, so the identity matrix used by the SLSQP implementation is a reasonable approximation for the initial Hessian. The Hessian approximation needed to determine the scale factors could be derived from the CMA-ES algorithm.

Other steps that could be taken in the future include further refining the R factor in cases where it is noisy [41]. This could be achieved by rerunning the entire algorithm within a very small interval around the already identified global minimum. In this case,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the parameters might also require some adjustments, depending on the exact R factor surface within the global minimum basin.

The surface structure search includes several reference calculations and parameter optimizations to reach the converged system and locate the global minimum. In the TensErLEED search and our new implementation, this process requires a lot of user input for each reference calculation and optimization. Due to the faster search, the need for automation of this process increases. Such automation could greatly increase the usability of the new implementation, ViPErLEED and tensor-LEED based optimizations in general.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

List of Figures

1.1. Schematic drawing of the LEED experiment. 1
1.2. Inelastic mean free path of electrons in solids. 2

2.1. All 17 plane symmetry groups are shown. 8

3.1. The thick and dashed ellipsoids show the one-σ contour for six different distributions. 11

4.1. Two-parameter R factor surface for the $Fe_2O_3(012) - (1 \times 1)$ structure. . . 21
4.2. Schematic drawing of the attempted methods to include boundaries. 24
4.3. Logistic function. 25
4.4. Change in the normal distribution when a part is cut off. 26
4.5. Points are drawn from a multivariate normal distribution. 27
4.6. Benchmarking for an entire run (L-BFGS-B). 30
4.7. Benchmarking for an entire run (SLSQP). 31
4.8. Benchmarking for an entire run with the scaled function (SLSQP). 32

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Bibliography

[1] P. Haidegger, On-the-fly calculation of diffraction intensities in the tensor-LEED approximation, project thesis TU Wien (2024).

[2] T. Fauster, L. Hammer, K. Heinz, and M. A. Schneider, *Oberflächenphysik: Grundlagen und Methoden* (Oldenbourg Wissenschaftsverlag GmbH, 2013).

[3] K. Heinz, Low-Energy Electron Diffraction (LEED), in *Surface and Interface Science. Volume 1: Concepts and Methods*, edited by K. Wandelt (Wiley-VCH, Weinheim, 2013) pp. 93–150.

[4] V. Blum and K. Heinz, Fast LEED intensity calculations for surface crystallography using Tensor LEED, Computer Physics Communications **134**, 392–425 (2001).

[5] T. Kißlinger, A. Schewski, A. Raabgrund, H. Loh, L. Hammer, and M. A. Schneider, Surface telluride phases on Pt(111): Reconstructive formation of unusual adsorption sites and well-ordered domain walls, Physical Review B **108**, 205412 (2023).

[6] T. Kißlinger, M. A. Schneider, and L. Hammer, Submonolayer copper telluride phase on Cu(111): ad-chain and trough formation, Physical Review B **104**, 155426 (2021).

[7] J. Qi, P. M. Weber, T. Kißlinger, L. Hammer, M. A. Schneider, and M. Bode, Structure-property relationship of reversible magnetic chirality tuning, Physical Review B **107**, L060409 (2023).

[8] P. Rous and J. Pendry, The theory of tensor LEED, Surface Science **219**, 355–372 (1989).

[9] M. Kottcke and K. Heinz, A new approach to automated structure optimization in LEED intensity analysis, Surface science **376**, 352–366 (1997).

[10] P. Rous, J. Pendry, D. Saldin, K. Heinz, K. Müller, and N. Bickel, Tensor LEED: A Technique for High-Speed Surface-Structure Determination, Physical Review Letters **57**, 2951 (1986).

[11] F. Jona, K. Legg, H. Shih, D. Jepsen, and P. Marcus, Random Occupation of Adsorption Sites in the c(2×2) Structure of CO on Fe{001}, Physical Review Letters **40**, 1466 (1978).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[12] Y. Gauthier, Y. Joly, R. Baudoing, and J. Rundgren, Surface-sandwich segregation on nondilute bimetallic alloys: $Pt_{50}Ni_{50}$ and $Pt_{78}Ni_{22}$ probed by low-energy electron diffraction, Physical Review B **31**, 6216 (1985).

[13] M. Schmid, F. Kraushofer, A. M. Imre, T. Kißlinger, L. Hammer, U. Diebold, and M. Riva, ViPErLEED package II: Spot tracking, extraction and processing of $I(V)$ curves, Phys. Rev. Res. **6** (2024), accepted, arXiv:2406.18413 .

[14] F. Kraushofer, A. M. Imre, G. Franceschi, T. Kißlinger, E. Rheinfrank, M. Schmid, U. Diebold, L. Hammer, and M. Riva, ViPErLEED package I: Calculation of $I(V)$ curves and structural optimization, Phys. Rev. Res. **6** (2024), accepted, arXiv:2406.18821 .

[15] T. Hable, Properties and interpolation of delta-amplitudes in TensErLEED, Bachelor's thesis TU Wien (2023).

[16] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, *et al.*, Array programming with NumPy, Nature **585**, 357–362 (2020).

[17] The JAX authors, JAX: High-Performance Array computing, `https://jax.readthedocs.io/en/latest/index.html` (2024), last accessed March 6, 2024.

[18] J. B. Pendry, Reliability factors for LEED calculations, Journal of Physics C: Solid State Physics **13**, 937–944 (1980).

[19] M. Van Hove, W. Moritz, H. Over, P. Rous, A. Wander, A. Barbieri, N. Materer, U. Starke, and G. Somorjai, Automated Determination of Complex Surface Structures by LEED, Surface Science Reports **19**, 191–229 (1993).

[20] Y. Liu, R. T. Collins, and Y. Tsin, A Computational Model for Periodic Pattern Perception Based on Frieze and Wallpaper Groups, IEEE Transactions on Pattern Analysis and Machine Intelligence **26**, 354–371 (2004).

[21] J. C. Huang, An Approach to Program Testing, ACM Computing Surveys (CSUR) **7**, 113–128 (1975).

[22] W. E. Howden, Functional Program Testing, IEEE Transactions on Software Engineering **6**, 162–169 (1980).

[23] N. Hansen, The cma evolution strategy: A comparing review, in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

edited by J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea (Springer, Berlin, Heidelberg, 2006) pp. 75–102.

[24] N. Hansen, The CMA Evolution Strategy: A Tutorial, (2016), arXiv:1604.00772 .

[25] R. Wanzenböck, F. Buchner, P. Kovács, G. K. Madsen, and J. Carrete, Clinamen2: Functional-style evolutionary optimization in Python for atomistic structure searches, Computer Physics Communications **297**, 109065 (2024).

[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. (Springer, 2006).

[27] P. Wolfe, Convergence Conditions for Ascent Methods, SIAM Review **11**, 226–235 (1969).

[28] P. Wolfe, Convergence Conditions for Ascent Methods. II: Some Corrections, SIAM Review **13**, 185–188 (1971).

[29] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nature Methods **17**, 261–272 (2020).

[30] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, A Limited Memory Algorithm for Bound Constrained Optimization, SIAM Journal on Scientific Computing **16**, 1190–1208 (1995).

[31] D. Kraft, *A Software Package for Sequential Quadratic Programming*, Tech. Rep. DFVLR-FB 88-28 (Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt, Oberpfaffenhofen, 1988).

[32] D. J. Wales and J. P. Doye, Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms, The Journal of Physical Chemistry A **101**, 5111–5116 (1997).

[33] M. Sporn, E. Platzgummer, S. Forsthuber, M. Schmid, W. Hofer, and P. Varga, The accuracy of quantitative LEED in determining chemical composition profiles of substitutionally disordered alloys: a case study, Surface Science **416**, 423–429 (1998).

[34] M. J. D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in *Advances in Optimization and Numerical Analysis*, edited by S. Gomez and J.-P. Hennart (Springer, Dordrecht, 1994) pp. 51–67.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[35] J. A. Nelder and R. Mead, A simplex method for function minimization, The Computer Journal **7**, 308–313 (1965).

[36] M. J. Powell, On search directions for minimization algorithms, Mathematical Programming **4**, 193–201 (1973).

[37] R. S. Dembo and T. Steihaug, Truncated-newton algorithms for large-scale unconstrained optimization, Mathematical Programming **26**, 190–212 (1983).

[38] R. H. Byrd, M. E. Hribar, and J. Nocedal, An Interior Point Algorithm for Large-Scale Nonlinear Programming, SIAM Journal on Optimization **9**, 877–900 (1999).

[39] `https://vsc.ac.at//systems/vsc-5/`.

[40] `https://vsc.ac.at//systems/vsc-4/`.

[41] P. Rous, The tensor LEED approximation and surface crystallography by low-energy electron diffraction, Progress in surface science **39**, 3–63 (1992).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

A. Tables

Name	R factor	time (s)
COBYLA	0.1665	64.9
	0.4218	141.0
	0.3046	164.3
	0.2260	169.4
	0.2263	218.9
	0.2559	188.2
Nelder-Mead	0.1587	122.1
	0.3316	1513.5
	0.3151	832.4
	0.2678	860.3
	0.1823	575.0
	0.1687	1069.2
Powell	0.8072	59.5
	0.6139	231.3
	0.7696	176.2
	0.6994	121.0
	0.5208	57.0
	0.5955	115.5
L-BFGS-B	0.1578	1486.9
	0.1570	2149.5
	0.1567	2958.1
	0.1562	2444.1
	0.1584	2353.3
	0.1581	1915.0

Table A.1.: R factor and runtime for six runs of each local minimization algorithm tested in this work. - Part 1

Name	R factor	time (s)
TNC	0.1577	2231.6
	0.1564	3065.2
	0.1569	2840.1
	0.1557	1415.3
	0.1583	3335.6
	0.1571	1685.3
Trust-Constr	0.1583	2042.6
	0.1569	2433.4
	0.6155	3912.2
	0.1625	3389.8
	0.1564	2579.7
	0.1582	2943.1
SLSQP	0.1594	185.5
	0.1568	339.6
	0.2106	382.4
	0.1571	201.8
	0.1568	236.2
	0.1567	203.4
Scaled SLSQP	0.1581	369.6
	0.1571	132.0
	0.1573	300.8
	0.1561	240.9
	0.1572	160.4
	0.1564	162.8

Table A.2.: R factor and runtime for six runs of each local minimization algorithm tested in this work. - Part 2

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

B. Copyright Clearances

Copyright clearances for all reproduced figures included in this work have been obtained and are provided on the next page.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

JOHN WILEY AND SONS LICENSE
TERMS AND CONDITIONS

Nov 18, 2024

This Agreement between Paul Haidegger ("You") and John Wiley and Sons ("John Wiley and Sons") consists of your license details and the terms and conditions provided by John Wiley and Sons and Copyright Clearance Center.

License Number	5903641280084
License date	Nov 07, 2024
Licensed Content Publisher	John Wiley and Sons
Licensed Content Publication	Wiley Books
Licensed Content Title	Introduction: An Intuitive Approach to Surface and Interface Science
Licensed Content Author	Klaus Wandelt
Licensed Content Date	Nov 21, 2014
Licensed Content Pages	12
Type of use	Dissertation/Thesis
Requestor type	University/Academic
Format	Electronic
Portion	Figure/table
Number of figures/tables	2
Will you be translating?	No
Title of new work	Evolutionary and Gradient-based optimization methods in tensor-LEED
Institution name	TU Vienna
Expected presentation date	Jan 2025
Portions	Figure 3.2.1.2 a) and Figure Figure 3.2.1.4 a)
The Requesting Person / Organization to Appear on the License	Paul Haidegger
Requestor Location	Paul Haidegger Altstraße 19 Holzleiten, 3042 Austria
Publisher Tax ID	EU826007151
Billing Type	Invoice
Billing Address	Paul Haidegger Altstraße 19 Holzleiten, Austria 3042
Total	0.00 EUR

Figure B.1.: Copyright clearance for Fig. 1.1 and 1.2 originally by Wandelt [3]

C. Optimization Code

```
1  """Module optimization."""
2  
3  __authors__ = (
4      'Paul Haidegger (@PaulHai7)',
5      'Alexander M. Imre (@amimre)',
6  )
7  __created__ = '2024-11-20'
8  
9  import time
10 from abc import ABC, abstractmethod
11 
12 import numpy as np
13 from clinamen2.cmaes.params_and_state import (
14     create_sample_from_state,
15     create_update_algorithm_state,
16 )
17 from clinamen2.utils.script_functions import cma_setup
18 from scipy.optimize import minimize
19 from viperleed.calc import LOGGER as logger
20 
21 
22 class Optimizer(ABC):
23     """Base class for all the optimizers.
24 
25     Parameters
26     ----------
27         fun: Objective function
28     """
29 
30     def __init__(self, fun):
31         self.fun = fun
32         self.fun_history = []
33 
34 
35     @abstractmethod
36     def __call__(self):
37         """Start the optimization."""
38 
39 
40 class GradOptimizer(
41     Optimizer
42 ):
43     """Class for optimizers that use a gradient.
44 
45     Parameters
46     ----------
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
47          fun: Objective function to be optimized.
48          grad: Gradient of the objective function.
49          fun_and_grad: Function value and gradient together. This approach
50              is faster, but it only makes sense, if they are always or almost
51              always computed together (e.g., in L-BFGS-B). Avoid using it when
52              having a lot function calls without the gradient (e.g., in SLSQP).
53      """
54
55      def __init__(self, fun=None, grad=None, fun_and_grad=None):
56          self.fun = fun
57          if grad is None and fun_and_grad is None:
58              raise ValueError(
59                  'At least one of grad or fun_and_grad must be set.'
60              )
61          if grad is None and fun_and_grad is None:
62              raise ValueError(
63                  'At least one of grad or fun_and_grad must be set.'
64              )
65          # if only one of the two is set, the other one is set to the default
66          self.grad = grad or (lambda arg: fun_and_grad(arg)[1])
67          self.fun_and_grad = fun_and_grad or (lambda arg: (fun(arg), grad(arg)))
68
69          super().__init__(fun=fun)
70
71          self.current_fun = 0
72          self.current_grad = 0
73
74
75  class NonGradOptimizer(Optimizer):
76      """Class for optimizers that do not use gradients."""
77
78      def __init__(self, fun):
79          self.fun = fun
80          super().__init__(fun=fun)
81
82
83  class LBFGSBOptimizer(GradOptimizer):
84      """Class for setting up the L-BFGS-B algorithm for local minimization.
85
86      The BFGS algorithm uses the BFGS approximation of the Hessian, which is
87      always positive definite. Gradients and Hessians (approximation) are used to
88      determine teh search direction. A line search is performed along this
89      direction, which must satisfy the Wolfe conditions. These conditions provide
90      an upper and lower limit for the step size, and one condition also ensures
91      that the function value monotonically decreases for each iteration.
92
93      Parameters
94      ----------
95          fun_and_grad: Function value and gradient together. This approach
96              is faster, but it only makes sense, if they are always or almost
97              always computed together (e.g., in L-BFGS-B). Avoid using it when
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

```
98              having a lot function calls without the gradient (e.g., in SLSQP).
99          bounds: Since the parameters are normalized, all bounds are by default
100             set to [0,1].
101         ftol: Convergence condition that sets a lower limit on the difference
102             in function value between two iterations. A higher value has been
103             shown to lead to a faster termination without significantly
104             changeing the R factor. It can even be set to 1e-6 for a faster
105             converence, but this slightly worsens the R factor.
106         maxiter: Maximal number of iterations for the algorithm. Usually, the
107             algorithm stops earlier due to convergence.
108     """
109
110     def __init__(self, fun_and_grad, bounds=None, ftol=1e-7, maxiter=1000):
111         self.fun_and_grad = fun_and_grad
112         self.bounds = bounds
113         self.ftol = ftol
114         self.maxiter = maxiter
115         super().__init__(fun_and_grad=fun_and_grad, grad=None, fun=None)
116
117     def __call__(self, start_point):
118         """Start the optimization algorithm.
119
120         This function prints a termination message and returns all the values
121         that are also returned by the SciPy function, plus a list of the
122         function values for each iteration (fun_history) and the
123         runtime (duration).
124
125         Parameters
126         ----------
127             start_point: Starting point of the algorithm.
128         """
129
130         def fun_and_grad_with_storage(arg):
131             """Save function value and grad in variables."""
132             self.current_fun, self.current_grad = self.fun_and_grad(arg)
133             return self.current_fun, self.current_grad
134
135         def callback_function(arg):
136             """This function is called in every iteration to save the function
137             value.
138             """
139             self.fun_history.append(self.current_fun)
140
141         # Setting up the bounds
142         if self.bounds is None:
143             bounds = [(0, 1) for _ in range(len(start_point))]
144         else:
145             bounds = self.bounds
146
147         # Performing the optimization
148         start_time = time.time()
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

```
149         result = minimize(
150             fun_and_grad_with_storage,
151             x0=start_point,
152             method='L-BFGS-B',
153             jac=True,  # assume that the function returns the (val, grad) tuple
154             bounds=bounds,
155             callback=callback_function,
156             options={'maxiter': self.maxiter, 'ftol': self.ftol},
157         )
158         end_time = time.time()
159         duration = end_time - start_time
160         result.fun_history = self.fun_history
161         result.duration = duration
162         logger.info('Optimization Result:\n')
163         logger.info(f'{str(result)}\n\n')
164         return result
165 
166 
167 class SLSQPOptimizer(GradOptimizer):
168     """Class for setting up the SLSQP algorithm for local minimization.
169 
170     The SLSQP algorithm uses a quadratic approximation of the Lagrangian to
171     include equality constraints. Inequality constraints are incorporated by
172     defining active and passive sets, as well as an upper limit of the step
173     size. For the Hessian, the BFGS approximation is used, and the line search
174     cannot satisfy the Wolfe conditions. Due to a poor Hessian approximation in
175     the initial iterations, the algorithm tends to overshoot during these
176     iterations. To address this issue, a damping factor (damp_fact) is used to
177     reduce the gradient at the beginning. A damping factor of 0.1 has shown good
178     results.
179 
180     Parameters
181     ----------
182         fun: Objective function.
183         grad: Gradient of the objective function.
184         bounds: Since the parameters are normalized, all bounds are by default
185             set to [0,1].
186         damp_factor: Damping factor.
187         ftol: Convergence condition that sets a lower limit on the difference
188             in function value between two iterations.
189         maxiter: Maximal number of iterations for the algorithm. Usually, the
190             algorithm stops earlier due to convergence.
191     """
192 
193     def __init__(
194         self, fun, grad, bounds=None, damp_fact=1, ftol=1e-6, maxiter=1000
195     ):
196         self.fun = fun
197         self.grad = grad
198         self.bounds = bounds
199         self.damp_fact = damp_fact
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

```
200         self.ftol = ftol * damp_fact
201         self.maxiter = maxiter
202         super().__init__(fun_and_grad=None, grad=grad, fun=fun)
203
204     def __call__(self, start_point):
205         """Start the optimization.
206
207         This function prints a termination message and returns all the values
208         that are also returned by the SciPy function, plus a list of the
209         function values for each iteration (fun_history) and the
210         runtime (duration).
211
212         Parameters
213         ----------
214             start_point: Starting point of the algorithm.
215         """
216
217         def dampened_grad(x):
218             self.fun_history.append(self.current_fun)
219             return self.damp_fact * self.grad(x)
220
221         def dampened_fun_storage(x):
222             self.current_fun = self.fun(x)
223             return self.current_fun * self.damp_fact
224
225         # Setting up the bounds
226         if self.bounds is None:
227             bounds = [(0, 1) for _ in range(len(start_point))]
228         else:
229             bounds = self.bounds
230
231
232         # Performing the optimization
233         start_time = time.time()
234         result = minimize(
235             fun=dampened_fun_storage,
236             x0=start_point,
237             method='SLSQP',
238             jac=dampened_grad,  # use separate call for gradient
239             bounds=bounds,
240             options={'maxiter': self.maxiter, 'ftol': self.ftol},
241         )
242         end_time = time.time()
243         duration = end_time - start_time
244         result.fun_history = self.fun_history
245         result.duration = duration
246         logger.info('Optimization Result:\n')
247         logger.info(f'{str(result)}\n\n')
248         return result
249
250
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

```
251 class CMAESOptimizer(NonGradOptimizer):
252     """Class for setting up the CMA-ES optimizer for global exploration.
253
254     In each evolution, a number of individuals are drawn from a distribution,
255     and the distribution is updated based on the fanction values of the
256     individuals. For the normalized vector, a step size of 0.5 showed very
257     good results. A population size of 30 for 33 dimensions has proven
258     successful. However, the population size should increase with the number
259     of dimensions (not linearly, but more logarithmically). For such a large
260     step size, 100-200 generations have shown great success.
261
262     Parameters
263     ----------
264         fun: Objective function.
265         pop_size: Number of individuals in each generation.
266         n_generations: Maximum number of generations to be performed.
267         step_size: The standard deviatian in the initial step and a
268             parameter for how much the algorithm should focus on exploring.
269             A step size of 0.5 is quite large, but showed the best results.
270         ftol: Convergence condition on the standard deviation of the minimum
271             function value of the last five generations.
272     """
273
274     def __init__(self, fun, pop_size, n_generations, step_size=0.5, ftol=1e-4):
275         self.fun = fun
276         self.step_size = step_size
277         self.pop_size = pop_size
278         self.n_generations = n_generations
279         self.ftol = ftol
280         super().__init__(fun=fun)
281
282     def __call__(self, start_point):
283         """Start the optimization.
284
285         This function prints a termination message, and the return values
286         are explained below.
287
288         Parameters
289         ----------
290             start_point: Starting point of the algorithm. Usually it start at
291                 0.5 for each dimension, as this is in the middle of the bounds.
292
293         Returns
294         -------
295             min_individual: Parameters of the individual with the smallest
296                 function value.
297             message: A message indicating wether the algorithm finished due to
298                 convergence or reaching the maximum nuber of generations.
299             current_generation: Number of performed generations.
300             duration: Total runtime.
301             fun_history: All function values of all generations stored in a
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
302                 2D array.
303             step_size_history: Step size of each generation stored.
304         """
305         # Set up functions for the algorithm
306         parameters, initial_state = cma_setup(
307             mean=start_point, step_size=self.step_size, pop_size=self.pop_size
308         )
309         sample_individuals = create_sample_from_state(parameters)
310         update_state = create_update_algorithm_state(parameters=parameters)
311         sample_and_evaluate = create_resample_and_evaluate(
312             sample_individuals=sample_individuals,
313             evaluate_single=self.fun,
314         )
315         state = initial_state
316 
317         start_time = time.time()
318         step_size_history = []
319         loss_min = np.full((5,), fill_value=10.0)
320         termination_message = 'Maximum number of generations reached'
321         # Perform the optimization
322         for g in range(self.n_generations):
323             # Perform one generation
324             generation, state, fun_value = sample_and_evaluate(
325                 state=state, n_samples=parameters.pop_size
326             )
327             self.fun_history.append(fun_value)
328             step_size_history.append(state.step_size)
329             # To update the AlgorithmState pass in the sorted generation
330             state = update_state(state, generation[np.argsort(fun_value)])
331             i = g % 5
332             loss_min[i] = fun_value.min()
333             if np.std(loss_min) < self.ftol:
334                 termination_message = (
335                     f'Evolution terminated early at generation {g}.'
336                 )
337                 break
338 
339         end_time = time.time()
340         duration = end_time - start_time
341         if (generation[fun_value.argmin()] < 0.1).any() or (
342             generation[fun_value.argmin()] > 0.9
343         ).any():
344             logger.warning('Parameter(s) close to the bounds!')
345         # Create result object
346         result = CMAESResult(
347             min_individual=generation[fun_value.argmin()],
348             fun=fun_value.min(),
349             message=termination_message,
350             current_generation=g,
351             duration=duration,
352             fun_history=self.fun_history,
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

```
353             step_size_history=step_size_history,
354         )
355         # print the minimum function value in the final generation
356         logger.info(
357             f'Loss {fun_value.min()} for individual '
358             f'{fun_value.argmin()} in generation {g}. '
359             f'With Parameters: {generation[fun_value.argmin()]} \n'
360             f'evaluation time: {duration} seconds'
361         )
362         return result
363 
364 
365 class CMAESResult:
366     """Class for the output of the CMA-ES algorithm.
367 
368     Parameters
369     ----------
370         min_individual: Parameters of the individual with the smallest
371             function value.
372         fun: Smallest function value.
373         message: A message indicating wether the algorithm finished due to
374             convergence or reaching the maximum nuber of generations.
375         current_generation: Number of performed generations.
376         duration: Total runtime.
377         fun_history: All function values of all generations stored in a
378             2D array.
379         step_size_history: Step size of each generation stored.
380     """
381 
382     def __init__(
383         self,
384         min_individual,
385         fun,
386         message,
387         current_generation,
388         duration,
389         fun_history,
390         step_size_history,
391     ):
392         self.min_individual = min_individual
393         self.fun = fun
394         self.message = message
395         self.current_generation = current_generation
396         self.duration = duration
397         self.fun_history = fun_history
398         self.step_size_history = step_size_history
399 
400     def __repr__(self):
401         """Return a string representation of the optimization result."""
402         return (
403             f'OptimizationResult(x = {self.min_individual}\n'
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
404                 f'fun = {self.fun}\n'
405                 f'message = {self.message}\n'
406                 f'current_generation = {self.current_generation}\n'
407                 f'duration = {self.duration:.2f}s)'
408             )
409 
410 
411 class SequentialOptimizer(Optimizer):
412     """Class to run two optimizers sequentially.
413 
414     First, a global optimizer (e.g., CMA-ES) is run, followed by a local
415     optimizer (e.g., SLSQP) for refining the result.
416 
417     Parameters
418     ----------
419         global_optimizer: Instance of a global optimizer.
420         local_optimizer: Instance of a local optimizer.
421     """
422 
423     def __init__(self, global_optimizer, local_optimizer):
424         self.global_optimizer = global_optimizer
425         self.local_optimizer = local_optimizer
426         super().__init__(fun=global_optimizer.fun)
427 
428     def __call__(self, start_point):
429         """Run the optimization pipeline.
430 
431         Parameters
432         ----------
433             start_point: Starting point for the global optimizer.
434 
435         Returns
436         -------
437             A dictionary containing the results of both optimizations.
438         """
439         logger.info('Starting global optimization (CMA-ES)...')
440         # Run the global optimizer
441         global_result = self.global_optimizer(start_point)
442 
443         logger.info('Global optimization finished.\n')
444         logger.info('Starting local optimization')
445         # Use the result of the global optimizer for the local optimizer
446         local_result = self.local_optimizer(global_result.min_individual)
447 
448         logger.info('Local optimization finished.')
449 
450         # Combine results into a single dictionary
451         return {
452             'global_result': global_result,
453             'local_result': local_result,
454         }
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
455
456
457 def create_resample_and_evaluate(
458     sample_individuals,
459     evaluate_single,
460 ):
461     """Create function that samples a population and evaluates its function.
462 
463     Samples that are outside the bounds are partially resampled, which
464     means that only the components which are outside are resampled.
465 
466     Parameters
467     ----------
468         sample_individuals: Function that samples a number of individuals from
469             a state.
470         evaluate_single: Function that returns a tuple containing the loss of
471             an individual and additional information in a dictionary (at least
472             exception if applicable).
473 
474     Returns
475     -------
476         A function to sample (with resampling) and evaluate a population from a
477         state.
478     """
479 
480     def resample_and_evaluate(
481         state,
482         n_samples,
483         n_attempts=int(1e6),
484     ):
485         """Sample a population from a state and evaluate the objective function.
486 
487         When the resampling number reaches n_attempts an OverflowError is
488         raised.
489 
490         Parameters
491         ----------
492             state: State of the previous CMA step.
493             n_samples: Number of successfully evaluated individuals to be
494                 returned.
495             n_attempts: Maximum number of attempts to reach n_samples.
496                 Default is 1e6 to avoid infinite loops.
497 
498         Returns
499         -------
500             tuple containing the following elements:
501             - A population of individuals sampled from the AlgorithmState.
502             - The new AlgorithmState.
503             - An array containing the function value of all passing individuals.
504         """
505         population = []
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
506         loss = []
507         attempts = 0
508 
509         # resample and single evaluate individuals
510         while attempts <= n_attempts and len(population) < n_samples:
511             attempts += 1
512             resampled_population, state = sample_individuals(state, n_samples=1)
513             while (resampled_population[0] < 0.0).any() or (
514                 resampled_population[0] > 1.0
515             ).any():
516                 resampled_population2, state = sample_individuals(
517                     state, n_samples=1
518                 )
519                 condition = np.logical_or(
520                     resampled_population[0] > 1, resampled_population[0] < 0
521                 )
522                 resampled_population = np.where(
523                     condition, resampled_population2, resampled_population
524                 )
525             population.append(resampled_population[0])
526             loss.append(evaluate_single(resampled_population[0]))
527 
528         if len(population) < n_samples:
529             msg = f'Evaluation attempt limit of {n_attempts} reached '
530             raise OverflowError(msg)
531 
532         return (
533             np.asarray(population),
534             state,
535             np.asarray(loss),
536         )
537 
538     return resample_and_evaluate
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.