TECHNISCHE
UNIVERSITÄT
WIEN

BACHELOR THESIS

Sparse-Tight-Binding for TMDCs: Parameter Reduction through Gradient-based Optimization

Bastian Guggenberger

Institute of Theoretical Physics
Vienna University of Technology

Supervisor & Co-supervisor:
Prof. Florian Libisch & Max Sinner, MSc

Wiedner Hauptstrasse 8-10/136
1040 Vienna, Austria

Abstract

Monolayer molybdenum disulfide (MoS_2) is a well known transition metal dichalcogenide (TMDC) with a direct band gap of approximately 1.8 eV, making it a promising candidate for applications in optoelectronics and the semiconductor industry. Tight-binding models provide an efficient framework for capturing the electronic properties of TMDCs, but their predictive power often comes at the cost of a high computational complexity.

In this thesis, an existing tight-binding model for MoS_2 was optimized with a gradient descent algorithm. The optimization was designed to reduce the number of hopping terms while simultaniously preserving the accuracy of the predicted band structure, with special emphasis on the band gap region. The results demonstrate that gradient-based optimization can successfully simplify tight-binding models, reducing computational cost while maintaining essential predictive features.

Contents

1. Introduction

1.1. Motivation

Monolayered molybdenum disulfide (MoS_2) structures are a well known example of transition-metal dichalcogenide (TMDC) monolayers. In 2010, researchers demonstrated that monolayer MoS_2 exhibits a direct band gap of approximately 1.8 eV. The discovery of this bandgap has triggered a wave of new research in the field. TMDCs quickly stood out, since the previous hype about graphene was limited due to the lack of a natural bandgap. This natural bandgap makes TMDCs like MoS_2 especially suitable for applications in optoelectronics such as photodetectors, solar cells or LEDs.

As more deeply discussed in Chapter 2, the core of a tight-binding model are the parameters ϵ_i, the so called on-site energies and t_{ij}, the hopping terms between the different orbitals in the model. While theoretical calculations of these parameters are possible, a different methodology is to fit the parameters numerically, to achieve reasonable band structures with the model. For that, a reference band structure can be obtained either by experimental data, or calculations with other methods such as DFT. Once this calculation is done, the tight-binding model can then be fitted to the reference by numerical variation of the tight-binding parameters.

While many of these tight-binding models are able to reasonably reproduce the reference band structures, these models are often computationally very expensive in their calculations. Accordingly, a successful model is expected to provide reliable predictions of electronic properties, most notably the bandstructure, while at the same time maintaining the lowest possible level of complexity. In this thesis, a model with these characteristics is developed on the basis of an existing tight-binding approach for MoS_2. The primary objective is to significantly reduce the model's complexity while preserving the accuracy of its predictions.

1.2. Structure of the Thesis

This thesis presents the theoretical part of the project, carried out at the Institute of Theoretical Physics (ITP) at TU Wien. The technical component - the Python implementation - is not discussed in this thesis. The complete code and details on the software implementation are available in the corresponding GitHub repository: `https://github.com/BastianGuggenberger/sparse-tight-binding-TMDCs`.
After discussing the theory of tight-binding and gradient-based-optimization methods in Chapter 2, Chapter 3 will focus on an existing tight-binding model for MoS_2 . Finally a gradient descent algorithm will be used in Chapter 4 to simplify this model and reduce the amount of hopping terms. In Chapter 5 a short outlook to potential future work on the subject is provided.

2. Theoretical Background

2.1. Introduction to the tight-binding theory

The tight-binding model is based on the Bloch theorem, which describes the quantum mechanical behaviour of electrons in a crystal. In general the theorem states, that the eigenstates of the crystal electrons are built as follows:

$$\psi_{\vec{k}}(\vec{r}) = u_{\vec{k}}(\vec{r})\, e^{i\vec{k}\cdot\vec{r}} \tag{2.1}$$

Here $u_{\vec{k}}(\vec{r})$ is a periodic function having the periodicity of the crystal. Since the derivation of Blochs theorem does only require periodicity for the crystal potential, a variety of models can be built on Blochs theorem, with different potential strenghts. While the nearly free electron model assumes nearly vanishing periodic potentials, the tight binding model is built on the assumption of strong periodic potentials, where the electrons are bounded tightly to the ions ("tight binding"), as visualized in figure 2.1: [1].

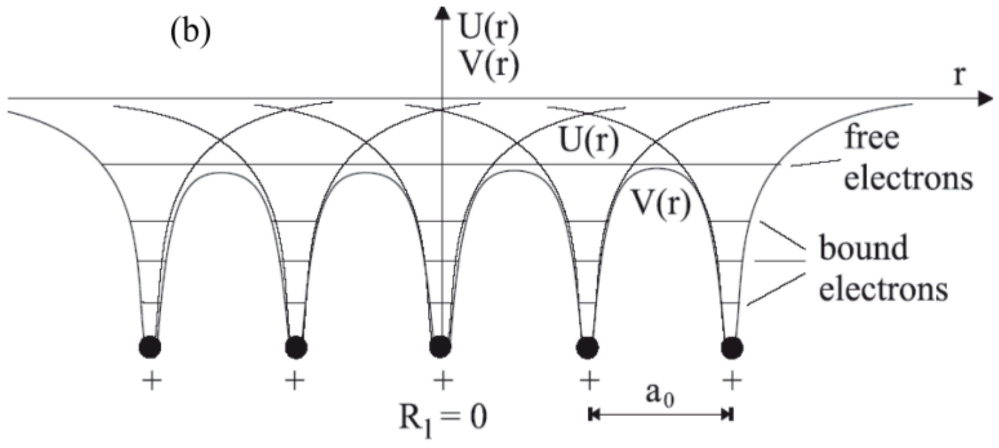


Figure 2.1.: Periodic potential in the tight-binding theory [1]

The bloch waves of the crystal electrons in the tight binding theory are based on the eigenstates ϕ_j of the free atom:

$$\Psi_{j,\mathbf{k}}(\mathbf{r}) = \sum_{l} e^{i\mathbf{k}\mathbf{R_l}} \phi_j(\mathbf{r} - \mathbf{R_l}) \tag{2.2}$$

where j describes the orbital of the eigenstate.[1] An important assumption is, that the atomic wavefunctions ϕ do not share a large overlap in the crystal:

$$\int \phi_j^*(\mathbf{r})\phi_j(\mathbf{r} - \mathbf{R_l})d\tau \approx 0 \tag{2.3}$$

In the following, we will only use the outermost orbital ϕ., the dispersion relation $E(\mathbf{k})$ can be found using perturbation theory. For that we assume the difference $(V-U)$ between the crystal hamiltonian H and the hamiltonian $H_{at,l}$ of the free atom to be small [1] :

$$\begin{aligned} H_{at,l} &= \left[-\frac{\hbar^2}{2m}\Delta + U(\mathbf{r}-\mathbf{R_l})\right] \\ H &= \left[-\frac{\hbar^2}{2m}\Delta + V(\mathbf{r})\right] \\ H &= H_{at,l} + (V-U) \end{aligned} \tag{2.4}$$

Now the dispersion relation can be calculated with perturbation theory:

$$E = E_0 + \sum_{\mathbf{R}=\mathbf{0},nn} e^{i\mathbf{k}\cdot\mathbf{R}} \int \phi^*(\mathbf{r}-\mathbf{R})(V-U)\phi(\mathbf{r})d\tau \tag{2.5}$$

The sum over the $\mathbf{R}$ vectors is usally only calculated for the nearest (first order) neighbours, since all other transfer integrals vanish due to the lack of overlap between their atomic wavefunctions. Equation 2.5 can then be rewritten as [1]:

$$E = E_0 - A - \sum_{nn} e^{i\mathbf{k}\cdot\mathbf{R}} T(\mathbf{R}) \tag{2.6}$$

with the definitions

$$-A = \int \phi^*(\mathbf{r})(V-U)\phi(\mathbf{r})d\tau \tag{2.7}$$

$$-T(\mathbf{R}) = \int \phi^*(\mathbf{r}-\mathbf{R})(V-U)\phi(\mathbf{r}) \tag{2.8}$$

Since the value of T is proportional to the probability of an electron "hopping" from an original lattice point to a different point, T is often referred to as the "hopping integral", while A is called the "on-site energy" of the orbital. In this derivation we only considered the outermost orbital of the atom, which is not accurate enough for many applications, such as the MoS_2 model considered in this thesis, since hoppings can also exist between inner orbitals.

In order to consider more than just one orbital in the system, more complex tight binding models can be developed by using a Hamiltonian with hoppings between different orbitals. In matrix form the Hamiltonian can then be written as [2]:

$$H_{ij}(\mathbf{k}) = \epsilon_i \delta_{ij} - \sum_{\mathbf{R}\neq 0 \vee i\neq j} t_{ij}(\mathbf{R}) e^{i\mathbf{k}\cdot\mathbf{R}} \tag{2.9}$$

usually a hermitian matrix with the on-site energies on the diagonal and the hopping terms as off-diagonal elements. The possible energies E can then be obtained by solving for the eigenvalues of the hamiltonian. The on site energy A from equation 2.7 is now a different constant for every orbital:

$$\epsilon_i = \int \phi_i^*(\mathbf{r})\hat{h}_0\phi_i(\mathbf{r})d\mathbf{r} \tag{2.10}$$

The hopping integral T is now a matrix containing the hopping integrals for all orbital combinations. Further it is also dependent on the distance $\mathbf{R}$ between the atoms:

$$t_{ij}(\mathbf{R}) = -\int \phi_i^*(\mathbf{r})\hat{h}_0\phi_j(\mathbf{r}-\mathbf{R})d\mathbf{r} \tag{2.11}$$

where in the above equations $\hat{h}_0$ denotes the Hamiltonian of the free atom. Note that the sum of hopping terms t_{ij} in equation 2.9 only has terms with $\mathbf{R} \neq 0$ or $i \neq j$. This means that the hopping terms exist only between different orbitals ($i \neq j$), except from hoppings between equal orbitals that are located on different atoms ($\mathbf{R} \neq 0$). The integrals between equal orbitals on equal atoms are the so called on-site energies ϵ_i in equation 2.10 [2].

Once the Hamiltonian is given, the dispersion relationship and density of states (the possible states per energy interval) can be calculated by diagonalization. Since exact diagonalization is very resource expensive especially for large models, the Tight Binding simulation package "TBPLaS" used in this thesis uses a more efficient method called the "tight-binding propagation method", which achieves "linear scaling with system size in both memory and CPU costs" [2].

The core of a tight binding model are the tight binding parameters ϵ_i and t_{ij}. While it is possible to calculate them directly with the formulas in equations 2.10 and 2.11, there are also methods such as the Slater-Koster formula or numerical fitting of the model to band structures obtained by experimental or theoretical methods [2]. The tight binding model used in this thesis is based on a model built by *R Roldán et al*[3], where the tight binding parameters are found by numerical fitting of the tight-binding band structure to a band structure obtained by DFT calculations.

Slater-Koster Tight Binding

The Slater-Koster tight binding method was originally suggested in 1954 by J.C. Slater and G.F. Koster as a solution to the problem of the enormous computational complexity of the tight binding method. Depending on the complexity of the system, hopping integrals between a large number of different wave functions will have to be evaluated with the formula in equation 2.11. Since this was hardly possible at the time, Slater and Koster derived a further developed method, using a predefined set of parameters (later called "Slater-Koster parameters") to describe the hoppings between given atomic orbitals ($s, p, d, f, ...$). For each different geometry of two neighbored orbitals there exists a different SK parameter. Additionally, on site energies also exist in this model. [4]

While the Slater-Koster parameters can be theoretically derived, the success of the method lies in the fact, that the parameters can and should be adapted to fit bandstructures obtained by first-principle calculations like DFT. The important idea in Slater and Kosters paper was the suggestion that the integrals can be seen as constants that can be adjusted to the results of other calculations, without loosing their ability to predict properties also in other conditions. [4] The Slater-Koster idea is fundemental to this thesis, aiming at a satisfying representation of a bandstructure by a different and less complex set of parameters.

2.2. The gradient descent method

Gradient Descent is an iterative optimizing algorithm for minimizing a certain function, often referred to as the "error function". A really simple gradient descent algorithm can be built like this [5]:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \lambda \nabla f(\mathbf{x}_t) \tag{2.12}$$

where $f(\mathbf{x})$ is the error function to be minimized and λ some parameter, determining the step size of each iteration. This classic version of the algorithm is intuitive and easy to visualize, as it simply just takes one step a time in the direction of the steepest descent (see figure 2.2).

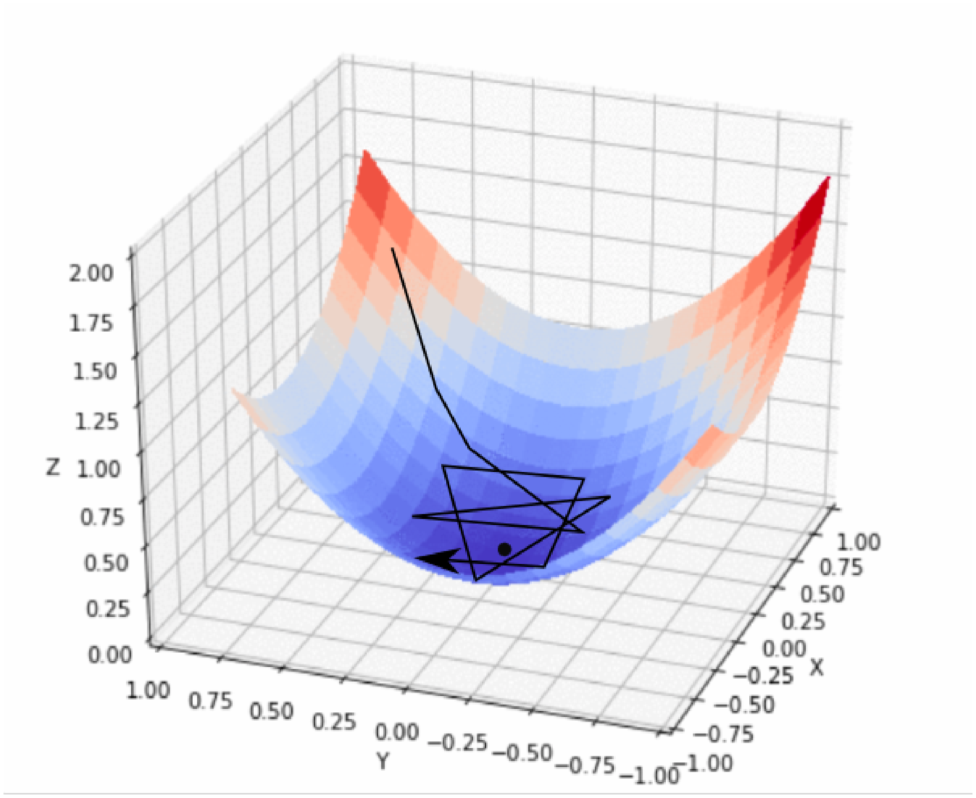


Figure 2.2.: Visualization of the gradient descent idea[6]

The idea is, that the algorithm then converges to a local minimum of the error function f. While this can be effective for many functions, one problem is, that even if the algorithm finds a local minimum, there is still the possibility that this is not the global minimum. Additionally the classic gradient descent method can take many iterations, when f has for example a constant small gradient in a certain region. A perhaps more useful alternative in these cases is "Nesterov Accelerated Gradient Descent" [7].

Nesterov Accelerated Gradient Descent (NAG)

The Nesterov acceleration can be best visualized as a ball rolling down a hill. While the force always pulls the ball to the direction of the steepest descent, the velocity of the ball in general points in a different direction, since the ball has a certain momentum. In the following, we shall implement this idea in an algorithm.
We will start with the general definition of an iterative algorithm, updating the current "best guess" $\mathbf{x}_t$ by a correction $\mathbf{v}_t$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_t$$

The correction term v_t (the velocity of the ball in the analogon) should now be composed of a part of the old correction term v_{t-1}, and additionaly the "force term", a term pointing in the direction of the steepest descent. A possible method would be the following:

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} - \eta \nabla f(\mathbf{x_t})$$

While it is an intuitive possibility to evaluate the gradient of f at the current x_t as in the above formula, usually a different method is used, where the gradient is evaluated at a point $\bar{\mathbf{x}}_t$ nearer to the future point $\mathbf{x}_{t+1}$:

$$\begin{aligned}\bar{\mathbf{x}}_t &= \mathbf{x}_t + \gamma \mathbf{v}_{t-1} \\ \mathbf{v}_t &= \gamma \mathbf{v}_{t-1} - \eta \nabla f(\bar{\mathbf{x}}_t)\end{aligned}$$

which leads to the overall NAG algorithm:

$$\begin{aligned}\bar{\mathbf{x}}_t &= \mathbf{x}_t + \gamma \mathbf{v}_{t-1} \\ \mathbf{v}_t &= \gamma \mathbf{v}_{t-1} - \eta \nabla f(\bar{\mathbf{x}}_t) \\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{v}_t\end{aligned} \tag{2.13}$$

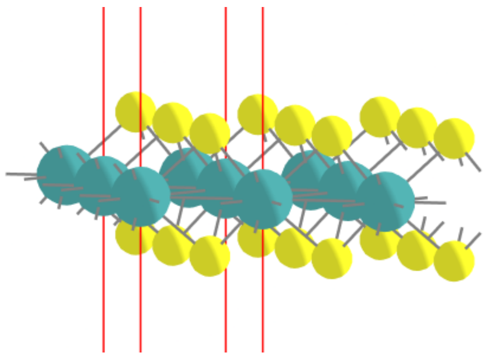
where η takes the role as the "force coefficient", determining how much a steep descent "pulls" the ball in its direction. γ in contrast represents the inertia of the ball, or in our context, how much of the old velocity is maintained for the next iteration.

3. Implementation of a MoS_2 Tight Binding Model in TBPLaS

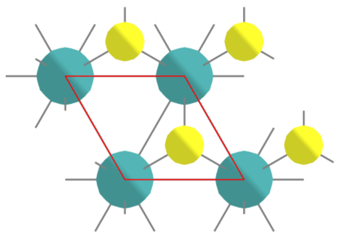
In this thesis an existing tight-binding model for MoS_2 is used, published by "R Roldán et al. [3]" . The TBPLaS library fortunately has a built in function for the implementation of the Roldán et al. MoS_2 model, which will be used in this thesis: $tbplas.make\_mos2\_soc()$ [8]. An introduction on the usage of this function and the further implementation of the tight binding model in python can be found on github, while the following chapter will give an overview over the physical aspects of the Roldán model.

3.1. Material Properties of MoS_2 monolayers

A monolayer MoS_2 structure is built of a molybdenum layer, sandwiched between two layers of sulphur atoms (see figure 3.1).



Monolayer MoS_2



MoS_2 primitive cell

Figure 3.1.: Different MoS_2 plots from the Computational 2D Materials Database. Note, that the Roldán et al. model uses different choices for the primitive cell and lattice vectors. [9] [10]

Each primitive cell consists of one molybdenum atom and 2 sulphur atoms. A useful set of lattice vectors is the following [1] [3]:

Lattice vector	x [nm]	y [nm]	z [nm]
$\mathbf{a}_1$	0.31600	0.00000	0.00000
$\mathbf{a}_2$	-0.15800	0.27366	0.00000
$\mathbf{a}_3$	0.00000	0.00000	0.31720

Table 3.1.: Lattice vectors of monolayer MoS_2

[1]There are different choices of lattice vectors in literature, however we will stick with the Roldán et al. convention, as we are later going to use a python implementation based on their paper. Unfortunately, the Roldán et al. paper does not directly show the choice of lattice vectors, however, they can be found over their TBPLaS implementation [8] and the $print()$ method of the $tbplas.PrimitiveCell$ object [11]. The same holds true for the locations of the molybdenum and sulphur atoms.

Given these lattice vectors, the locations of the molybdenum and sulphur atoms in each primitive cell can be expressed by $\mathbf{R} = c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + c_3\mathbf{a}_3$ with the following coefficientcs c_i (the so called "fractional coordinates") [3]:

Atom	c_1	c_2	c_3
Sulphur (lower layer)	0.00000	0.00000	0.00000
Molybdenum	0.66667	0.33333	0.50000
Sulphur (upper layer)	0.00000	0.00000	1.00000

Table 3.2.: atom locations in monolayer MoS_2

3.2. Tight Binding Model for MoS_2

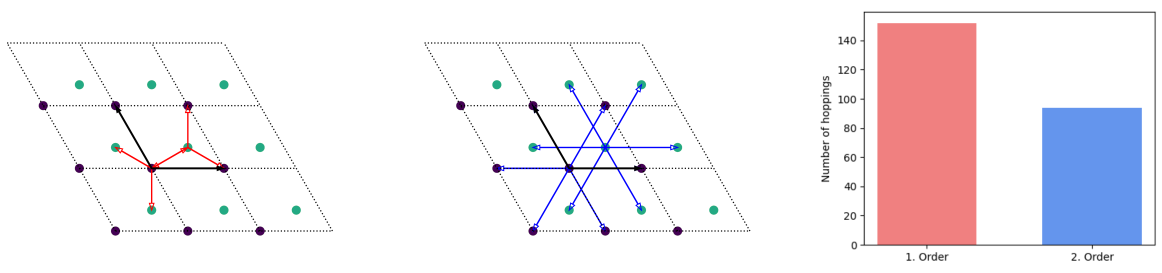
As discussed before, the blueprint to be optimized in this thesis is the Roldán et al. tight binding model. This section will serve as a short summary of the model in their paper.

Orbitals

The Roldán et al. model considers 11 orbitals to be relevant for electron hoppings: five d-orbitals of the molybdenum and three p-orbitals for each of the two sulphur atoms. Since every orbital can be filled with a spin up and a spin down electron, there are in total 22 relevant orbitals for each primitive cell. Table A.1 in appendix A shows the label, location and on-site energy of each of these orbitals [3].

Hoppings

In the Roldán model hoppings between neighbours of first and second order are considered. Their energies t_{ij} are obtained by numerical fitting to a DFT band structure, as discussed before. Since the model considers spin-orbital-coupling, it uses in general complex hopping energies. However, spin-orbital-coupling will be neglected in this thesis, motivating us to only consider the real part of the hopping energies, leaving us with a total of 388 hoppings. Details on how this is done in python can be found on github.
As the goal of this thesis is to build a less complex tight binding model, the 388 hoppings are further reduced by only considering hopping energies $|t_{ij}| > 0.1$ eV. Remaining are 246 hopping terms, of which 152 are hoppings between first- and 94 between second order neighbours, as illustrated in figures 3.2 and 3.3.



1st order hoppings   2nd order hoppings   Numbers of hopping terms

Figure 3.2.: MoS_2 hopping terms 1st and 2nd order in the final tight binding model.

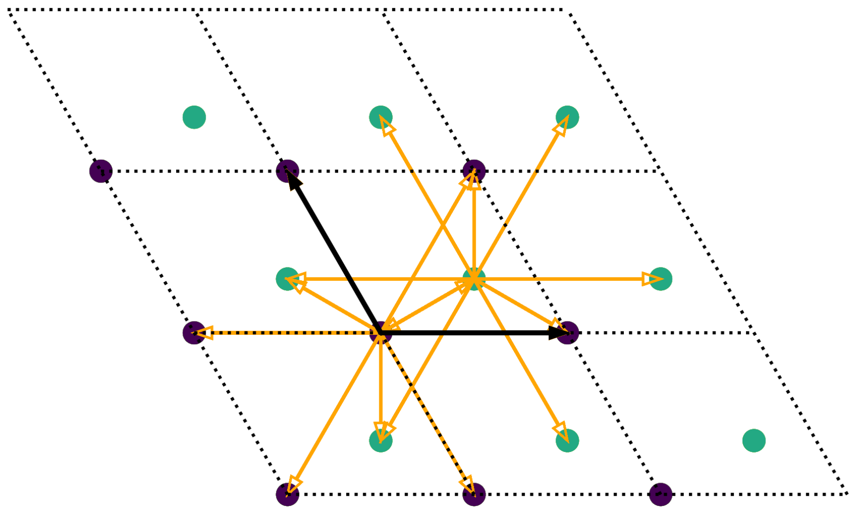


Figure 3.3.: All hoppings in the Roldán model.

4. Gradient Descent for reducing Hopping Terms of MoS_2

4.1. General Idea, Error function

The main idea of this thesis is to use a Gradient Descent algorithm to optimize the set of hoppings of our MoS_2 tight-binding model. The optimization should fulfill two goals: reducing the overall number of hoppings in the model, while keeping the band structure close to the original.
From these two goals, a gradient descent algorithm can be used to minimize an error function, which punishes both a high number of hoppings and a high deviation from the original band structure. For better optimization results, a Nesterov Gradient Descent (NGD) algorithm will be used.

Gradient Descent parameter

In order to work with a gradient descent algorithm, a parameter is needed, to represent the set of hopping terms. We want our final hoppings set to be based on the original hopping set, but allow the hoppings to have modified values of the hopping integral, or even to be removed completely from the model (which would be equivalent to having a hopping integral of 0). This leads us to choose a weight vector $\mathbf{x}$ as GD-parameter, which carries the new hopping energies of all hopping terms as ratio in comparison to the original hopping terms:

$$x_i := \frac{E_{i\text{new}}}{E_{i\text{original}}} \tag{4.1}$$

Error function

The properties of the error function can be directly derived by the two main goals described above. In order to reduce the total number of hopping terms, the errorfunction should contain a term that promotes low $|x_i|$ and punishes high $|x_i|$. The square root function turns out to be really useful for that, which is not surprising, given its shape: For N hopping terms, the square root term in the error function would then look like this:

$$EF(\mathbf{x}) = const. + \lambda_1 \sum_{i=1}^{N} \sqrt{|x_i|}$$

As it turns out, the squareroot term of the error function does a great job at bringing many x_i's close to zero, however, it does not prevent the x_i's from diverging, which quickly leads to overflow errors. In order to prevent that, a higher order polynomial is added to the error function:

$$EF(\mathbf{x}) = const. + \lambda_1 \sum_{i=1}^{N} \sqrt{|x_i|} + \lambda_2 \sum_{i=1}^{N} x_i^6$$

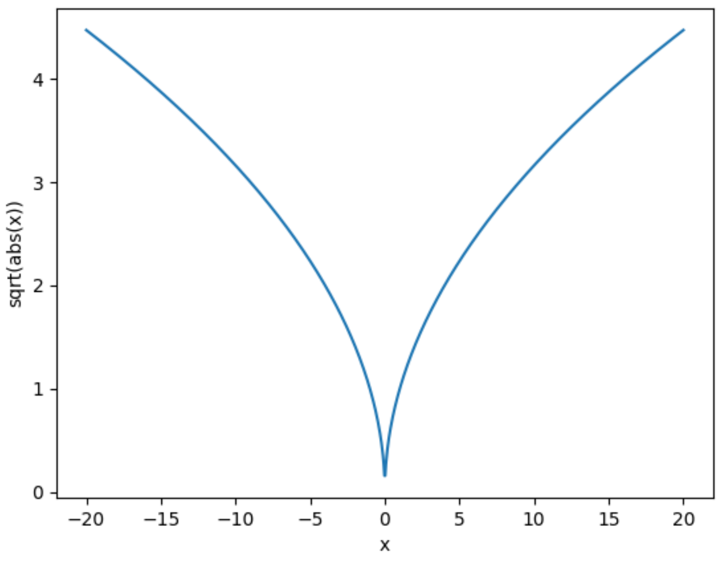


Figure 4.1.: f(x) $= \sqrt{|x|}$ is a great function for reducing hopping terms, because of its global minimum at $x = 0$, which is also a singularity.

where λ_2 is chosen as really small, e.g. 0.05.

The second goal of the gradient descent algorithm should be to let our model result in a band structure as similar as possible to the original band structure. The difference between the original band structure, and the one predicted by the new model with the weight vector $\mathbf{x}$, can be measured by a certain metric m. The metric used in this thesis takes a sum over the quadratic differences in energy of n different points in k-space. The n points are spaced linearly on a path in k-space between the points Γ, M, K and Γ :

$$m(\mathbf{x}) = \sum_{\text{bands}} \sum_{j=1}^{n} [E_{\mathbf{x}}(\mathbf{k}_j) - E_{\text{original}}(\mathbf{k}_j)]^2 \tag{4.2}$$

Due to the high computational costs of calculating dispersion relations, n is hold lower ($n = 45$) during the gradient descent algorithm. Once the algorithm is finished, n can be chosen higher ($n = 150$) when the metric is calculated for the final result and analysis.

The final error function then looks like this:

$$EF(\mathbf{x}) = \underbrace{\lambda_0 m(\mathbf{x})}_{\text{deviation of the bandstructure}} + \underbrace{\lambda_1 \sum_{i=1}^{N} \sqrt{|x_i|}}_{\text{amount of hoppings}} + \underbrace{\lambda_2 \sum_{i=1}^{N} x_i^6}_{\text{divergent terms}} \tag{4.3}$$

Because the error function should punish high m, high $\sqrt{|x_i|}$ and high x_i^6, all three constants λ_i have to be positive. Finally, the optimization problem to be solved is the following:

$$x^* = \arg\min_x f(x) \tag{4.4}$$

4.2. Implementation in Python

For the band structure calculations and all changes to the hoppings of the MoS_2 Cell in the gradient descent algorithm, the class "mcell" and multiple TBPLaS based functions are used in the implementation, which can in detail be accesed on the Github repository. The fundamental algorithm can be seen in Listing 4.1, the full code is also available on Github.

```python
 1  #NESTEROV GRADIENT DESCENT:
 2
 3  #Gradient-Descent-Parameter x:
 4  x = [1 for hop in idealhops]
 5
 6  #velocity vector v in nesterov gradient descent:
 7  v = [0 for hop in idealhops]
 8
 9  #building a MoS2 Cell with all hoppings > E_min:
10  currentcell = mcell("currentcell",E_min)
11
12  for wdh in range(iterations):
13
14      #1.Step NGD: y(t) = x(t) + gamma*v(t-1):
15      y = x.copy()
16      y = [y[i] + gamma * v[i] for i in range(len(y))]
17
18      #2.Step NGD: v(t) = gamma*v(t-1) - kappa*d(EF)/dx |y
19      currenthopvec = mxtohopvec(y,idealhops.copy())
20      currentcell.mchangehops_tohopvec(currenthopvec)
21      EF_current = EF(y,currentcell)
22      for i in range(len(y)):
23          partial = part_deriv_EF(y,i,EF_current,currentcell)
24          v[i] = gamma * v[i] - kappa * partial
25
26          #3.Step NGD: x(t+1) = x(t) + v(t)
27          x[i] = x[i] + v[i]
```

Listing 4.1: Nesterov gradient descent algorithm in python

The step 2 in the NGD algorithm is more complex here, because of the evaluation of the partial derivatives of the error function.
The function $part\_deriv\_EF(y, i, EF\_current, currentcell)$ calculates the partial derivative $\frac{\partial EF}{\partial x_i}$ numerically:

$$\left.\frac{\partial EF}{\partial x_i}\right|_y \approx \frac{EF(\mathbf{y}) - EF(..., x_i + \Delta x)}{\Delta x} \tag{4.5}$$

Since all N calculations of the partial derivative involve calculating $EF(\mathbf{y})$, the algorithm calculates this quantity only once in line 21 and gives the result to $part\_deriv\_EF$ as an argument. For this calculation, the MoS2 cell needs to be updated first to the hopping corresponding to the vector $\mathbf{y}$, which happens in lines 19 and 20.
$EF(..., x_i + \Delta x)$ is then calculated by changing the hopping integral of the hopping corresponding to x_i to $E_{new} = (x_i + \Delta x)E_{original}$ and then calculating the error function of the resulting cell and $\mathbf{x}$ - vector.

4.3. Hyperparameters

The gradient descent algorithm leaves many parameters open, both in the physics and in the machine learning context. After experimenting with many different sets of hyperparameters, the following set of parameters turned out to be most useful:

Parameter	Value	Description
λ_0	4.0	metric weight factor in EF
λ_1	variable	$\sqrt{\lvert x_i \rvert}$ weight factor in EF
λ_2	0.05	x_i^6 weight factor in EF
E_{min}	0.1 eV	minimal hopping integral taken into account
$x_i(t=0)$	1	starting point in NGD
κ	$\frac{1}{1700}$	"speed" factor κ in NGD
γ	0.3	"inertia" factor γ in NGD
Δx	0.05	Δx factor in partial derivative computation
Iterations	400, 1200	number of iterations in NGD

Table 4.1.: hyperparameters and preferred values in the NGD algorithm

Since λ_1 indirectly determines the number of hoppings left after the algorithm is done, various values for λ_1 will be tried out in order to get a curvature in the metric vs N_{hoppings} space.
For the starting point, the possibilities $x_i(t=0)=1$ and $x_i(t=0)=0$ were used, representing either an algorithm trying to reduce as many hoppings as possible or an algorithm trying to build up the necessary hoppings to best represent the band structure. After 400 iterations, the algorithm has nearly reached its convergence point, however there is still a small improvement until 1200 iterations, which can be seen in chapter 4.4.

4.4. Results

In order to determine the success of the gradient descent algorithm, we will compare it with a much simpler yet uneffective method for reducing hoppings: simply eliminating hopping terms in order of ascending hopping integrals. This method seems reasonable from a physical standpoint, at least for really small hopping integrals. The gradient descent algorithm can be run for multiple λ_1 values in order to get a corresponding curvature in the N vs m space. The results for 400 or 1200 iterations are plotted in figure 4.2. For a remaining hoppings / total hoppings ratio that is smaller than 0.5, the NGD algorithm by far outperforms the ascending energies algorithm, since for the same metric, there are much less hoppings remaining.

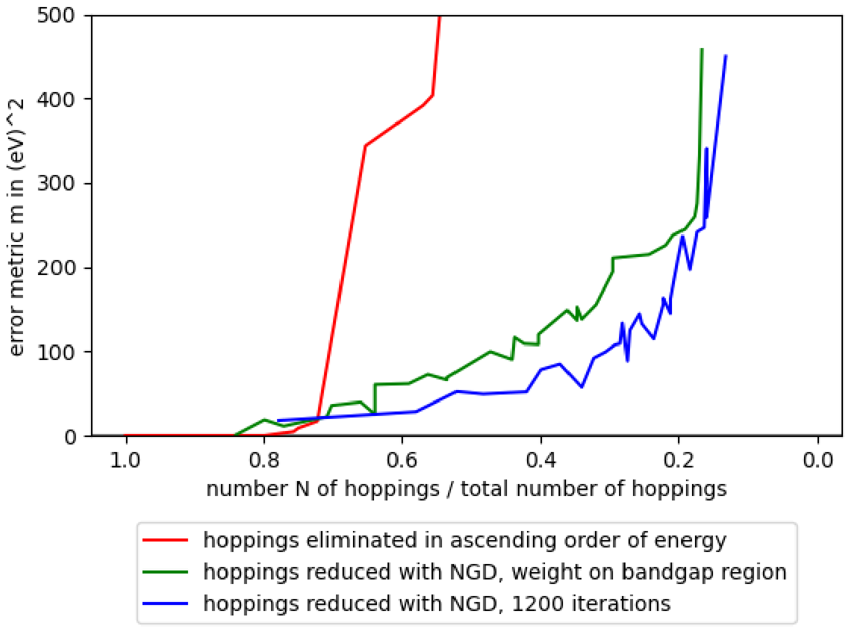


Figure 4.2.: N vs m: comparison of NGD and ascending order of energy algorithms

The calculated band structures from both algorithms are compared to the original band structures in figure 4.3. For this specific run, resulting with 94 remaining hoppings, the largest deviation near the bandgap of a band from the original model is 0.3606 eV. This result will be further improved in Chapter 4.6, when we optimize the bands near the bandgap.

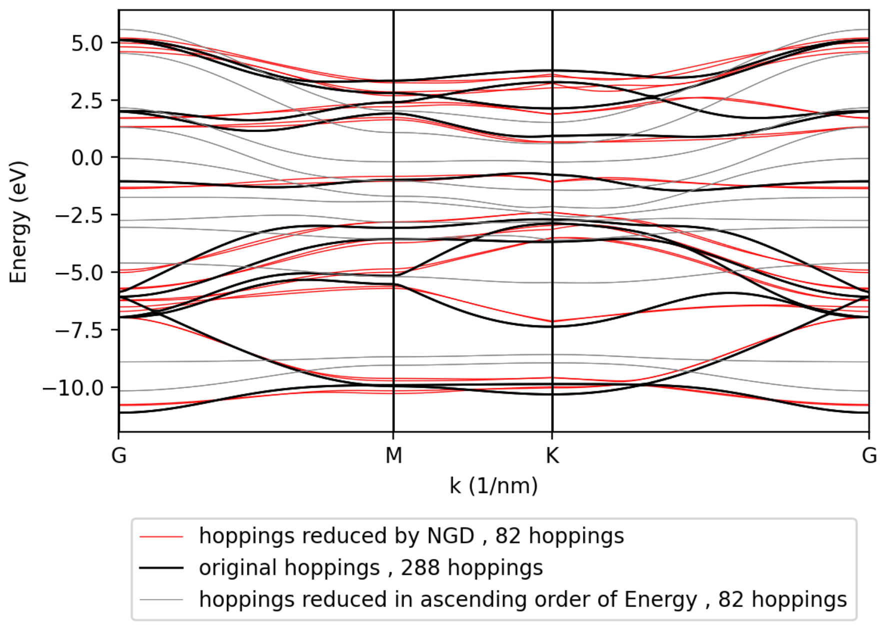


Figure 4.3.: Comparison of original bandstructure and bandstructures with hoppings reduced by NGD and ascending order of energy algorithm

Reduction in calculation time

The simplification of the tight-binding model not only makes the model more intuitive, but also has the benefit of a significant reduction of the calculation time of the band structure. While the simplification has many benefits, it obviously comes with the cost of a less accurate prediction of the band structure. This error is compared to the time savings for the band structure calculation in figure 4.4:

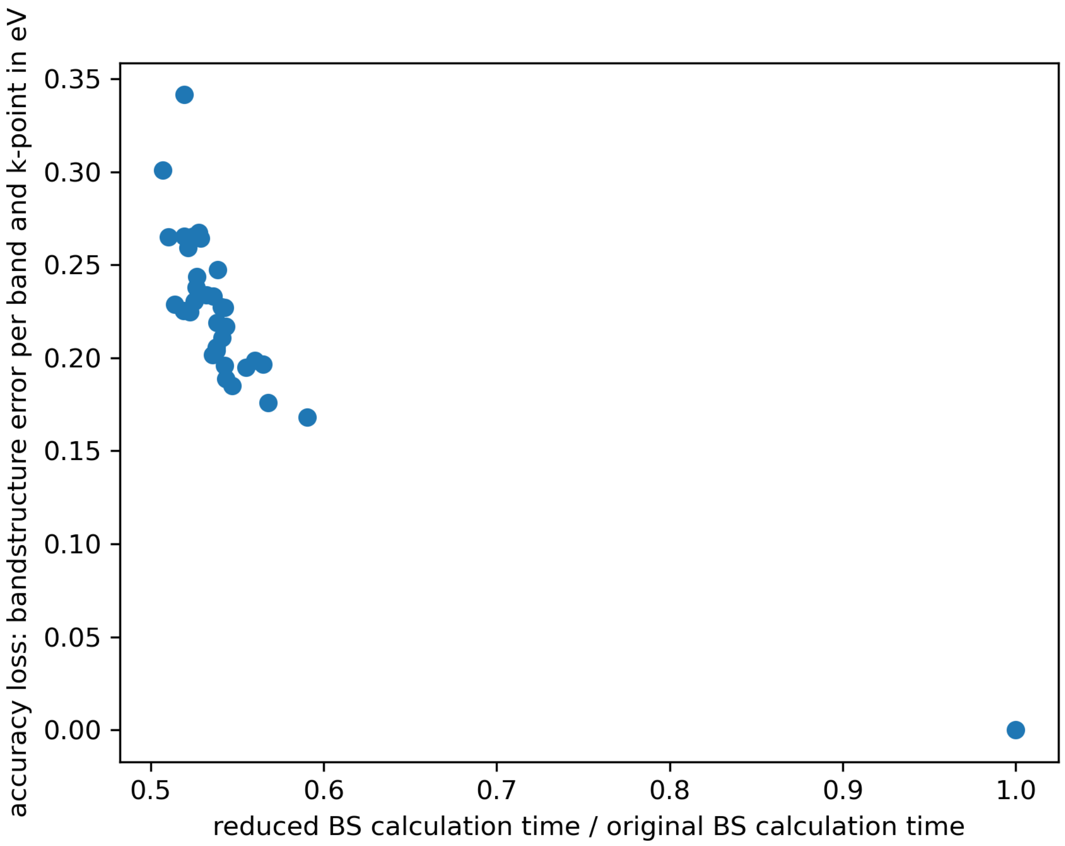


Figure 4.4.: Reduction of the band structure calculation time for the NGD based simplified MoS_2 model (1200 iterations)

4.5. Analyzing degrees of order

The degrees of neighbourhood of the remaining hopping terms can be interesting for further development of the algorithm. The original hopping terms from the Roldán et al. model are of first and second degree of order, as discussed in section 3.2. We now want to analyze the order of the hoppings taken away for different periods of the algorithm. For that, the method *mget_neighbours_orders* is built, which is based on the *tbplas.PrimitiveCell.dr_nm* attribute in TBPLaS. The method enables us to keep track of how many hoppings exist for each degree of order over the evolvement of the algorithm. The results for the previous run resulting in 94 remaining hoppings are shown in figure 4.5:

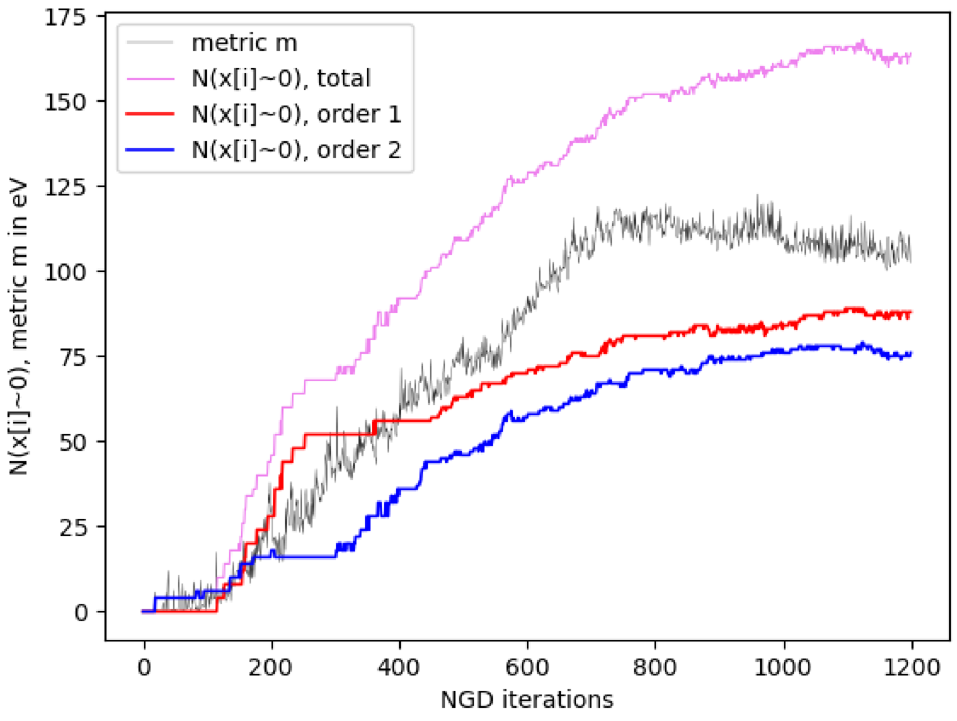


Figure 4.5.: Analysis of degrees of order of hopping terms over the evolvement of the NGD algorithm

While the algorithm does reduce different orders at different times, the principle of having to optimize for $N \approx 300$ parameters at once still is a huge downside of the algorithm. This opens the question, whether the algorithm could be more effective when it is dealing with only the hoppings of a given order for some given time. The most effective form of that algorithm turns out to be an iteration of optimizations, alternately only for order 2 and than only for order 1. The NGD algorithm from Listing 4.1 has to be slightly modified for that, by checking if the hopping index i corresponds to a hopping of the right order, before changing x_i. Also the squareroot term in the error function now only considers x_i that belong to the current degree of order. Figure 4.6 compares the modified algorithm with the classic NGD method:

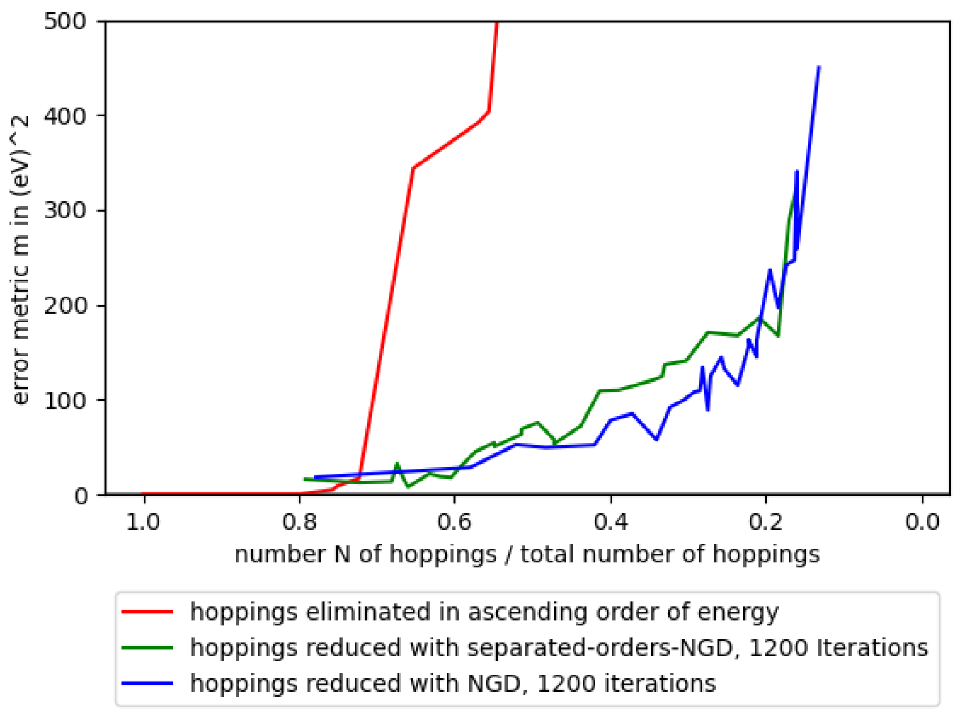


Figure 4.6.: N vs m: comparison of classic NGD algorithm vs modified separated-order NGD algorithm

As it turns out, optimizing the two orders seperately does not improve the performance of the NGD algorithm, the performance of the two algorithms is approximately the same for most reduction Numbers N.

4.6. Analysis of the band gap

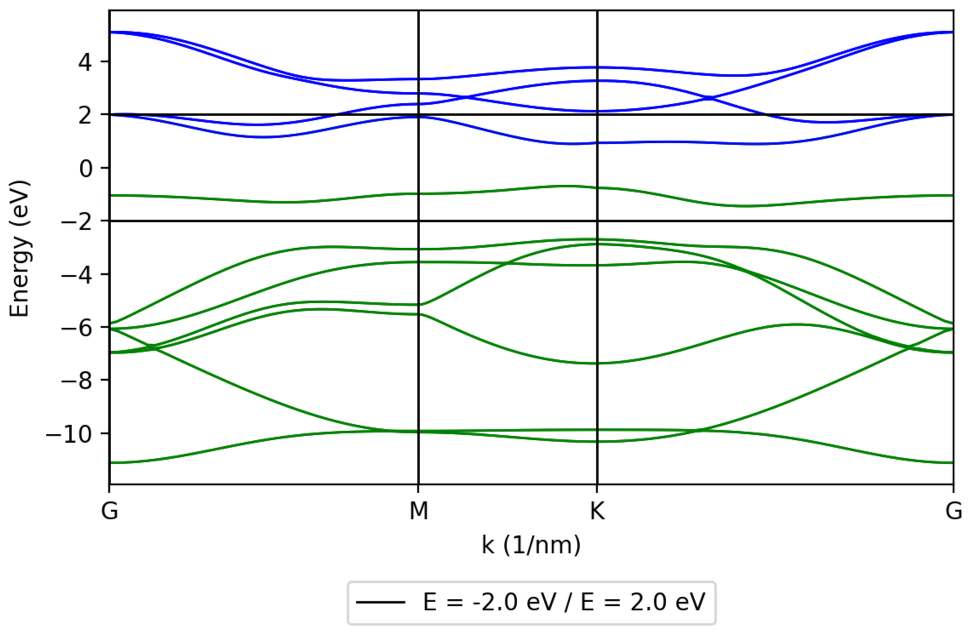


Figure 4.7.: "MoS_2 band structure of the original tight binding model"

As laid out before, an important feature of TMDCs such as MoS_2 is the band gap. Monolayer MoS_2 has a bandgap of ≈ 1.8 eV between the valence and conduction bands in figure 4.7 . Since the bands near the band gap are the most relevant ones for applications, the above algorithm can be further modified to prioritize these bands in the numerical fitting process. This prioritization is done by modifying the metric m in equation 4.2 using a weight for the bandgap error terms:

$$m = \sum_{\text{bands}} \sum_{j=1}^{n} w(\mathbf{k}_j)[E_{\text{new}}(\mathbf{k}_j) - E_{\text{original}}(\mathbf{k}_j)]^2 \tag{4.6}$$

, with

$$w(\mathbf{k}_j) = \begin{cases} w_0 & E_{new}(\mathbf{k})_j \vee E_{original}(\mathbf{k})_j \in (-2 \text{ eV}, 2 \text{ eV}) \\ 1 & E_{new}(\mathbf{k})_j \wedge E_{original}(\mathbf{k})_j \notin (-2 \text{ eV}, 2 \text{ eV}) \end{cases}$$

Using this modified metric, a simplified tight binding model with - as before - 94 hoppings now has more useful results, achieving a maximum absolute error of 0.4992 eV for the bands in the bandgap area from -2.0 eV to 2.0 eV. Figure 4.8 shows the obtained band structure:

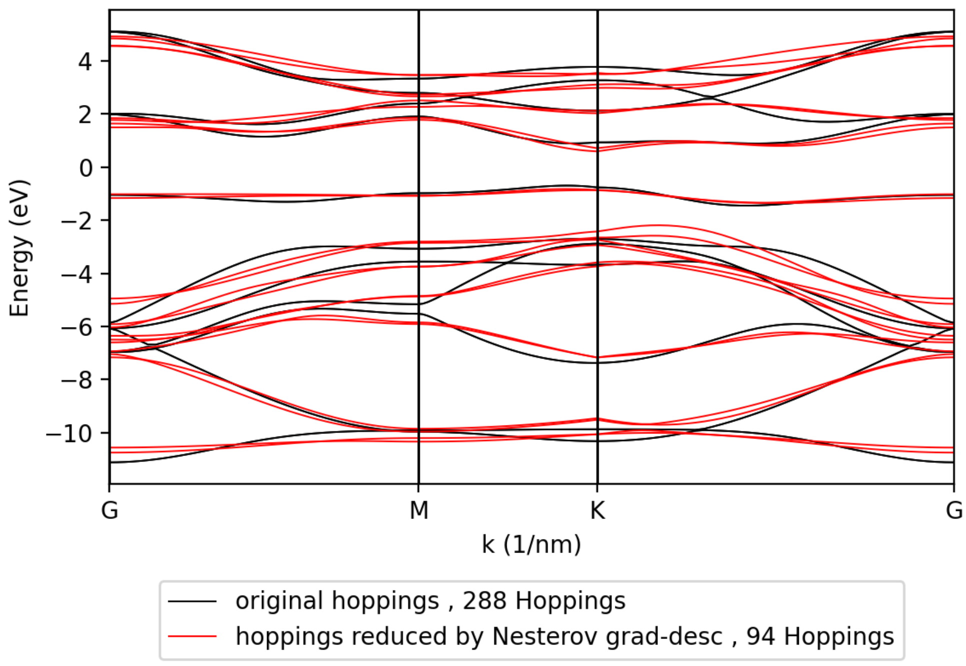


Figure 4.8.: "Comparison of original bandstructure and bandstrucutre of the NGD reduced MoS_2-cell with higher weight of the metric in the band gap area."

5. Outlook

While this thesis has concentrated on a Gradient-based method for the parameter optimization of TMDC tight-binding models, there are many other suitable methods, which could in principle be used. One of these methods, often used for dimensionality reduction in machine learning, is the Singular Value Decomposition (SVD). With SVD it is possible to clear the most irrelevant data, in order to achieve a simpler but still precise representation of that data.
SVD could therefore also be used in our case to sparsify tight-binding hamiltonians. An attempt of that was also carried out in this project for MoS_2, however it was quite unsuccesful yet, since the model considered in this thesis uses orbitals only on 3 different fractal positions, leaving the SVD with not enough instances. However, for tight-binding models or materials with a larger number of instances, SVD-based parameter reduction appears to be a promising approach that warrants further investigation.

Future work will also be needed to focus on extending the tight-binding model to additional orbitals, considering more original hopping terms and the spin orbital coupling, which was not handled in this thesis yet. Moreover, a model could be extended to more complex systems, such as defects or edge-states, which are especially important for future applications.

Bibliography

[1] Silke Bühler-Paschen, Marta Gibert, and Herwig Michor. *Festkörperphysik I, Skriptum zur LVA 138.017*. Deutsch. Wien, 2025.

[2] Yunhai Li et al. “TBPLaS: A tight-binding package for large-scale simulation”. In: *Computer Physics Communications* 285 (Apr. 2023), p. 108632. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2022.108632. URL: https://www.sciencedirect.com/science/article/pii/S0010465522003514 (visited on 06/13/2025).

[3] R Roldán et al. “Momentum dependence of spin–orbit interaction effects in single-layer and multi-layer transition metal dichalcogenides”. en. In: *2D Materials* 1.3 (Nov. 2014). Publisher: IOP Publishing, p. 034003. ISSN: 2053-1583. DOI: 10.1088/2053-1583/1/3/034003. URL: https://dx.doi.org/10.1088/2053-1583/1/3/034003 (visited on 06/18/2025).

[4] J. C. Slater and G. F. Koster. “Simplified LCAO Method for the Periodic Potential Problem”. In: *Physical Review* 94.6 (June 1954). Publisher: American Physical Society, pp. 1498–1524. DOI: 10.1103/PhysRev.94.1498. URL: https://link.aps.org/doi/10.1103/PhysRev.94.1498 (visited on 09/19/2025).

[5] Zhouchen Lin, Huan Li, and Cong Fang. *Accelerated Optimization for Machine Learning: First-Order Algorithms*. en. Singapore: Springer, 2020. ISBN: 978-981-15-2909-2 978-981-15-2910-8. DOI: 10.1007/978-981-15-2910-8. URL: http://link.springer.com/10.1007/978-981-15-2910-8 (visited on 06/23/2025).

[6] *Intro to optimization in deep learning: Gradient Descent | DigitalOcean*. en. URL: https://www.digitalocean.com/community/tutorials/intro-to-optimization-in-deep-learning-gradient-descent (visited on 09/18/2025).

[7] Markus Wallerberger. “Lecture Notes, "Machine Learning in Physics, LVA 138.128"”. 2025.

[8] *tbplas.make_mos2_soc — TBPLaS 2.0.0 documentation*. URL: https://www.tbplas.net/_api/tbplas.make_mos2_soc.html (visited on 06/24/2025).

[9] Sten Haastrup et al. “The Computational 2D Materials Database: high-throughput modeling and discovery of atomically thin crystals”. en. In: *2D Materials* 5.4 (Sept. 2018). Publisher: IOP Publishing, p. 042002. ISSN: 2053-1583. DOI: 10.1088/2053-1583/aacfc1. URL: https://dx.doi.org/10.1088/2053-1583/aacfc1 (visited on 06/24/2025).

[10] Morten Niklas Gjerding et al. “Recent progress of the Computational 2D Materials Database (C2DB)”. en. In: *2D Materials* 8.4 (July 2021). Publisher: IOP Publishing, p. 044002. ISSN: 2053-1583. DOI: 10.1088/2053-1583/ac1059. URL: https://dx.doi.org/10.1088/2053-1583/ac1059 (visited on 06/24/2025).

[11] *tbplas.PrimitiveCell — TBPLaS 2.0.0 documentation*. URL: https://www.tbplas.net/_api/tbplas.PrimitiveCell.html#tbplas.PrimitiveCell.print (visited on 06/24/2025).

Acknowledgements

During my time as a bachelor's student at TU Wien I have had the pleasure of meeting many inspiring people, students, tutors, professors and staff members. The amount of people dedicated to supporting each other on their path to a better understanding of nature and technology has inspired and motivated me in times when the complexity and volume of the matter we were studying seemed overwhelming.

Among all of these people, I would especially like to point out my supervisors Prof. Florian Libisch and Max Sinner for the time and energy they have had for me and my curiosity. In the process of working on this thesis, they have not only supported me with their technical knowledge, but also provided valuable insights into their profession.

In the past three years, I would never have had the great time I had without my friends Désirée Bieberle, Gabriel Fuchs, Philipp Hälbig, Jakob Schwanda and Leo Thalhammer. I am very grateful for our countless conversations and discussions while improving our understanding of physics, and more importantly, for all the fun we had, making this process so enjoyable and entertaining.

A. MoS$_2$ in TBPLaS

A.1. TBPLaS Implementation of the Roldán et al. Model

... The function *tbplas.make_mos2_soc*() adds the following 22 orbitals to the *tbplas.PrimitiveCell* (as discussed in section 3.2):

Label	Position in fractional coordinates [(c_1, c_2, c_3)]	On-site energy [eV]
'Mo:dz2:up'	(0.66667, 0.3333299999999999), 0.5)	-1.512
'Mo:dzx:up'	(0.66667, 0.3333299999999999), 0.5)	0.419
'Mo:dyz:up'	(0.66667, 0.3333299999999999), 0.5)	0.419
'Mo:dx2-y2:up'	(0.66667, 0.3333299999999999), 0.5)	-3.025
'Mo:dxy:up'	(0.66667, 0.3333299999999999), 0.5)	-3.025
'S1:px:up'	(0.0, 0.0, 0.0)	-1.276
'S1:py:up'	(0.0, 0.0, 0.0)	-1.276
'S1:pz:up'	(0.0, 0.0, 0.0)	-8.236
'S2:px:up'	(0.0, 0.0, 1.0)	-1.276
'S2:py:up'	(0.0, 0.0, 1.0)	-1.276
'S2:pz:up'	(0.0, 0.0, 1.0)	-8.236
'Mo:dz2:down'	(0.66667, 0.3333299999999999), 0.5)	-1.512
'Mo:dzx:down'	(0.66667, 0.3333299999999999), 0.5)	0.419
'Mo:dyz:down'	(0.66667, 0.3333299999999999), 0.5)	0.419
'Mo:dx2-y2:down'	(0.66667, 0.3333299999999999), 0.5)	-3.025
'Mo:dxy:down'	(0.66667, 0.3333299999999999), 0.5)	-3.025
'S1:px:down'	(0.0, 0.0, 0.0)	-1.276
'S1:py:down'	(0.0, 0.0, 0.0)	-1.276
'S1:pz:down'	(0.0, 0.0, 0.0)	-8.236
'S2:px:down'	(0.0, 0.0, 1.0)	-1.276
'S2:py:down'	(0.0, 0.0, 1.0)	-1.276
'S2:pz:down'	(0.0, 0.0, 1.0)	-8.236

Table A.1.: orbitals added to the Primitive Cell in the Roldán et al. model. "S1" denotes the lower sulphur atom, "S2" the upper sulphur atom and "Mo" the molybdenum atom.