DISSERTATION

Optimizing the Density Functional Theory Code WIEN2k

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Doktor der Technischen Wissenschaften

unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Blaha

Institut für Materialchemie (E165), TU Wien, Österreich

eingereicht an der Technischen Universität Wien

Fakultät für Technische Chemie

von

Dipl.-Ing. Thomas Ruh, BSc.

Matrikelnummer 00825393

Wien, 5. Juni 2023

Thomas Ruh

TECHNISCHE UNIVERSITÄT WIEN

DISSERTATION

Optimizing the Density Functional Theory Code WIEN2k

carried out in partial fulfilment of the requirements for the degree of

Doktor der Technischen Wissenschaften

under the supervision of

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Blaha

Institut für Materialchemie (E165), TU Wien, Österreich

submitted to the TU Wien

Faculty of Technical Chemistry

by

Dipl.-Ing. Thomas Ruh, BSc.

Registration Number 00825393

Vienna, 5th June, 2023

Thomas Ruh

But we all got a
chicken-
duck-
woman-
thing
waiting for us.

BUSHES OF LOVE – BAD LIP READING,
YouTube, 02.01.2016

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Thomas Ruh, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 5. Juni 2023

Thomas Ruh

Kurzfassung

Erkenntnisse über moderne Materialien auf einem atomistischen Level sowie ein Verständnis von Struktur-Eigenschaftsbeziehungen sind von größter Bedeutung, wenn es um das Design und/oder die Optimierung von Materialien geht. Eine Möglichkeit, solche Erkenntnisse zu gewinnen, stellen Computersimulationen dar, für deren Durchführung heute eine Vielzahl verschiedener Modelle und Näherungsmethoden im Einsatz sind.

Das Softwarepaket WIEN2k verwendet „Full-Potential Linearized Augmented Plane Wave“ Kohn-Sham-Dichtefunktionaltheorie (DFT), um Materialien zu beschreiben und deren Eigenschaften zu berechnen und/oder vorherzusagen. Die Grundlage dafür sind fundamentale quantenmechanische Simulationen ihrer elektronischen Strukturen.

Heutzutage ist die Leistungsfähigkeit moderner Computerhardware so weit fortgeschritten, dass es möglich ist, derartige Simulationen auf einem „normalen“ Laptop zu rechnen, sofern es sich um „einfache“ Materialien handelt. Allerdings wachsen die komputativen Kosten solcher Simulationen mit der dritten Potenz der Zahl der Atome, die notwendig ist, um das Material zu modellieren. Dies führt dazu, dass die Verwendung von hocheffizienten und optimierten Computercodes unerlässlich ist, um deshalb immer noch herausfordernde komplexere Materialien mit Simulationen behandeln zu können.

Eine der größten Hürden bei der Behandlung von komplexen Materialien besteht in der Tatsache, dass im Zuge von DFT-Simulationen wiederholt generalisierte Eigenwertprobleme zu lösen sind, die immer noch zu den für Computer am schwierigsten zu lösenden algebraischen Problemen gehören.

In dieser Arbeit wird die Leistungsfähigkeit der in WIEN2k verwendeten Algorithmen zur Lösung von Eigenwertproblemen getestet und optimiert – sowohl im Zusammenhang mit High Performing Computing (deutsch etwa „Hochleistungsrechnen“) als auch für Fälle, in denen alle statt nur „einige“ der niedrigsten Eigenwerte benötigt werden (wie es bei Routinerechnungen mit WIEN2k der Fall ist). Darüber hinaus wird mithilfe eines neuen Programms (3DDENS) eine neue signifikant schnellere Methode zur Berechnung von Elektronendichten auf 3D-Rastern aus WIEN2k-Daten implementiert. Schließlich wird WIEN2k auf einige Fallbeispiele unterschiedlicher Komplexität angewendet, um unterschiedliche Anforderungen verschiedener Problemstellungen an komputative Parameter sowie Optimierungsansätze zu zeigen.

Abstract

Insights into modern advanced materials on an atomistic level as well as an understanding of structure-property relations are crucial for designing and/or optimizing materials with specifically tailored properties. Simulations are an effective way of gaining such understanding. A vast variety of different models and approximations are used nowadays.

The software package WIEN2k utilizes full-potential linearized augmented plane wave Kohn-Sham **D**ensity **F**unctional **T**heory (DFT) to describe materials and obtain and/or predict their properties based on fundamental quantum mechanical simulations of their electronic structure.

Nowadays, hardware performance has grown to a degree that lets laptops run DFT simulations of "simple" materials. However, the computational cost of simulations grows cubically with the number of atoms needed to model it, which means that more complex materials are still challenging and need efficient and well-optimized codes.

A major cause of these challenges with DFT simulations is the fact that during these simulations generalized eigenvalue problems have to be solved repeatedly, which still belong to the most difficult algebraic problems for computers to solve efficiently.

In this thesis, the performance of WIEN2k is assessed and optimized with a particular focus on the utilized eigensolvers – both in context of high performance computing and in cases when all eigenvalues are needed (instead of the lowest "few" eigenvalues that are used in routine calculations). Furthermore, a new and much faster method to obtain electronic densities on a 3D grid from WIEN2k data was implemented in WIEN2k by adding a new program (`3DDENS`). Finally, WIEN2k is applied to several problems with different levels of complexity to illustrate how different simulations have different computational requirements and benefit from different optimizations.

Acknowledgements

A doctoral thesis is an undertaking that cannot be completed without the support of a multitude of different people:

First and foremost, I want to thank my supervisor Peter Blaha for the possibility to join his group and work on my thesis, for sharing his knowledge and experience, and for his help in all matters DFT and WIEN2k. Second, I would like to thank Fabien Tran (who shared his office with me) for his help in making sense of WIEN2k.

It would be remiss of me to not acknowledge all the friends I made along the way of my academic endeavours. They made the good times so much better and substantially lightened the frustration during the more challenging and stressful stretches: Christian, Danny, and Max, with whom I began my studies back in 2008; Jesús, Andi, Andre, Christoph, Florian (two actually), Glix, Hedda, Leila, Lorenz, Karin, KoPé, Peter, Raffael, Ralf, Sebastian, Ulrike, Tobi, Vera, Verena, and all the others, who have – sadly – currently slipped my mind, I had the great pleasure of meeting during my PhD work at TU Wien; all the people from Ghent University (Stefaan, Kurt, my office mates Michiel and Sam, and so many more) who made my stay in Belgium that much more fun...

Finally, I owe my deepest gratitude to my husband, Klaus Ruh, my parents Karin and Christian Ruh as well as my sister Hannah for their constant support (morally, emotionally, and financially) during all of my studies (beginning with my bachelor studies all the way through to the completion of this doctoral thesis).

The computational results presented have been achieved using the Vienna Scientific Cluster (VSC). Therefore, I would also like to express my gratitude to the good people from the VSC team (Claudia, Dieter, Irene, Jan, Markus, and Sig), for their patience and their support with all HPC-related troubles.

Last but not least, I gratefully acknowledge financial support by the FWF (project P27738-N28 and the doctoral school Solids4Fun), and the VSC Research Center funded by the Austrian Federal Ministry of Science, Research, and Economy (bmwfw) for funding the VSC School.

Contents

CHAPTER 1

Introduction

In the beginning the Universe was created.
This has made a lot of people very angry and been widely regarded as a bad move.

Douglas Adams, The Restaurant at the End of the Universe (1980)

Solid materials in any shape or form have always been – and still are – the very foundation of day-to-day life: basic building materials (wood, concrete, steel, glass...) used in construction of infrastructure and housing, components in mechanical engineering (often metals or ceramics) or "high-tech" materials based on semi-conductors (transistors, solar cells...) are only a few examples.

Nowadays, applications grow ever more complex and need specific – and often specifically tailored – materials with certain desired properties. Therefore, it is not surprising that materials science plays a major role, since the understanding of materials – and relations between certain characteristics (e.g. structure or composition) and materials properties – becomes more and more essential.
One possible way to gain crucial insights are simulations – both complementary to experiments and stand-alone: In their capacity as complementary methods, simulations can support key findings, assist with interpretation of results, or shed light on the roots of interesting properties and their trends. As stand-alone tool, they can be used to predict properties of new materials or properties of known materials that are difficult to investigate experimentally. Another possible use of simulations – particularly due to the advent of machine learning – are screenings, in which a large pool of possible candidate materials are checked for the suitability for a given application.

Today, a multitude of theoretical methods is available that use different approaches and approximations; examples include empirical models with problem-/material-specific parameters [1, 2], Monte Carlo techniques using repeated random sampling and subsequent

stochastic analysis [3, 4], or ab-initio (or first-principle) methods like **D**ensity **F**unctional **T**heory (DFT) [5, 6], which takes the electronic density in a molecule or a unit cell into account.

Looking at the number of publications gives a clear proof of the growing importance of simulations in general and DFT in particular: Searching for the keywords "density functional theory" in the Web of Science Core Collection[i] and Scopus[ii] (both are widely used databases of scientific literature), yields more than 202 000 and 210 000 results, respectively, for the years 1990–2020 (see Figure 1.1). The milestone of 10 000 yearly publications listed in those databases has been reached in 2012 and 2011, respectively, and this number is still growing rapidly – resulting in an average of almost 40 daily DFT publications during the last decade.

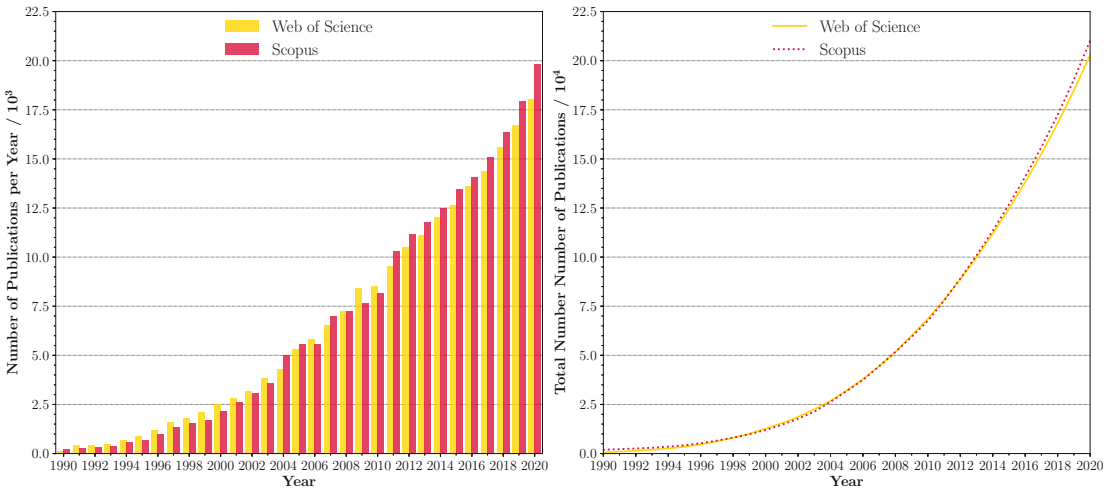

Figure 1.1: The yearly (left) and total (right) number of publications between 1990 and 2020 found when searching the Web of Science Core Collection and Scopus for the keywords "density functional theory".

DFT simulations of "simple" materials – with structures that can be modelled using small unit cells up to 50 atoms and structures that exhibit inversion symmetry – are feasible on any modern PC. However, the computational cost scales with the cube of the number of atoms. Therefore, efficient use of resources, both memory and time, during calculations is paramount to successfully run simulations on more complex structures (structures with defects, surfaces, layered materials...).
This work aims to examine the performance of the software package WIEN2k [7, 8] and implement modifications to improve its efficiency.

[i] www.clarivate.com/webofsciencegroup
[ii] www.scopus.com

1.1 Structure of the Thesis

This thesis is divided into three main parts:

Firstly, a general overview of DFT is presented after a brief introduction into the relevant fundamentals of quantum mechanics is given (Chapter 2 – "Density Functional Theory"). Details of the software WIEN2k (a quick summary of its history, implementation of DFT, flow of programs, parallelization schemes...) are provided in Chapter 3 – "WIEN2k".

In the second part, Chapter 4 ("Performance and Optimization") deals with benchmarks of the initial parallel performance of core parts of WIEN2k as of the start of the thesis project, modifications and optimizations that were done during the project, as well as another set of benchmarks of the "new" parallel performance after the main part of the project was done.

Lastly, three use cases of WIEN2k are shown to demonstrate various applications that benefit from different kinds of parallelization:

(i) *Optimization of Atomic Positions*: In Chapter 5, calculations on small-to-medium-sized unit cells are presented, which were needed to optimize atomic positions of "building blocks" to set up the **O**rder-**D**isorder (OD) structure of Chlorodithionite in order to properly simulate intensity data of an **X**-**R**ay **D**iffraction (XRD) experiment.

(ii) *Adsorption on Surfaces*: Chapter 6 deals with medium-to-large unit cells of mixed magnesium/calcium oxide surfaces – which means additional vacuum has to be considered, increasing computational cost – and the behaviour of adsorbed water molecules.

(iii) *Error Estimation in DFT Codes*: In Chapter 7, the Δ-Project[iii] is introduced, which is an on-going large-scale project of multiple research groups. The aim is to create benchmark sets to compare different DFT codes and provide estimates for their precision.
Calculations within the scope of this Δ-Project were generally done on small unit cells, however, the sheer number of test cases (thousands of small unit cells had to be computed) and the required accuracy still present a computational challenge.

[iii] `https://molmod.ugent.be/deltacodesdft`

CHAPTER 2

Density Functional Theory – From Wave Functions to Electronic Densities

In fact, the mere act of opening the box will determine the state of the cat, although in this case there were three determinate states the cat could be in: these being Alive, Dead, and Bloody Furious.

Terry Pratchett, Lords and Ladies (1995)

One of the main goals of computational chemistry is to gain insights into characteristics and properties of materials (molecules, nanoparticles, solids...) by describing them using models and simulations run by computer programs. A particular type of models is used in so-called "ab-initio methods" – **D**ensity **F**unctional **T**heory (DFT), the focus of this thesis, is a reputable example of such methods: No empirical parameters or experimental inputs aside of a (rough estimate of a) structure are necessary. Only fundamental principles ("first principles") of quantum mechanics are used and the resulting equations (Schrödinger or Schrödinger-like equations[i]) are solved.

In quantum mechanics, a set of postulates is used as foundation (which are described in many introductory textbooks on the topic – e.g. references [9] and [10]). A thorough overview of the main postulates as well as derived ones can be found in reference [11].

[i] Strictly speaking, relativistic systems – e.g. systems with heavy elements – would require the usage of the Dirac equation instead; however, for the relevant considerations in this thesis, the Schrödinger equation is sufficient.

In short, the main postulates of quantum mechanics are:

- ***State***: The state of any given quantum mechanical system is uniquely and completely defined by a normalized complex wave function $\Psi(\vec{r}, s, t)$, where $\vec{r}$ are the position vectors of the particles, s describes the spin degrees of freedom, and t is the time. Ψ is the subject of a more detailed discussion in Section 2.1.

- ***Probability***: Ψ is interpreted as probability amplitude, such that the probability to find a given particle in the volume $dV = d\vec{r} = dxdydz$ is given by the square of the complex modulus $|\Psi|^2$. This can also be expressed as the product of Ψ with its complex conjugate Ψ^* (absolute square) according to:

$$|\Psi(\vec{r}, s, t)|^2 = \Psi^*(\vec{r}, s, t)\Psi(\vec{r}, s, t)d\vec{r}. \tag{2.1}$$

  This interpretation is the reason that a valid wave function Ψ must be normalized, since the following condition must hold:

$$\int_{-\infty}^{\infty} \Psi^*(\vec{r}, s, t)\Psi(\vec{r}, s, t)d\vec{r} = 1\,, \tag{2.2}$$

  as the probability to find the particle **somewhere** in space must be 1.

- ***Schrödinger Equation***: The time evolution of Ψ of a non-relativistic system is derived from the **T**ime-**D**ependent **S**chrödinger **E**quation (TDSE):

$$\mathrm{i}\hbar\frac{\partial\Psi(\vec{r}, s, t)}{\partial t} = \hat{\mathcal{H}}\Psi(\vec{r}, s, t)\,, \tag{2.3}$$

  where $\hat{\mathcal{H}}$ is the Hamilton operator, which is a linear Hermitian operator, and $\hbar$ is the reduced Planck constant $\left(\hbar = \frac{h}{2\pi}\right)$.
  In case of stationary wave functions (e.g. for systems in the ground state), no time dependency has to be considered. Therefore, the **T**ime-**I**ndependent **S**chrödinger **E**quation (TISE) can be used:

$$\hat{\mathcal{H}}\Psi(\vec{r}, s) = E\Psi(\vec{r}, s)\,, \tag{2.4}$$

  where E is the total energy of the system, corresponding to the Hamiltonian $\hat{\mathcal{H}}$.

- ***Correspondence Principle***: In quantum mechanics, a particular Hermitian operator corresponds to every observable in classical mechanics (momentum, energy...). Examples for such operators are the momentum operator $\hat{p}$:

$$\hat{p} = \mathrm{i}\hbar\frac{\partial}{\partial x}\frac{\partial}{\partial y}\frac{\partial}{\partial z}\,, \tag{2.5}$$

  and the kinetic energy operator $\hat{T}$:

$$\hat{T} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2}\frac{\partial^2}{\partial z^2}\,, \tag{2.6}$$

  where m is the mass of the given particle.

- ***Results of Measurements***: The result of a measurement of an observable A of a quantum system will be the eigenvalue a_i of the corresponding operator $\hat{A}$ if the state Ψ is an eigenfunction of $\hat{A}$ (cf. Equation 2.7) – the system will be in the corresponding eigenstate Ψ_i after the measurement.

$$\hat{A}\Psi(\vec{r}, s, t) = a_i\Psi(\vec{r}, s, t) \tag{2.7}$$

  If Ψ is not an eigenfunction of $\hat{A}$, the measurement will result in a distribution of values with an expectation value $\langle A\rangle$ given by:

$$\langle A\rangle = \frac{\int \Psi^*(\vec{r}, s, t)\hat{A}\Psi(\vec{r}, s, t)d\tau}{\int \Psi^*(\vec{r}, s, t)\Psi(\vec{r}, s, t)d\tau}\,, \tag{2.8}$$

  where $d\tau$ is a shorthand for an integration over all arguments of Ψ.

- ***Completeness of Eigenfunctions***: The linearly independent eigenfunctions of a Hermitian operator $\hat{A}$ form a complete basis, which means that a general state Ψ can be expanded as linear combination of eigenfunctions Ψ_i of $\hat{A}$:

$$\Psi = \sum_{i=1}^{n} c_i\Psi_i\,, \tag{2.9}$$

  where c_n are the linear coefficients and n might go to infinity.
  This is not per se its own postulate, but follows from the correspondence principle. However, this expansion plays an important role in many schemes to solve the Schrödinger equation (cf. Sections 2.1 and 2.5). Thus, it is mentioned explicitly.

Many fundamental principles of physics and chemistry can be derived directly from these postulates. As an example, Heisenberg's uncertainty principle directly follows from the correspondence principle – two measurements can only be independent if the corresponding operators commute.

2.1 The Wave Function Ψ

In order to treat a material computationally, a model of the state of a given system is necessary. The first step in such models is the "translation" of the state of a real material (atom, model, solid...) into a mathematical representation that defines the state (with respect to the constituents and geometry of the material):
In Figure 2.1, a schematic representation of a 4-particle system (two nuclei and two electrons – for instance H_2) is shown as an example: The positions of all particles are defined by position vectors $\vec{r}_1$, $\vec{r}_2$, $\vec{R}_1$, and $\vec{R}_2$ (by convention, electronic properties and positions are denoted by lower case letters; upper case letters are used for the nuclear analogues). Additionally, the masses of the nuclei M_1 and M_2, the electron mass m_e, and the charges of the particles (Z_1 and Z_2 for the nuclei and $-e$ for the electrons) as well as information about the spins of the electrons are needed for a complete description.

The second ingredient necessary for any computational treatment is a relation of the properties (e.g. the total energy) of the material to its state.
In case of quantum mechanical systems, the state is defined by a wave function Ψ depending on the quantities listed above, and the relation between Ψ and the properties of the system is given by the Schrödinger Equation.

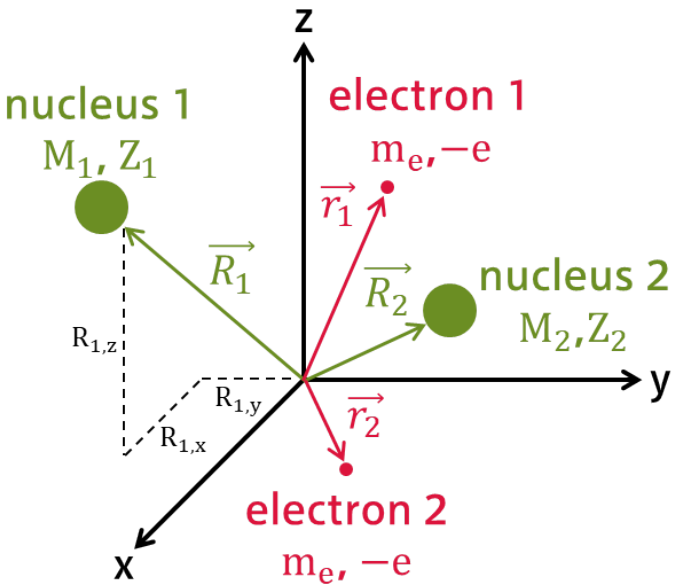

Figure 2.1: Schematic representation of a 4-particle system (comprised of two nuclei and two electrons, for instance H_2). The system is defined by the position vectors for all particles ($\vec{r}_1$, $\vec{r}_2$, $\vec{R}_1$, and $\vec{R}_2$), their masses (M_1, M_2, and m_e) and charges (Z_1, Z_2, and $-e$). By convention, lower case letters are used for electronic properties; upper case letters denote nuclear properties.

2.1.1 Schrödinger Equation

In 1926, Erwin Schrödinger published a series of communications [12–15] – which were summarized later that same year in his "Physical Review" publication "An Undulatory Theory of the Mechanics of Atoms and Molecules" [16] – in which he detailed a theory that relates "*material points*" [16] to what he calls "*wave-systems*" [16]. The ultimate consequence of Schrödinger's theory are the postulates that the state of a system is defined by an associated wave function $\Psi(\vec{r}, s, t)$ and that its dynamics are described by the time-dependent Schrödinger Equation (see above, Equation 2.3). In case of stationary states (e.g. ground states), the time-independent version can be used to obtain the total energy of any given system:

$$\hat{\mathcal{H}}\Psi(\vec{r}, s, t) = E\Psi(\vec{r}, s, t)\,. \qquad \text{(2.4 revisited)}$$

Depending on the system size (i.e. the number of particles in a given system), different constructions of the Hamiltonian $\hat{\mathcal{H}}$ have to be used:

The single particle Hamiltonian $\hat{\mathcal{H}}_{\text{sp}}$ is given by Equation 2.10 and includes kinetic energy contributions as well as the potential in which the particle is located.

$$\hat{\mathcal{H}}_{\text{sp}} = -\frac{\hbar}{2m}\nabla^2 + V(\vec{r})\,, \tag{2.10}$$

where m and $V(\vec{r})$ are the mass and the position-dependent potential of the particle, respectively, $\hbar$ is the reduced Planck constant, and ∇^2 is a shorthand notation for the sum of the partial double derivatives of the spatial coordinates x, y, and z:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\,. \tag{2.11}$$

In case of systems made up of more than one particle and more than one type of particle (electrons **and** nuclei), the many-body Hamiltonian $\hat{\mathcal{H}}_{\text{mb}}$ contains kinetic energy contributions for all particles as well as Coloumb interactions (attraction or repulsion) between particle pairs – as sums over all particles or possible pairs, respectively. It is given by:

$$\hat{\mathcal{H}}_{\text{mb}} = \hat{T}_{\text{e}} + \hat{T}_{\text{N}} + \hat{V}_{\text{eN}} + \hat{V}_{\text{NN}} + \hat{V}_{\text{ee}}\,, \tag{2.12}$$

where the following expressions for the individual operators are used:

kinetic energy of electrons:
$$\hat{T}_{\text{e}} = -\frac{\hbar^2}{2}\sum_{i}^{n}\frac{1}{m_i}\nabla_i^2\,, \tag{2.12a}$$

kinetic energy of nuclei:
$$\hat{T}_{\text{N}} = -\frac{\hbar^2}{2}\sum_{a}^{N}\frac{1}{M_a}\nabla_a^2\,, \tag{2.12b}$$

electron-nucleus attraction:
$$\hat{V}_{\text{eN}} = -\frac{1}{4\pi\varepsilon_0}\sum_{i}^{n}\sum_{a}^{N}\frac{Z_a e}{\vec{r}_{ai}}\,, \tag{2.12c}$$

electron-electron repulsion:
$$\hat{V}_{\text{ee}} = \frac{1}{4\pi\varepsilon_0}\sum_{i<j}^{n}\frac{e^2}{\vec{r}_{ij}}\,, \text{ and} \tag{2.12d}$$

nucleus-nucleus repulsion:
$$\hat{V}_{\text{NN}} = \frac{1}{4\pi\varepsilon_0}\sum_{a<b}^{N}\frac{Z_a Z_b}{\vec{R}_{ab}}\,. \tag{2.12e}$$

Here, n and N are the numbers of electrons and nuclei, respectively, m_i and M_a are the masses of electron i and nucleus a, respectively, and Z_a and Z_b are the nuclear charges of nuclei a and b. e is the elementary charge; and ε_0 is the vacuum permittivity. $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$, $\vec{r}_{ai} = \vec{R}_a - \vec{r}_i$, and $\vec{R}_{ab} = \vec{R}_a - \vec{R}_b$ are distances between pairs of particles – electron-electron, electron-nucleus, and nucleus-nucleus pairs.

2.1.2 Born-Oppenheimer Approximation

The **B**orn-**O**ppenheimer (BO) approximation [17] is a first step to reduce the dimensionality of the wave function Ψ – thus, making the solution of the Schrödinger equation more tractable.
The base consideration of this approximation is the fact that nuclei are substantially more massive than electrons ($m_{\text{electron}} \approx \frac{1}{1836}\, m_{\text{proton}}$) – consequently, their dynamics happen on vastly different time scales. Thus, the total wave function Ψ_{tot} is first separated into a product of an electronic part ψ_{elec} and a nuclear part ψ_{nuc} (which is not specific to the BO approximation but a common ansatz to solve partial differential equations)

$$\Psi_{\text{tot}} = \psi_{\text{elec}} \psi_{\text{nuc}}\,. \tag{2.13}$$

This allows splitting the total Hamiltonian $\hat{\mathcal{H}}_{\text{total}}$ in electronic and nuclear terms and subsequently neglecting the cross-terms – it is here where approximations take effect:
The kinetic energy of the nuclei is assumed to be 0 (the nuclei are kept "frozen" during the time scale of electronic motion) and ignored entirely. The electron-nucleus interactions (coulombic attraction) cannot be removed completely; however, they enter as static external Coulomb potential (v_{ext}) acting on the electrons. The fixed positions of the nuclei are now only parameters to determine this potential.
The remaining electronic Hamiltonian $\hat{\mathcal{H}}_{\text{elec}}$ is given by Equation 2.14:

$$\hat{\mathcal{H}}_{\text{elec}} = -\frac{\hbar^2}{2} \sum_i^n \frac{1}{m_i} \nabla_i^2 + \sum_i^N v_{\text{ext}}(\vec{r}_i) + \sum_{i<j}^N v_{\text{ee}}(\vec{r}_i, \vec{r}_j)\,; \tag{2.14}$$

where v_{ee} is the interaction potential between the electrons. $\hat{\mathcal{H}}_{\text{elec}}$ is then used to solve the electronic Schrödinger equation approximatively (cf. Section 2.1.3).
Obviously, the BO approximation does not give exact solutions; however, the errors remain small for many materials – especially in systems where nuclear motion does not play a significant role (e.g. in case of low-temperature simulations) and electrons are in the ground-state. Even if the approximation breaks down (e.g. in case of vibrational transitions [18] or occasionally in the presence of light elements like hydrogen [19]), it is often possible to add corrections [18–21].

2.1.3 Methods to Solve the Many-Body Schrödinger Equation

For the sake of brevity, **a**tomic **u**nits (a.u.) are introduced for the following considerations: In this unit system, distances are given in multiples of the Bohr radius a_0 (5.29×10^{-11} m), which corresponds to the most probable proton-electron distance in the ground-state hydrogen atom. The unit of energy is called hartree ($1\,\text{Ha} \approx 27.2\,\text{eV} \approx 4.36 \times 10^{-18}\,\text{J}$). $\hbar$, e^2, m_e, and the quantity $\frac{1}{4\pi\varepsilon_0}$ take the value of 1 a.u., leading to simplified equations with a reduced number of pre-factors.

After the BO approximation is applied, the electronic Schrödinger equation has to be solved to obtain the electronic wave function ψ_{elec} for a given set of nuclear coordinates. In the following, examples of possible ways to achieve such solutions are presented:

2.1.3.1 Exact and Numerical Solutions

Exact solutions are only possible in single-electron systems (hydrogen or "hydrogen-like" cations like H_2^+, He^+, Li^{2+}...) – the electron-electron interaction of multiple electrons would introduce additional terms, thus preventing the separation of variables. Such solutions are achieved by translating the Cartesian coordinates to spherical coordinates (i.e. giving the position by the distance to the origin r – usually put to the nucleus – and two angles θ and ϕ). Then, the (electronic) wave function ψ_{elec} depends on on these variables r, θ, and ϕ. Now, ψ_{elec} can be separated:

$$\psi_{\text{elec}}(r, \theta, \phi) = u(r) Y_{l,m}(\theta, \phi)\,, \tag{2.15}$$

where $u(r)$ is the radial part of the wave function that depends only on the distance to the nucleus, and $Y_{l,m}(\theta, \phi)$ are the famous spherical harmonics that contain the angular dependence of ψ_{elec}. They take different expressions for different values of the quantum numbers l and m and, thus, cause the different shapes of atomic orbitals (e.g. spherical for s-orbitals with $l = 0$, dumbbell-like for p-orbitals with $l = 1$, and so on).

Going beyond hydrogen (or related single-electron systems), it is no longer possible to solve the electronic Schrödinger equation exactly. In principle, one can use numerical methods to solve many-body equations, however, these methods grow increasingly costly with the number of electrons to be considered.

2.1.3.2 Hartree-Fock

An alternative approach is to use combinations of single-particle wave functions ϕ instead of the many-body wave function ψ_{elec}. One possibility is the so-called Hartree product [22]:

$$\Psi(\vec{r_1}, \vec{r_2}, \ldots, \vec{r_N}) = \prod_i^N \phi_i(\vec{r_i}) \tag{2.16}$$

that simply combines all ϕ_i multiplicatively. However, this ansatz is symmetric and, thus, does not satisfy the Pauli exclusion principle (which requires electronic wave functions to be anti-symmetric) [23, 24]. As a consequence, the exchange energy (the energy contribution to the total energy of a quantum system caused by the presence of indistinguishable particles without classical analogue) is neglected completely.

The **H**artree-**F**ock (HF)-method, an extension of the Hartree method, uses the same single-particle wave functions to construct Slater determinants instead [25, 26]:

$$\Psi(\vec{r_1}, \vec{r_2}, \ldots, \vec{r_N}) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_1(\vec{r_1}) & \phi_2(\vec{r_1}) & \ldots & \phi_N(\vec{r_1}) \\ \phi_1(\vec{r_2}) & \phi_2(\vec{r_2}) & \ldots & \phi_N(\vec{r_2}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\vec{r_N}) & \phi_2(\vec{r_N}) & \ldots & \phi_N(\vec{r_N}) \end{vmatrix}, \tag{2.17}$$

where N is the number of electrons and $\frac{1}{\sqrt{N!}}$ is a normalization factor.

Solving a Schrödinger equation in the HF framework results in a set of parameters ϵ_i that are interpreted as energies of single-particle orbital according to Koopmans' theorem [27]:

Theorem 1:
Koopman's Theorem

The ionization energy I of a given electron is equal to the negative orbital energy ϵ.

$$I = -\epsilon \tag{2.18}$$

The usage of such determinants recovers the anti-symmetry, as swapping of two columns (corresponding to pairwise electron swaps) will change the sign.
While using a determinant instead of a product accounts for exchange, electronic correlation (another type of electron-electron interaction exclusive to quantum systems that arises, for instance, from interactions of electrons of opposite spin) is still not included. In fact, the formal definitions of exchange and correlation energies are related to the Hartree and Hartree-Fock methods, respectively: The exchange energy is the difference of the energy given by the Hartree method compared to the HF energy; the correlation energy is the difference between the unknown "true" energy and the HF energy.

2.1.3.3 Post-Hartree-Fock Methods

To account for missing electronic correlation, so-called "post-Hartree-Fock"-methods are used to add correlation using the HF result as starting point (at increased computational cost compared to "pure" HF). Widely used methods include (details can be found in ref. [28]):

- **C**onfiguration **I**nteraction (CI):
  Here, the wave function is set up by a linear combination of additional Slater determinants corresponding to electronically excited states (configurations). "Interaction" in the name of the method refers to the mixing of these different configurations.
- **M**øller–**P**lesset (MP) perturbation theory [29]:
  In general, perturbation theory treats complex systems by adding a (small) perturbation to a (Hamilton) operator of a Schrödinger equation with an exact solution. These unperturbed operators are derived from simpler models that cannot accurately describe the given systems. In MP perturbation theory, the Fock operator of the HF-method is used as unperturbed operator and the correlation potential corresponds to the perturbation.
- **C**oupled **C**luster (CC):
  Similarly to CI, CC methods use excited Slater determinants; however, in contrast to CI, they are added perturbatively to the reference wave function derived via HF.
- Quantum Monte Carlo methods

2.2 Foundation of Density Functional Theory

One of the main problems when solving the electronic Schrödinger equation – both numerically or by means of wave function based methods like (post-)HF – is the fact that the electronic wave function ψ_{elec} still depends on $4\,N$ parameters – three spatial coordinates and one spin coordinate for each of the N electrons of the system at hand. As a direct consequence, the computational cost of calculations grows rapidly; and one quickly runs into what Walter Kohn called "*an exponential wall*" [30] that severely limits the size of tractable systems.

This exponential wall is a twofold challenge. Firstly, the computation times of wave function based methods increase exponentially with the system size N – HF, for instance, nominally scales with $\mathcal{O}(N^4)$ [31], which means that a system with twice the number of electrons would take 16 (2^4) times longer to compute. Post-HF methods tend to behave even worse than that: compute times of coupled cluster methods for instance exhibit naive scaling in the order of $\mathcal{O}(N^8)$ [32][ii].
Secondly, the memory requirements to store the necessary data blow up exponentially as well. This second problem can be illustrated with a simple thought experiment: Suppose, one wants to numerically calculate – and subsequently store – the wave function of a single oxygen atom with 8 electrons. The wave function Ψ_{O} (Equation 2.19) then depends on the positions of those electrons, or in other words, on 24 Cartesian coordinates (leaving aside the spin coordinate for each electron):

$$\Psi_{\text{O}} = \Psi_{\text{O}}(\vec{r}_1, \vec{r}_2, ..., \vec{r}_8) = \Psi_{\text{O}}[(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_8, y_8, z_8)]\,. \tag{2.19}$$

If Ψ_{O} is to be expressed on a course grid of only 10 grid points per dimension, 10^{24} values have to be saved. Assuming only the real part of Ψ_{O} is stored and a single value takes 8 B of memory (nowadays, this so-called "double precision" is widely used)[iii] leads to a total memory requirement of 8×10^{24} B or 8×10^9 PB. By comparison, the currently most powerful supercomputer (Frontier at the Oak Ridge National Laboratory, United States)[iv] has a total shared storage capacity of about 700 PB.

A possible way to work around these issues of wave function based methods is to use DFT instead. DFT replaces wave functions with electron densities, thus reducing the number of spatial coordinates from $3\,N$ to just 3.

[ii] In practice, it is usually possible to achieve better performance by leaving aside negligible contributions (provided they can be determined – inexpensively – on the fly), by using advanced, sometimes specially-tailored, algorithms, or by introducing additional approximations or simplifications. The scaling of HF for instance can be reduced even below $\mathcal{O}(N^3)$ [31].

[iii] Storing the complete complex wave function would take 16 B per value, thus doubling the required memory.

[iv] According to `www.top500.org`, which is a ranking of supercomputers compiled by Jack Dongarra, Erich Strohmaier, and Horst Simon twice a year. Its latest iteration of November 2022 lists Frontier at the top spot.

2.2.1 Hohenberg-Kohn Theorems

The foundation of DFT is built by two theorems formulated and proven by Pierre Hohenberg and Walter Kohn in 1964 [33]. The observable quantity "electron density" $\rho(\vec{r})$ lies at the heart of these theorems (and, subsequently, DFT); and it is defined in Equation 2.20:

$$\rho(\vec{r}) = N \int \cdots \int |\Psi(\vec{r}_1, \vec{r}_2, \vec{r}_3, \ldots, \vec{r}_n)|^2 d\vec{r}_2 d\vec{r}_3 \ldots d\vec{r}_n \tag{2.20}$$

$$N[\rho] = \int \rho(\vec{r}) d\vec{r} = N \tag{2.21}$$

The additional requirement for ρ that integrating over all of space must give the number of electrons N (Equation 2.21) is automatically contained in Equation 2.20, since only normalized wave functions Ψ are allowed.
As mentioned before (cf. Section 2.1), any given system (atom, molecule, or solid) is defined by the position of its constituents and their respective masses and charges. After the BO approximation is applied and the nuclear positions are fixed, the system can be summarily described by the external potential $v_{ext}(\vec{r})$ generated by the nuclear charges at their positions (cf. Figure 2.1), in which the electrons move. As $v_{ext}(\vec{r})$ enters into the electronic Hamiltonian $\hat{\mathcal{H}}_{\text{elec}}$ (Equation 2.14), it determines the wave function of the system (e.g. obtained via solving the TISE) and, subsequently, the properties of the system (e.g. total energy E_{tot}, the electron density $\rho \ldots$):

$$v_{ext} \rightarrow \hat{\mathcal{H}} \rightarrow \Psi \longrightarrow \textbf{all} \text{ properties } (E_{\text{tot}}, \rho ...) \,. \tag{2.22}$$

According to Hohenberg and Kohn, it can be shown "[...]*that conversely* $[v_{ext}(\vec{r})]$ *is a unique functional of* $[\rho(\vec{r})]$*, apart from a trivial additive constant.*" [33] This is the first Hohenberg-Kohn theorem:

Theorem 2:
1st Hohenberg-Kohn Theorem

The ground-state (GS) electron density $\rho_{GS}(\vec{r})$ of a given system unambiguously defines the electron number N **and** *the external potential $v_{ext}(\vec{r})$; and thus, subsequently, defines $\hat{\mathcal{H}}$ for the system and – by extension – everything else of the system (wave function, energy and derived properties...). Therefore:*

$$E_{GS} = E[\rho_{GS}] \,, \tag{2.23}$$

where E_{GS} is the energy of the ground-state [33, 34].

While the first theorem of Hohenberg and Kohn does prove that the energy of the ground-state of a system is a functional of the corresponding electron density ρ_{GS} and that there is no other system (with a different wave function: $\Psi \neq \Psi'$) that can have the same density, no further information about the nature of this functional is given.

The second Hohenberg-Kohn theorem now states that such a functional does not only exist, it can also be used to find the ground-state energy E_{GS} by way of a variational principle:

Theorem 3:
2^{nd} Hohenberg-Kohn Theorem – Density Variational Principle

A functional $F[\rho]$ of the electron density $\rho(\vec{r})$ exists that allows to express the energy of a system in a given potential v_{ext} as:

$$E_{v_{ext}}[\rho] = F[\rho] + \int v_{ext}(\vec{r})\rho(\vec{r})d\vec{r} \geq E_{GS}\,. \tag{2.24}$$

The second equality in Equation 2.24 holds true, if $\rho(\vec{r})$ is the density of the ground-state (i.e. for $\rho_{GS}(\vec{r})$ the functional $F[\rho]$ minimizes the energy).
Moreover, $F[\rho]$ is universal – valid for any number of particles and any external potential [33].

So far, no approximations or simplifications have been used – Equation 2.24 would give the true ground-state energy, provided $F[\rho]$ is known. However, the exact expression of $F[\rho]$ is **not** known, therefore, approximations have to be introduced.

2.2.2 Approximating $F[\rho]$

Approximating $F[\rho]$ in its entirety would introduce errors of the order of more than 10 %, thus failing to describe real systems sufficiently well for results to be predictive. Therefore, a more sophisticated approach to approximating $F[\rho]$ with the goal of achieving smaller errors is needed:

In the wave function picture, the total electronic energy E_{elec} can be split into several contributions: the kinetic energy T, the electronic interaction energy E_{ee}, and the external potential energy E_{ext}. E_{ee} can be split further into a classical Coulomb energy part E_{Coul} (arising from repulsive electron-electron interaction of a continuous classical charge distribution of density ρ) and two contributions without classical analogues that stem from the quantum nature of electrons: electronic exchange E_{x} (resulting from the exchange interaction between indistinguishable particles) and correlation E_{c} (which mostly arises from interactions of electrons of opposite spin)[v]. E_{elec} is then given by

$$E_{\mathrm{elec}} = T + E_{\mathrm{Coul}} + E_{\mathrm{x}} + E_{\mathrm{c}} + E_{\mathrm{ext}}\,, \tag{2.25}$$

with

$$E_{\mathrm{ee}} = E_{\mathrm{Coul}} + E_{\mathrm{x}} + E_{\mathrm{c}}\,. \tag{2.25a}$$

[v] As mentioned in Section 2.1.3.2, E_{x} is formally defined as the difference between the Hartree limit (neglecting exchange and correlation completely) and the Hartree-Fock limit (which treats exchange exactly, but neglects correlation completely. Analogously, E_{c} is the difference between the Hartree-Fock limit and the (unknown) true ground state energy.

A reasonable ansatz to construct the Hohenberg-Kohn functional is to assume an analogous split of $F[\rho]$ into a sum of functionals (Equation 2.26) and search for expressions for the individual parts.

$$F[\rho] = T[\rho] + E_{\text{Coul}}[\rho] + E_{\text{x}}[\rho] + E_{\text{c}}[\rho] \tag{2.26}$$

Note that $E_{\text{ext}}[\rho]$ is not a part in the construction of the unknown functional $F[\rho]$ as it is non-universal – it depends on the system at hand – while all parts of the sum in Equation 2.26 are universal. Moreover, $E_{\text{ext}}[\rho]$ is one of two contributions mentioned in Equation 2.25 that can be written down exactly as a functional of ρ (cf. Equation 2.24, second term on the right); the other being the Coulomb energy $E_{\text{Coul}}[\rho]$. All other parts need to be approximated; however, by being able to express $E_{\text{Coul}}[\rho]$ exactly, the error introduced is already smaller.

Two main approaches to tackle the approximations of the remaining terms (with the goal of reducing errors even further) exist:

- ***Kohn-Sham-DFT***
  In **K**ohn-**S**ham (KS)-DFT [35], single-particle functions Φ_i – called KS-orbitals – are introduced, thus replacing the many-body Hamiltonian with a sum of single-particle Hamiltonians. This method (details are presented in Section 2.3) is used in WIEN2k [7, 8] and is widely spread due to its high accuracy.

- ***Orbital-free DFT***
  Orbital-free (sometimes called "pure") DFT aims to find explicit expressions for the functionals and is based on the Thomas-Fermi model [36, 37], in which a way to calculate electronic energies from the density ρ was developed already in 1927. This model uses the kinetic energy of an **U**niform **E**lectron **G**as (UEG) T^{UEG} as approximation for T and approximates E_{ee} with E_{Coul} (neglecting exchange and correlation entirely):

  $$E_{\text{TF}}[\rho] = T^{\text{UEG}}[\rho] + E_{\text{Coul}}[\rho] + E_{\text{ext}}[\rho]\,, \tag{2.27}$$

  leading to

  $$E_{\text{TF}}[\rho] = C_1 \int \rho(\vec{r})^{\frac{5}{3}} d\vec{r} + \frac{1}{2}\int\int \frac{\rho(\vec{r})\rho(\vec{r}')}{|\vec{r}' - \vec{r}|} d\vec{r} d\vec{r}' + \int v_{ext}(\vec{r})\rho(\vec{r}) d\vec{r}\,, \tag{2.27a}$$

  with

  $$C_1 = \frac{3}{10}(3\pi^2)^{\frac{2}{3}}\,. \tag{2.27b}$$

  In 1930, this model was improved by Dirac [38] – leading to the **T**homas-**F**ermi-**D**irac (TFD) model, which added a functional for the exchange energy (second term in Equation 2.28) – again, the UEG was used as approximation:

  $$E_{\text{TFD}}[\rho] = E_{\text{TF}}[\rho] + C_2 \int \rho(\vec{r})^{\frac{4}{3}} d\rho(\vec{r})\,, \tag{2.28}$$

with

$$C_2 = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} . \tag{2.28a}$$

The advantage of orbital-free DFT (compared to KS-DFT) lies in the fact that it is much faster computationally – which makes it usable for large systems. However, it is less accurate; the underlying model (cf. Equation 2.27a), for instance, predicts molecules to be unstable in general, as all atoms repel each other [39]. Moreover, energies calculated by TFD exhibit errors of about 10 %, and the model fails to properly replicate the shell structure of atoms [35].
Despite these shortcomings, orbital-free DFT is still developed and refined, and more advanced versions of these models are in use [40].

2.3 Formalism of Kohn-Sham

In this formalism proposed by Kohn and Sham in 1965 [41], the "real" system of n interacting electrons is replaced by a system of non-interacting electrons with the same electron density. Thus, instead of the many-body wave function, a coupled system of n single-particle orbitals Φ_i is used to describe the material. The density $\rho(\vec{r})$ is then defined as

$$\rho(\vec{r}) = \sum_{i=1}^{n} |\Phi_i^2| , \tag{2.29}$$

where Φ_i are the KS orbitals and n is the number of occupied KS orbitals (i.e. n is chosen such that the orbital energies ϵ_i of all considered orbitals are smaller or equal to the Fermi energy E_F: $\epsilon_i \leq E_F$).

The non-interacting kinetic energy T_s for this system is given by:

$$T_s = -\frac{1}{2}\int \Phi^*(\vec{r})\nabla^2\Phi(\vec{r})d\vec{r}, \tag{2.30}$$

and the electronic interaction energy E_{ee} is approximated with the classical Coloumb energy:

$$E_{\text{ee}}[\rho] \approx E_{\text{Coul}}[\rho] = \frac{1}{2}\int\int \frac{\rho(\vec{r})\rho(\vec{r}')}{|\vec{r}' - \vec{r}|} d\vec{r}d\vec{r}' . \tag{2.31}$$

Using T_s and E_{Coul} in Equation 2.26 gives

$$F[\rho] = T_s[\rho] + E_{\text{Coul}}[\rho] + E_{\text{xc}}[\rho] , \tag{2.32}$$

with

$$E_{\text{xc}}[\rho] = (T[\rho] - T_s[\rho]) + (E_{\text{ee}}[\rho] - E_{\text{Coul}}[\rho]) . \tag{2.32a}$$

While the expressions T_s and E_{Coul} can be written down exactly, E_{xc} is set up in a way that it only contains the differences between the interacting and non-interacting kinetic energies (first term in E_{xc}) and between the electronic interaction energy E_{ee} and the Coloumb energy E_{Coul} (second term in E_{xc}).

This definition of E_{xc} is the central point of the KS-formalism: The total KS energy is given by

$$E_{\text{elec}} = T_s[\rho] + E_{\text{ext}}[\rho] + E_{\text{Coul}}[\rho] + E_{\text{xc}}[\rho]\,. \tag{2.33}$$

The only term of this equation that cannot be expressed explicitly and needs approximations is E_{xc} – which can be reasonably assumed to be small (as it only contains the above mentioned differences).

Applying the variational principal leads to the KS-Equations:

$$\left(-\frac{1}{2}\nabla^2 + v_{\text{eff}}(\vec{r})\right)\Phi_i(\vec{r}) = \epsilon_i\Phi(\vec{r})\,, \tag{2.34}$$

where v_{eff} is defined as:

$$v_{\text{eff}} = v_{\text{ext}}(\vec{r}) + v_{\text{Coul}}(\vec{r}) + v_{\text{xc}}(\vec{r})\,, \tag{2.35}$$

with

external potential: $$v_{\text{ext}} = \frac{Z}{r}\,, \tag{2.35a}$$

Coulomb potential: $$v_{\text{Coul}} = \frac{1}{2}\int\int \frac{\rho(\vec{r})\rho(\vec{r}')}{|\vec{r}' - \vec{r}|}d\vec{r}d\vec{r}'\,, \text{ and} \tag{2.35b}$$

exchange-correlation potential: $$v_{\text{xc}} = \frac{\partial E_{\text{xc}}[\rho]}{\partial\rho}\,. \tag{2.35c}$$

2.4 Exchange-Correlation Functionals

Having arrived at the KS equations, an approximation for the e**x**change-**c**orrelation (XC) part of the functional construction (Equation 2.32) is needed.
A hierarchical ranking has been proposed by Perdew and Schmidt, which they named "Jacob's Ladder"[vi] [42]. Starting from the bottom (what Perdew and Schmidt call the "*Hartree World*" [42]) without any exchange-correlation included, one can "climb the ladder" by adding improvements to the description of E_{xc} (and thus increasing computational cost as well) until "DFT Heaven" (i.e. chemical accuracy with errors below $1\,\text{kcal}\,\text{mol}^{-1} = 0.043\,\text{eV}$) is reached:
The first three rungs are the so-called "semi-local" approximations that contain only one integral:

$$E_{\text{xc}}^{\text{semi-local}} = \int \rho(\vec{r})\epsilon_{\text{xc}} \tag{2.36}$$

where ϵ_{xc} is the XC energy per unit volume that depends on the density and its derivatives (depending on the specific approximations) [42, 43].

[vi] Inspired by a ladder to heaven dreamt up by Jacob in the Bible: "*Jacob left Beer-sheba and went toward Haran. He came to a certain place and stayed there for the night, because the sun had set. Taking one of the stones of the place, he put it under his head and lay down in that place. And he dreamed that there was a ladder set up on the earth, the top of it reaching to heaven; and the angels of God were ascending and descending on it.*" (Genesis 28.10-12)

2.4.1 Local Density Approximation (LDA)

The **L**ocal **D**ensity **A**pproximation (LDA) is a very simple approach (the "first rung"), already proposed by Kohn and Sham in 1965 [41], and uses an ϵ_{xc} that only depends on the local density:

$$E_{xc}^{LDA}[\rho] = \int \rho(\vec{r})\epsilon_{xc}(\rho(\vec{r}))d\vec{r}. \tag{2.37}$$

The main assumption of LDA now is that $\epsilon_{xc} = \epsilon_{xc}^{UEG}$ – which is reasonable for systems with slowly varying densities; and agrees surprisingly well with experimental data for atoms and molecules [35].
A second assumption (generally used when constructing functionals) is that ϵ_{xc} is just the sum of an exchange contribution ϵ_x and a correlation contribution ϵ_c. ϵ_x for the UEG is explicitly known from TFD theory [38]:

$$\epsilon_x^{UEG} = \epsilon_x^{LDA} = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} \int \rho(\vec{r})^{\frac{4}{3}} d\vec{r}. \tag{2.38}$$

ϵ_c^{UEG}, on the other hand, cannot be written down in a closed form. Instead, quantum Monte-Carlo data (e.g. from [44]) are used to find and parametrize analytic representations for ϵ_c^{UEG} [45–47].

LDA also allows the treatment of spin-polarized systems, where the spin-densities $\rho_\uparrow$ and $\rho_\downarrow$ replace the total density ρ (with $\rho = \rho_\uparrow + \rho_\downarrow$), since up- and down-spin states are no longer occupied in equal parts. The exchange energy of this **L**ocal **S**pin **D**ensity **A**pproximation (LSDA) still has an analytical form [42, 48]:

$$E_x^{LSDA}(\rho_\uparrow, \rho_\downarrow) = \frac{1}{2}\left(E_x^{LSDA}(2\rho_\uparrow) + E_x^{LSDA}(2\rho_\downarrow)\right). \tag{2.39}$$

The correlation energy in the LSDA has to be solved numerically [49].

In an expansive study that compared more than 60 functionals of all flavours using a test set of 44 strongly bound solids of Tran and coworkers [43], LDA agrees reasonably well with experimental lattice constants (with a **M**ean **R**elative **E**rror (MRE) of $-1.5\,\%$), while (severly) overestimating bulk moduli (MRE of $8.1\,\%$) and cohesive energies (MRE of $17.2\,\%$). The too small lattice constants together with too large cohesive energies nicely illustrate LDA's systematic overbinding.

2.4.2 Generalized Gradient Approximation (GGA)

While LDA is a good starting point and works reasonably well for certain cases, clearly real materials exist that cannot be (approximatively) described by the uniform electron gas. Therefore, functionals that capture more complex densities in a more realistic way beyond the LDA are necessary as well.
In **G**eneralized **G**radient **A**pproximations (GGAs), gradient corrections are introduced, i.e. ϵ_{xc} now additionally depends on the gradients of the spin densities:

$$E_{xc}^{GGA}(\rho_\uparrow, \rho_\downarrow) = \int d\vec{r} f(\rho_\uparrow, \rho_\downarrow, \nabla\rho_\uparrow \nabla\rho_\downarrow). \tag{2.40}$$

The construction of functionals of this family (as well as the functionals of the next rung) is no longer clear-cut, instead, there exist two philosophies:

- Semi-empirical approximations contain initially undetermined parameters fitted against accurate atomic data (spearheaded by Axel Becke [50]).
- Non-empirical approximations do not use fitted parameters at all, but are constructed obeying theoretical constraints, e.g. exactly reproducing the uniform electron gas or the second gradient expansion in the limit of slowly varying densities [6, 51]. Perdew, who laid seminal groundwork in this particular field, called this ansatz *"constraint satisfaction"* [51]. Arguably among the most "famous" GGAs is **P**erdew-**B**urke-**E**rnzerhof (PBE) [52], which – despite being introduced more than 25 years ago – is still widely used[vii]. Other well-known examples include Wu-Cohen (WC) and PBEsol [53], which is a revised version of PBE to improve properties of solids.
- Mixed approaches are possible as well: Zhang and Yang, for example, obtain a different value for a certain parameter of PBE (by fitting exchange-only atomic energies to results from another method, thus creating a revised version of PBE called "revPBE" that substantially improves PBE's atomic total energies as well as reducing errors in atomization energies for molecules [54].

GGAs give slightly improved lattice constants and bulk moduli in strongly bound solids compared to LDA; however, they greatly improve cohesive energies [43]. Notably, most GGAs invert LDA's overbinding (leading to too small lattice constants and too large cohesive energies) by underestimating bond strengths giving (slightly) too large lattice constants.

2.4.3 meta-GGA

meta-**GGA**s (mGGAs) add even more corrections to $E_{\mathrm{xc}}^{\mathrm{LDA}}$: While most mGGAs found in literature use the kinetic-energy density τ (which is given by Equation 2.41a), mGGAs that use Laplacians $\nabla^2\rho$ instead have been proposed as well [55]. Using τ, E_{xc} is then given as:

$$E_{\mathrm{xc}}^{\mathrm{mGGA}} = \int \rho(\vec{r})\epsilon_{\mathrm{xc}}^{\mathrm{mGGA}}(\rho_\uparrow, \rho_\downarrow, \nabla\rho_\uparrow\nabla\rho_\downarrow, \tau_\uparrow, \tau_\downarrow)d\vec{r} \tag{2.41}$$

with

$$\tau(\vec{r}) = \frac{1}{2}\sum_{i}^{\mathrm{occ}} |\nabla\phi_i(\vec{r})|^2 \tag{2.41a}$$

A very promising non-empirical mGGA in that it fulfils all known constraints known for mGGAs is SCAN ("Strongly Constraints and Appropriately Normed Functional") [56]. Other well-known functionals are, for example, PKZB (a mGGA that has been proposed more than 20 years ago) [57], TPSS [58], revTPSS [59], and the empirical MGGA_MS2

[vii] The three use cases of WIEN2k reported in this dissertation (cf. Chapters 5–7) utilize PBE as well.

[60]. Moreover, specialised mGGAs have been published as well, e.g. mBEEF [61] that has been trained (i.e. parametrized) on real data especially for surface chemistry studies.

In their functional test on solids [43], Tran et al. found mGGAs to perform similarly to GGAs in case of strongly bound solids (while the results on lattice constants and bulk moduli are very comparable, the mGGAs appear to improve cohesive energies very slightly). For the weakly bound solids, most mGGAs tested improve the GGA results of both lattice constants and cohesive energies (but still severely deviate from the experimental values).
As the authors mention explicitly, a clear advantage of some mGGAs (e.g. MGGA_MS2 and SCAN) over GGAs is the fact that GGAs usually cannot well describe molecules and solids at the same time while mGGAs yield satisfactory results for both types of systems. Therefore, mGGAs should be the better choice in mixed systems (e.g. when studying adsorption of small molecules on surfaces) [43].

2.4.4 Beyond Semi-Local Functionals

All approximations so far attempted to improve upon the basis that was set up by the LDA – via the inclusion of more information about the system (gradient of ρ, kinetic energy density τ and/or second derivatives of ρ). Beyond that, it is no longer straightforward to add systematic corrections to (try to) improve a functional.

One possible approach is breaking the (semi-)locality of LDA and (m)GGAs by adding non-local contributions (for instance via the inclusion of exact exchange from the UEG): Becke suggested in 1993 using hybrid functionals – i.e. replacing a fraction of E_x of a given functional with exact exchange (e.g. from HF, E_x^HF) [62] to reduce a slight overbinding tendency still present in some GGAs. A general ansatz for a hybrid functional is given here (instead of GGAs, mGGAs can be used as well):

$$E_\mathrm{xc}^\mathrm{hybrid} = E_\mathrm{xc}^\mathrm{LDA/GGA} + a\left(E_\mathrm{x}^\mathrm{exakt} - E_\mathrm{x}^\mathrm{LDA/GGA}\right) + b\Delta E_\mathrm{x}^\mathrm{GGA} + c\Delta E_\mathrm{c}^\mathrm{GGA}\,, \tag{2.42}$$

where a is the fraction of exact exchange "mixed in" and b and c are some fractions of included gradient corrections for E_x and E_c (in case of a (m)GGA being used, b and c will be 0). Becke used $a = 0.20$ with hybrid-LDAs [62], while Perdew et al. found $a = 0.25$ to be a better choice in hybrid-GGAs [63].

A variety of other methods exist to fix known shortcomings of density functionals (e.g. the failure to describe strongly correlated ground states) or extend them to include more "real physics" like non-local interactions (e.g. London dispersion or van der Waals interactions in general). Examples for such methods are the **R**andom **P**hase **A**pproximation (RPA)[64], DFT+U [65], or DFT-D3 [66]. Functionals that go even beyond hybrid functionals are, for instance, non-local van der Waals (nlvdw) functionals [67, 68] or so-called double-hybrid functionals, that include HF exchange and correlation from pertubation theory [69, 70].

2.5 Solving the KS-Equations – Eigenvalue Problem

Introducing any approximation for the unknown parts of the density functional in the KS-formalism leads to a set of coupled single-particle equations (see Eq. 2.34) that still cannot be solved exactly, since there is no analytical expression for the single-particle orbitals Φ_k. Therefore, the Φ_k are expanded into basis functions ϕ_i (i.e. Φ_k is represented by a linear combination of the basis functions comprising the basis set):

$$\Phi_k(\vec{r}) = \sum_n c_i \phi_i(\vec{r})\,. \tag{2.43}$$

To find the coefficients c_i of the expansion of Φ_k, the variational principle is applied to find the minimum of E_k. From the postulates of quantum mechanics follows that the energy E_k of a state Φ_k can be derived from the eigenvalues from the Hamiltonian $\hat{\mathcal{H}}$:

$$\langle E_k \rangle = \frac{\int \Phi_k^*(\vec{r})\hat{\mathcal{H}}\Phi_k(\vec{r})d\vec{r}}{\int \Phi_k^*(\vec{r})\Phi_k(\vec{r})d\vec{r}}\,. \tag{2.44}$$

$$\frac{\partial E_k}{\partial c_i} = 0\,. \tag{2.45}$$

Setting the derivatives of E_k with respect to the coefficients c_i to 0 (Eq. 2.45) leads to the secular equations (a set of linear equations), the matrix representation of which is given by:

$$HC = ESC\,, \tag{2.46}$$

with

$$H_{m,n} = \int \phi_m^* \hat{\mathcal{H}} \phi_n d\vec{r} \tag{2.46a}$$

and

$$S_{m,n} = \int \phi_m^* \phi_n d\vec{r}\,. \tag{2.46b}$$

Here, H and S denote the Hamilton and the overlap matrix, respectively, which are obtained by solving the integrals shown above. C is the eigenvector matrix, and the diagonal matrix E contains the corresponding eigenvalues ϵ_k.

Despite the fact that the eigenvalues ϵ are usually called "(one-electron) orbital energies", they do not a priori correspond to physical energies[viii] [71]. They arise during the solution of Equation 2.45, where they are introduced as Lagrange multipliers to find the minimum of E_k.

Within the KS-formalism, the physical meaning of ϵ_i is described by Janak's theorem [72].

[viii] The highest occupied orbital is the exception – it corresponds to the negative ionization energy $-I$.

Theorem 4:
Janak's Theorem

Each ϵ_i (corresponding to a given orbital Φ_i) relates to the derivative of the total energy E with respect to the occupation n_i of the orbital Φ_i:

$$\epsilon_i = \frac{\partial E}{\partial n_i} \,. \tag{2.47}$$

The choice of basis functions depends on the system one wants to treat and should be informed by the underlying physics. For instance, atom centred basis functions are typically used to describe atoms/molecules in quantum chemistry, while basis sets based on plane waves are well-suited for periodic materials. Thus, different implementations of DFT (in different codes) expand the KS-orbitals in different basis sets. The following (non-exhaustive) list gives a few examples of widely used basis sets:

- **S**later-**T**ype **O**rbitals (STOs) and **G**aussian-**T**ype **O**rbitals (GTOs):
  STOs and GTOs are widely used examples of localized orbitals. They are used to approximate the radial part of "hydrogen-like orbitals" (i.e. orbitals obtained by solving the Schrödinger equation of e.g. hydrogen). While the radial behaviour of STOs reproduces that of hydrogen-like orbitals close to the nucleus ($\phi(r) \propto e^{-\alpha r}$), they do not exhibit radial nodes. This issue can be addressed by using more than one STOs per atomic orbital.
  Due to their discontinuity (they display a cusp at $r = 0$), STOs are computationally challenging and usually are replaced by GTOs ($\phi(r) \propto e^{-\alpha r^2}$), which can be handled more easily. To recover the proper behaviour near the nucleus, multiple GTOs per STO are needed though.

- Plane waves:
  Plane waves are cheap to compute and appear promising for periodic systems (due to their own periodicity), however, they are not well suited to describe fast variations close to the nucleus, as a large number of plane waves would be needed to adequately describe this part of the potential. Therefore, the "real" potential is replaced with a so-called pseudopotential [73, 74] that smooths the oscillations near the nucleus but reproduces the potential farther away (which is important for chemical bonding) – leading to a smaller number of necessary plane waves.

- Augmented (or mixed) basis sets:
  Augmented basis sets – which are used in WIEN2k – are an alternate approach to address the shortcomings of plane waves. The modeled material is partitioned into different regions where different basis functions are employed: In regions close to the nuclei, basis functions capable of mimicking an oscillating wave function are used (in case of WIEN2k, these are linear combinations of radial functions and spherical harmonics – more details see in Section 3.2), and in regions of slowly varying potentials (few) plane waves are sufficient.

- **P**rojector **A**ugmented **W**aves (PAWs) [75]: Proposed by Blöchl in 1994, PAWs offer a different approach compared to augmented basis sets. In this method, the pseudo wave function $\tilde{\psi}$ (which is usually expanded into plane waves) is the variational quantity. Outside of an atomic sphere, $\tilde{\psi}$ is equal to the true wave function ψ, while inside of spheres it is coupled via a projector function p to an atomic all-electron partial wave expansion ϕ.

Further details on different basis set choices in electronic structure calculations can be found in textbooks on the topic – e.g. reference [76].

2.6 Self Consistent Field (SCF)

When trying to solve the KS equations, one runs into a circular dependency: The ultimate goal is to obtain the ground-state density $\rho(\vec{r})$ of a given system, which requires knowledge about the effective potential v_{eff} (cf. Equation 2.35) as it enters into the KS Hamiltonian. However, $\rho(\vec{r})$ is needed to calculate v_{eff} since the latter is a functional of the former.

The appearance of dependencies like this is common in mean-field approximations, which use an averaged "mean field" to describe much more complex many-body interactions of the system at hand. The Hartree potential (Section 2.1.3.2) and v_{eff} in KS-DFT (Section 2.3) are examples for such approximative mean-fields[ix] – they reduce the many-body problem with its associated many-body equations to a system of coupled one-particle equations with independent particles. The "isolated" particles are treated as moving in the mean field, which in turn is constructed from the particle positions [77].

To resolve this issue, the method of a "**S**elf-**C**onsistent **F**ield (SCF)" is applied. According to Hartree, who introduced this concept into electronic structure calculation, the following definition is established: "*If the final field is the same as the initial field, the field will be called 'self-consistent',*[...]." [78] When applying SCF methods, this requirement is relaxed and the initial and final fields are assumed equal if the differences remain below a given threshold (prescribed by the desired accuracy).
In practice of KS-DFT, the "initial field" v_{eff} is derived from some starting estimate of the electron density ρ. Subsequently, this v_{eff} is used to calculate an updated density ρ which determines the "final field" of the given step. Usually, the densities rather than the effective potentials are compared to check whether self-consistency has been reached.

The detailed steps of an SCF cycle in the context of electronic structure simulations (schematically depicted in Figure 2.2) are:

[ix] In principle, DFT as laid out by the Hohenberg-Kohn theorems is **not** a mean-field approximation, as the relevant mean field is an **exact** one. However, approximations have to be made in KS-DFT, as the exact field is unknown.

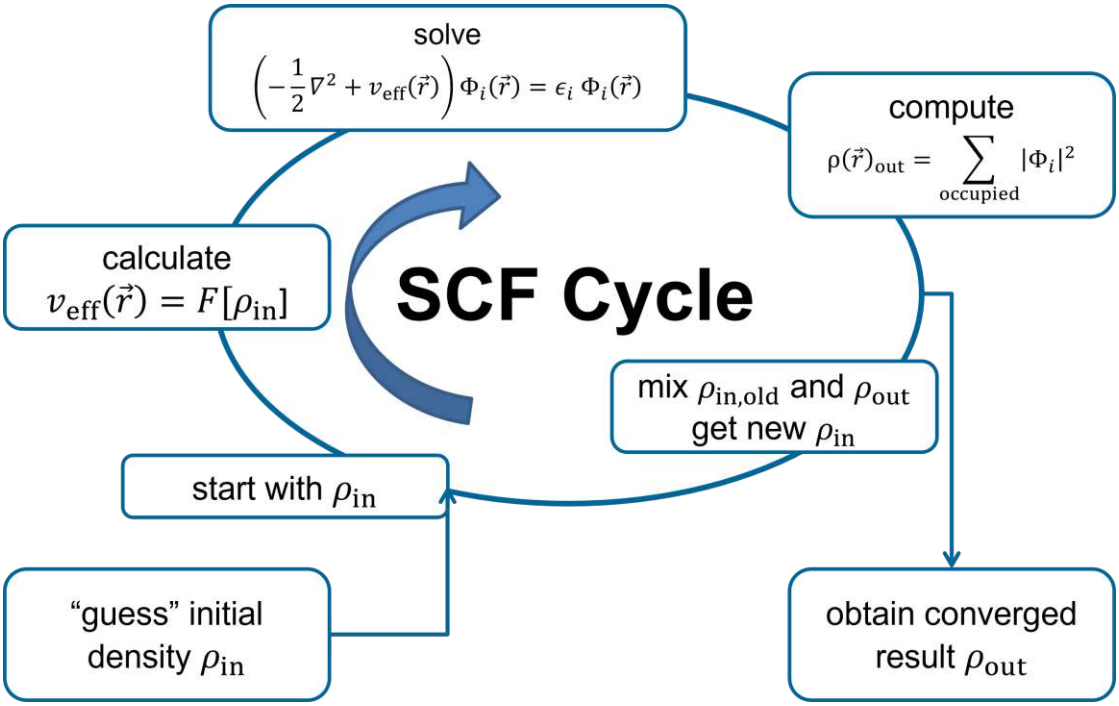

Figure 2.2: Schematic representation of the SCF cycle to solve KS equations. The starting input is an initial guess of the electron density ρ_0^{in} (the first element in an – ideally – converging series of ρ_j) that is needed to calculate the effective potential v_{eff}. Subsequently, the KS equations are solved to get new KS orbitals Φ_i, which, in turn, are used to compute the output density ρ_j^{out}. The two densities ρ_j^{in} and ρ_j^{out} are compared – if the difference is small enough to fulfil the desired accuracy (specified by the "convergence criteria") the SCF cycle is exited and a final converged ρ_{out} is obtained. Otherwise, the procedure is repeated with a new ρ_{j+1}^{in}, that is created by mixing ρ_j^{in} and ρ_j^{out}.

- ***Initial Guess of*** ρ_0^{in}:
  To start the SCF cycle, an initial guess of the input density ρ_0^{in} is needed. One such possible guess would be the assumption that – in a first approximation – the "combined" electronic density of a molecule or a solid is simply a superposition of its constituent atoms. In other words, ρ_0^{in} can be derived from the electron densities of the "free" atoms making up the material. This method is used in WIEN2k, as well as in many other solid-state DFT codes (e.g. VASP [79–81] or `exciting` [82]).
- ***Calculation of the Effective Potential*** v_{eff}:
  The effective potential v_{eff} is a functional of $\rho(\vec{r})$ – using the newly generated input density ρ_0^{in} (which is the first element in an – ideally – converging series ρ_j), v_{eff} can be calculated.
- ***Solution of KS Equations***:

Subsequently, this calculated effective potential v_{eff} is plugged into the KS equations, leading to a generalized eigenvalue problem that has to be solved to get KS orbitals Φ_i (cf. Section 2.5).

- ***Computation of*** ρ_j^{out}:
  In the next step, the output density ρ_j^{out} can be derived from the newly obtained KS orbitals Φ_i according to Equation 2.29.
- ***Comparison of*** ρ_j^{in} ***and*** ρ_j^{out}:
  The input density ρ_j^{in} (creating the "initial field") is compared with the resulting output density ρ_j^{out} (determining the "final field" of the iteration) – the difference between the two densities is evaluated with respect to a given desired accuracy. Depending on this, two different next steps are possible:
  - ***Creation of new Density*** ρ_{j+1}^{in}:
    In case of a too large difference between input and output densities, a new input density ρ_{j+1}^{in} is created by combining (or "mixing") the previous input density ρ_j^{in} and the corresponding output density ρ_j^{out}. A variety of algorithms – which are themselves still topic of current research – to create the new input density are in use: The foundation for the method used in WIEN2k (without simultaneous movement of atom positions) is laid out in reference [83]. An in-depth overview of mixing algorithms in general can be found in reference [77].
    This newly created input ρ_{j+1}^{in} is then used to run through the cycle again to get a new output density ρ_{j+1}^{out}.
  - ***Converged Result***:
    If the density difference stays below the desired limit, the cycle is exited and the converged density ρ^{out} is obtained after j iterations.

After running through j steps – which results in a series of output densities (ρ_0^{out}, ρ_1^{out}, ρ_2^{out},...,ρ_j^{out}) – the obtained converged density can then be used to determine the total ground-state energy (and derived properties).

CHAPTER 3

WIEN2k

It is you, who defines the number of k-points. If you want, just specify 1 k-point. But beware: your results may be complete nonsense !

Peter Blaha, WIEN2k Mailing List (2012-06-29)

The software package WIEN2k [7, 8] utilizes DFT to describe a wide range of crystalline solid materials on an ab-initio level, including:

- infinite ("perfect") bulk solids,
- solids with impurities or vacancies,
- surfaces,
- nanostructures,
- atoms or molecules...

As mentioned before, to run a simulation only a (rough estimate of the) structure of a material is needed without any further experimental input. WIEN2k then uses fundamental principles of quantum mechanics to calculate the total ground state energy of the system. This, in turn, can then be used to derive further materials properties, such as:

- atomic positions after structure optimization (i.e. minimization of atomic forces),
- phase transitions,
- electronic properties (band structure, **D**ensity **o**f **S**tates (DOS)),
- spectroscopic properties (IR or Raman shifts, **X**-ray **P**hotoelectron **S**pectroscopy (XPS), **X**-ray **A**bsorption **S**pectroscopy (XAS)),

- magnetism,
- **N**uclear **M**agnetic **R**esonance (NMR) and NMR Knight shifts,
- **S**canning **T**unneling **M**icroscopy (STM) and **A**tomic **F**orce **M**icroscopy (AFM) images of surfaces,
- optical properties...

3.1 History

WIEN2k has been developed and continuously improved for more than 30 years at the Institute of Materials Chemistry at the TU Wien[i]. Initially programmed to simulate only solids with rocksalt structure, the code was gradually extended to deal with all space groups.
The first public release under the name WIEN happened in 1990 [84], at which point cubic, tetragonal, orthorhombic, and hexagonal structures with inversion symmetry could be treated – all other space groups were added later on. The WIEN package received major new releases in 1993 and 1997 (as WIEN93 and WIEN97, respectively). The current name of the package has been in use since the fourth release as WIEN2k in 2001 [85, 86].
A fundamental change to the code happened in 2003, when the previous limitation of 99 inequivalent atoms per unit cell was extended to 999 atoms. Since then, the code has been updated regularly (at least once a year) with both minor improvements (bugfixes and introduction of convenience tools such as scripts for plotting) as well as major additions, including new packages (e.g. non-local van der Waals interactions [87], valence band photo electron spectroscopies (e.g. XPS) [88]...), new functionals (e.g. alternative exchange-correlation potentials like mBJ-TB [89] or meta-GGAs like SCAN [56]), or inclusion of new basis functions (e.g. **h**igh **e**nergy **L**ocal **O**rbitals (HELOs) and **h**igh **d**erivative **L**ocal **O**rbitals (HDLOs) [90]).

3.2 Expansion of Kohn-Sham-Orbitals

According to the definition of "ideal" crystals, bulk solid materials are treated with periodic boundary conditions in WIEN2k, i.e. a small unit cell (that contains all necessary information about the material) is repeated indefinitely in all three spatial dimensions[ii]. Therefore, the Bloch theorem can be applied to the wave functions of the crystalline material.

[i] As of May 2023, it is still maintained by Peter Blaha.
[ii] This complicates the treatment of non-3D materials – like surface slabs, nano materials, or molecules. For a use case of surface slabs see Chapter 6.

Theorem 5:
Bloch's Theorem

In a periodic crystal (i.e. a periodic potential), there exists a basis of eigenstates (called Bloch states) of the form [91]:

$$\psi(\vec{r}) = e^{i\vec{k}\vec{r}} u(\vec{r}) \,, \tag{3.1}$$

where $\vec{r}$ is a position vector and $\vec{k}$ is a wave vector in reciprocal lattice space. The exponential part $e^{i\vec{k}\vec{r}}$ is a plane wave that is modulated with the lattice periodic function $u(\vec{r})$. Lattice periodic means that

$$u(\vec{r}) = u\left(\vec{r} + \vec{n}\vec{a}\right) \,, \tag{3.2}$$

where $\vec{n}$ is any triple of integer numbers and the vector $\vec{a}$ contains the lattice constants of the crystal – i.e. $u(\vec{r})$ is invariant against any translation of a full lattice vector.

Seeing that Bloch states already contain plane waves, a reasonable approach might be to expand the KS orbitals into plane waves as well. However, this turns out to be computationally challenging. Close to the nuclei, the potential diverges at small values of r, the distance from the nucleus, and the wave functions vary strongly and even oscillate for higher principal quantum numbers; the variation of the wave functions becomes much slower at larger distances. As mentioned before, a plane wave basis cannot easily capture these oscillations (a large number of basis functions would be necessary, thus increasing computational cost); however, it is well adapted to represent the slow variations "in between atoms" with only a few basis functions.

3.2.1 Augmented Plane Waves (APW)

The **A**ugmented **P**lane **W**aves (APW) method was proposed by Slater in 1937 [92] to address these issues, and it builds the foundation of all basis sets used in WIEN2k.

The unit cell of a material is decomposed into two regions (shown in Figure 3.1):

1. ***Spheres around the atoms,*** S:
   Within the atomic spheres (defined via their "Muffin Tin (MT)" radii R_{MT}) the quickly varying wave functions are expanded into "atom-like" basis functions (consisting of a linear combination of products of radial functions and spherical harmonics). The amount of charge contained within an atomic sphere informs the categorization of states[iii]:

[iii] The fact that WIEN2k treats all electrons in the full potential (instead of, for instance, using a smooth pseudo-potential in the core regions) is the reason that it is called a "all electron" or "full-potential" code.

a) core states with very low energies have their charge contained completely,

b) semi-core states, while having low energies, still have some of their charge outside the sphere, and

c) valence states that have a considerable amount of charge outside the sphere and, therefore, relevant for chemical bonds.

2. ***Interstitial,*** I:
is the region outside the atomic spheres with slowly varying potential where plane wave basis functions can be efficiently used to expand the wave functions. The interstitial is the region of interest with respect to chemical bonds, as overlap between electronic states of neighboring atoms happens "far away" of nuclei.

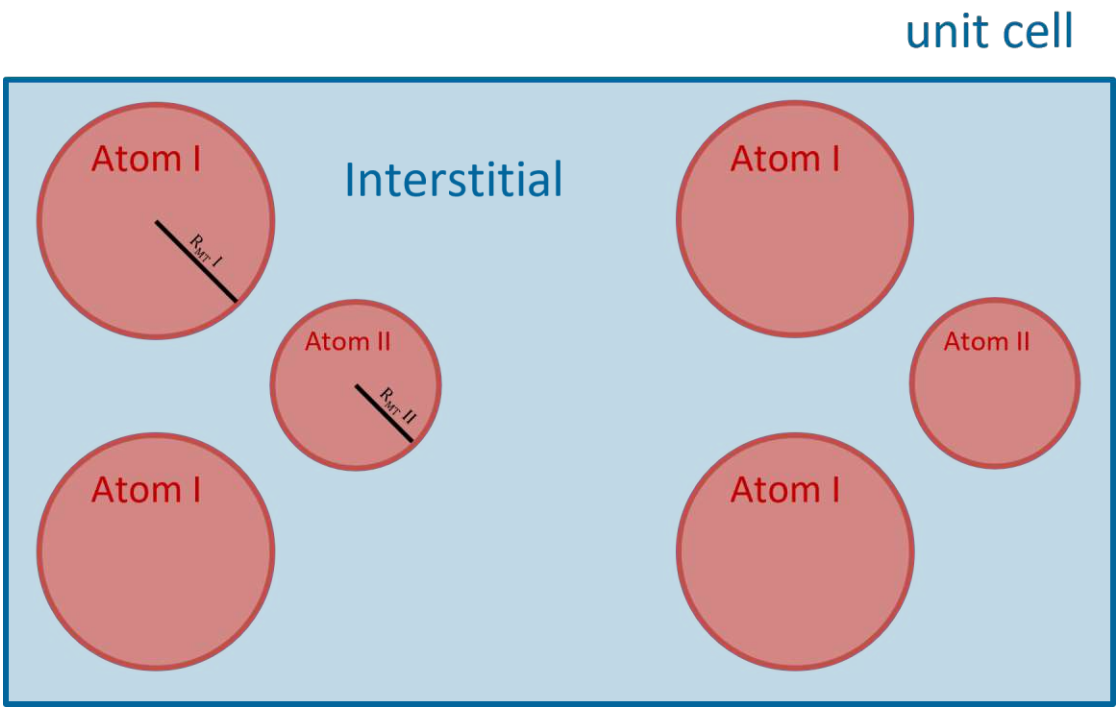

Figure 3.1: The unit cell of the simulated material is decomposed into two types of regions: spheres around the atoms and interstitial in between the atoms. Different basis functions are used in the different regions. The radii $R_{\mathrm{MT}}(i)$ define the size of each atomic sphere and are input parameters of a DFT calculation with WIEN2k.

Every plane wave is augmented with an atom-like function (hence the name of the method). The respective basis functions for the two regions are then given by:

$$\phi^{\mathrm{APW}}_{\vec{k},\vec{K}} = \begin{cases} \frac{1}{\sqrt{V}} e^{i(\vec{k}+\vec{K})\vec{r}} & \vec{r} \in I \\ \sum\limits_{l,m} A^{\vec{k}+\vec{K}}_{l,m} u_l(r,E) Y_{l,m}(\hat{r}) & \vec{r} \in S \end{cases} \tag{3.3}$$

Here, $\vec{k}$ is the wave vector restricted to the first Brillouin zone, $\vec{K}$ are reciprocal lattice vectors and V is the volume of the unit cell. The number of used plane waves is determined by a largest value $K_{\max}$ and the possible $\vec{k}$ vectors are sampled on a reciprocal k-mesh. $u(r, E)$ is the solution of the radial Schrödinger equation for energy E depending on the distance r from the nucleus (obtained numerically on a logarithmic radial grid inside the sphere), $Y_{\mathrm{lm}}(\hat{r})$ are spherical harmonics for the quantum numbers l and m, and the coefficients $A_{\mathrm{l,m}}^{\vec{k}+\vec{K}}$ are chosen such that the atom-like function and the corresponding plane wave match in value at the sphere boundary.

Nowadays, the APW basis set alone (see below for the alternative APW+lo) is de facto not used at all. This is due to two issues that arise during computation: First, the secular equation of APW is a non-linear function of the energy, which has to be evaluated for its zeroes (as the energy parameter E in Eq. 3.3 is required to match the eigenstate energy ϵ) at every point of the k-mesh. Second, singularities in the secular equation appear, when a node of $u_l(r, E)$ coincides with a sphere boundary. Both of these problems make the APW method computationally very demanding even for moderately sized systems. [84]

3.2.2 Linearized Augmented Plane Waves (LAPW)

To address the computational challenges of APW, Andersen suggested a linearization of energy dependency of the basis functions inside the atomic spheres that additionally use the first derivative of the radial solution of the Schrödinger equation $\dot{u}(r, E_l) = \frac{\partial u(r,E)}{\partial E}|_{E_l}$ [93, 94]. In this **L**inearized **A**ugmented **P**lane **W**aves (LAPW) method, ϕ^{LAPW} in the interstitial region remains the same, but the basis function within atomic spheres no longer depend on the energy: Instead of requiring that the energy parameter E_l matches the corresponding KS eigenvalue, it remains fixed to an expected value (roughly in the middle of the occupied band) for a given value of l that can be pre-determined relatively cheaply. The LAPW basis functions are then given by:

$$\phi_{\vec{k},\vec{K}}^{\mathrm{LAPW}} = \begin{cases} \frac{1}{\sqrt{V}} e^{i(\vec{k}+\vec{K})\vec{r}} & \vec{r} \in I \\ \sum\limits_{l,m} \left[A_{l,m}^{\vec{k}+\vec{K}} u_l(r, E_l) + B_{l,m}^{\vec{k}+\vec{K}} \dot{u}_l(r, E_l) \right] Y_{l,m}(\hat{r}) & \vec{r} \in S \end{cases} \tag{3.4}$$

The coefficients $A_{l,m}^{\vec{k}+\vec{K}}$ and $B_{l,m}^{\vec{k}+\vec{K}}$ are again determined via matching to the corresponding plane wave; this time, however, a second boundary condition is needed, thus both value and slope have to match.

The basis set derived from this method is already expected to describe electrons in a solid well. However, to improve errors introduced by the linearization and add even more flexibility – especially in cases with semi-core states – additional $\vec{k}$-independent functions called "**L**ocal **O**rbital (LO)" can be introduced [95]. These LOs contain two energy parameters $E_{1,l}$ and $E_{2,l}$, thus making it possible to treat two states with same l- but different n-values (e.g. 3s and 4s). This might be necessary, if there is a valence electron for the higher principal quantum number, while the charge of the lower n state (of the

same l-value) is not completely contained in the sphere). The LO is completely contained inside the sphere (hence "local") and 0 in the interstitial (thus adding no additional plane wave to the basis set); in the spheres it is given as:

$$\phi_{lm}^{LO} = \left[A_{l,m}u_l(r,E_{1,l}) + B_{l,m}\dot{u}_l(r,E_{1,l}) + C_{l,m}u_l(r,E_{2,l})\right]Y_{l,m}(\hat{r})\,. \tag{3.5}$$

The necessary boundary conditions in this case (to find the coefficients $A_{l,m}$, $B_{l,m}$, and $C_{l,m}$) are that the function goes to zero (and has a zero slope) at the sphere boundary and has to be normalized within the sphere.

3.2.3 Augmented Plane Waves and Local Orbitals (APW+lo)

Sjöstedt et al. showed that the linearization method of LAPW is not the most efficient and can be improved upon by using almost the basis functions in the spheres as in the original APW method (i.e. with only a radial function u). The difference is that in this alternative to APW the energy is fixed in the same manner as with LAPW – i.e. a pre-determined E_l will be used as well. [96]

$$\phi_{\vec{k},\vec{K}}^{\text{APW+lo}} = \begin{cases} \frac{1}{\sqrt{V}}e^{i(\vec{k}+\vec{K})\vec{r}} & \vec{r}\in I \\ \sum\limits_{l,m} A_{l,m}^{\vec{k}+\vec{K}}u_l(r,E_l)Y_{l,m}(\hat{r}) & \vec{r}\in S \end{cases} \tag{3.6}$$

To introduce enough flexibility in the radial functions, a new **l**ocal **o**rbital (lo) (note the lower cases to distinguish this lo from the LAPW method) is added [96]:

$$\phi_{l,m}^{lo} = \left[A_{l,m}u_l(r,E_l) + B_{l,m}\dot{u}(r,E_l)\right]Y_{l,m}(\hat{r}) \tag{3.7}$$

It has been shown that these alternative method converges to the same results as the LAPW method, but does so much faster, as it is possible to reduce the basis set size up to 50 % [97].

3.2.4 Higher Derivative Local Orbitals (HDLO)

Recently, Karsai et al. implemented the possibility to include HDLOs to certain states to increase the flexibility of the used basis set even further [90]. The use of HDLOs has been suggested before (e.g. in Ref. [98]) to reduce linearization errors and improve basis set convergence. The general idea is (again) to introduce additional basis functions (entirely contained within the atomic spheres) that make use of the second energy derivative of $u_l(r,E_l)$:

$$\phi_{l,m}^{\text{HDLO}} = \begin{cases} \left[A_{l,m}^{\text{HDLO}}u_l(r,E_l) + C_{l,m}^{\text{HDLO}}\ddot{u}(r,E_l)\right]Y_{l,m}(\hat{r}) & \vec{r}\in S \\ 0 & \vec{r}\in I \end{cases} \tag{3.8}$$

Tests by Karsai et al. [90] reveal that using HDLOs in case of d- and f-block elements greatly reduces linearization errors (especially in connection with large R_{MT} values above

2.5 bohr). Moreover, they showed that HDLOs remove the R_{MT}-dependence in many cases, thus facilitating accurate results even with large R_{MT} values. Larger sphere radii are advantageous (provided the results are accurate enough) for two reasons: (i) Core leakage (i.e. charge of core electrons not contained in the spheres) gets reduced, which might be a problem for elements late in the periodic table (e.g. f-block elements). And, (ii), calculations grow faster, as a smaller plane wave cutoff K_{max} can be used.

3.3 Flow of Calculations

WIEN2k as a software package consists of a large number of independent programs that perform different tasks: the necessary input parameters and calculation specifics are provided via dedicated input files (each program reads its own file), and the results are written out in corresponding output files.
The execution of these programs can be directly controlled from the command line via shell scripts: The script `x_lapw` enables manual execution of a single program, while `run_lapw` and `runsp_lapw` start SCF cycles without or with spin-polarization (i.e. without or with magnetism taken into account), respectively.
Alternatively, a web-based graphical user interface (`w2web`) can be used as well.

The general flow of a WIEN2k calculation is depicted schematically in Figure 3.2.

This general scheme only shows the programs that **have to** be run at least once during a "standard" WIEN2k SCF cycle (in case of spin-polarization, the programs `LAPW1`, `LAPW2`, and `LCORE` are run twice per iteration – once for spin-up electrons and once for spin-down electrons). Further programs can be switched on as needed, for instance, to add orbital-dependent potentials with the program `ORB` (e.g. in LDA+U calculations [65, 99, 100]), to add spin-orbit coupling with `LAPWSO`, or add corrections for long-range dispersion by using pair-wise atomic dispersion coefficients (`DFT-D3` [66, 101] or `DFT-D4` [102–104]) or using non-local van der Waals functionals (`NLVDW` [67, 68, 87]).

3.3.1 Core Programs

The main (or "core") programs of WIEN2k, which are executed at least once in **every** calculation, consist of the programs depicted in the scheme shown in Figure 3.2. They fall into two different groups:

- Initialization: `NN`, `SGROUP`, `SYMMETRY`, `LSTART`, `KGEN`, and `DSTART`.
- SCF cycle (cf. Section 2.6): `LAPW0`, `LAPW1`, `LAPW2`, `LCORE`, and `MIXER`.

During initialization, the input structure (which is provided via the `struct`-file, see below in Section 3.3.2) is passed through the programs `NN`, `SGROUP`, and `SYMMETRY`: `NN` calculates next neighbour distances and checks for sphere overlaps (which must not occur). `SGROUP` finds the space group of the material at hand as well as point groups of the inequivalent atom sites. `symmetry` generates the symmetry operation matrices of

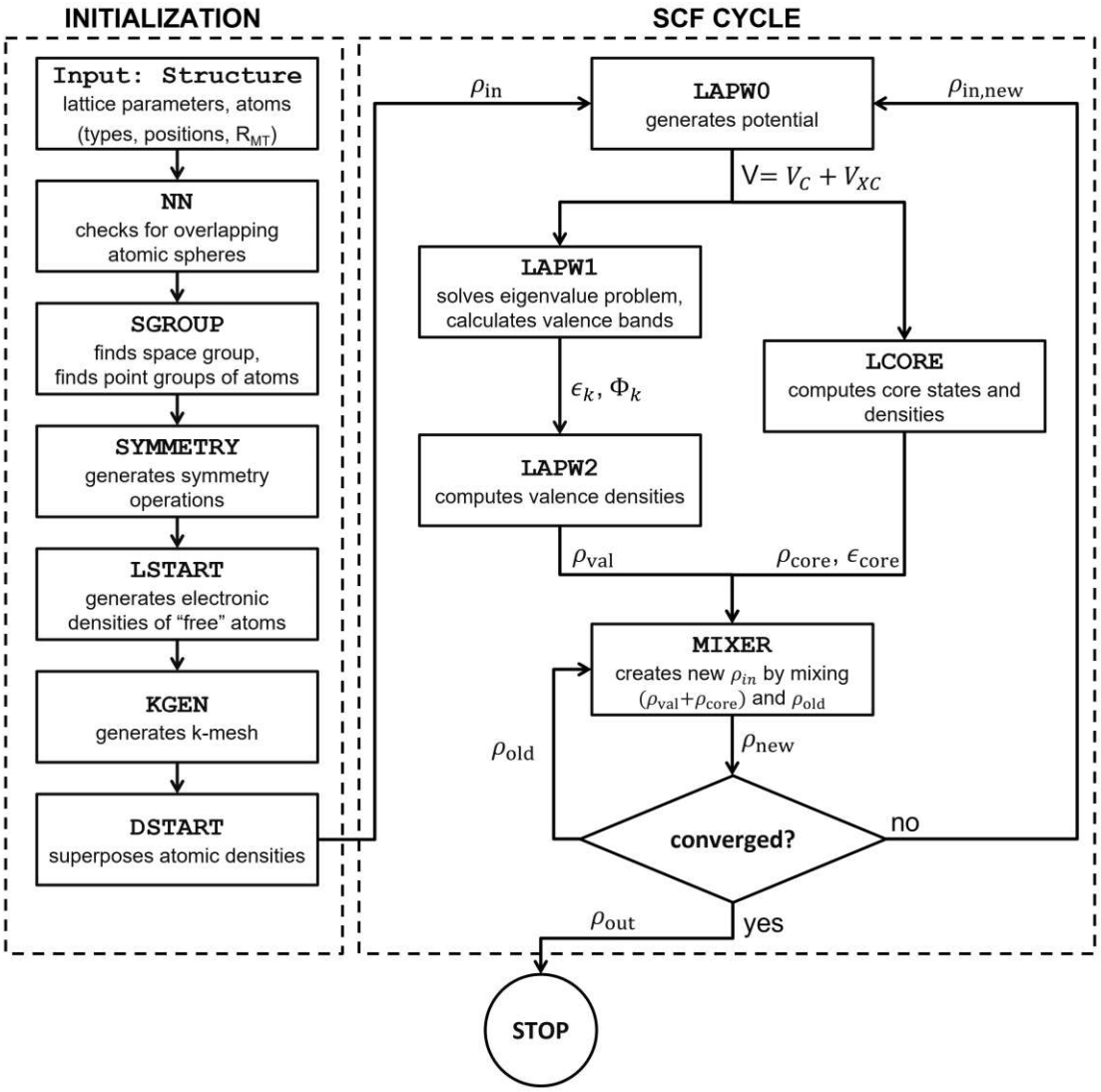

Figure 3.2: Flowchart of a general WIEN2k calculation consisting of initialization and SCF cycle: The input structure is passed through the programs `NN`, `SGROUP`, and `SYMMETRY`, which update the `struct`-file with additional symmetric information. `KGEN` sets up the k-mesh, and `LSTART` generates atomic densities that are superposed by `DSTART` to obtain an initial ρ_{in}. This initial density is used as starting point for the SCF cycle, in which the programs `LAPW0`, `LAPW1`, `LAPW2`, `CORE`, and `MIXER` are cycled until convergence of ρ is reached. Scheme reproduced from [7].

the respective space groups. All three of those programs add necessary information to the `struct`-file, thus completing it.
Subsequently, `LSTART` generates electronic densities of the "free" atoms (i.e. the electronic densities the atoms would have if they would not be part of a solid). `LSTART` is a modified version of a relativistic DFT code programmed by Desclaux in the 1960s (the non-relativistic version was published in reference [105] and reference [106] contains details of the physics used in the relativistic version).
The atomic densities generated by `LSTART` are then superposed by `DSTART` to obtain a density ρ_{in} which is used as starting point for the SCF cycle.
The last ingredient needed is the k-mesh, which is a special point grid that is used to sample the unit cell in reciprocal space. This mesh is provided by `KGEN`.

After an SCF cycle is started, the five main programs are cycled until a converged electronic density is obtained:
The potential V_{tot} (a sum of the Coloumb potential V_{Coul} and the exchange-correlation potential V_{xc}) is calculated from the electronic density with the program `LAPW0`.
This potential is used to compute the eigenvalues for core (`LCORE`, which is based on the same DFT code as `LSTART` [105, 106]), semi-core[iv], and valence states (`LAPW1`), respectively. `LCORE` generates the electronic density contribution (ρ_{core}) of the core states directly, while `LAPW1` passes the eigenvalues ϵ_k and eigenvectors Φ_k to `LAPW2`, where the valence density ρ_{val} is generated.
In the final step of an SCF cycle, `MIXER` combines ("mixes") the newly calculated densities ρ_{val} and ρ_{core} with the old input density ρ_{old} to obtain a new density ρ_{new}. These densities are compared to ascertain if the cycle can be stopped and a converged density was found or if another iteration is necessary.

After the SCF cycle has ended with a converged electronic density ρ, a third – much larger – group of programs can be used to analyse the results or calculate further materials properties. Commonly used examples include:

- `TETRA`, which calculates total or partial densities of states,
- `SPAGHETTI`, which is used to generate electronic band structures a long defined k-paths,
- `LAPW5`, which generates the electronic density in a 2D cut through the unit cell for plotting,
- `3DDENS` (contributed by the author as part of this dissertation), with which the electronic density of the whole unit cell can be generated on a 3D grid[v],
- `NMR`, which calculates NMR shifts,
- `OPTIC` for the calculation of optical properties...

[iv] A semi-core state is a state at a comparatively low energy that has charge that is not completely contained inside the atomic sphere. These states sometimes lead to complications during calculations; to properly treat theses states, additional basis functions are necessary, cf. Section 3.2.

[v] Moreover, `3DDENS` allows the simulation of constant current STM images.

3.3.2 The `case.struct`-File as Master Input

As mentioned before, the only input necessary to start a calculation is the structure of the material. This is provided via the `case.struct`[vi] file that contains information about:

1. ***the unit cell***,
   such as lattice parameters a, b, and c (given in bohr) as well as the lattice angles α, β, and γ, which span the unit cell, and information about the lattice type (i.e. the centering of the unit cell).

2. ***the atoms in the unit cell***,
   including the total number of inequivalent atoms in the cell, their positions (as positive fractions and multiplicities (i.e. the number of equivalent atoms – the positions of which need to be given as well), and the atom types (given by the atomic charge number).
   The radii of the respective atomic spheres R_{MT} should be specified as well, as the internal default is usually not optimal.

3. ***parameters concerning the calculation***,
   including, for instance, symmetry related data (space group, matrix representations of the symmetry operations – usually automatically set during initialization) or the number of points on the mesh used to numerically solve the radial Schrödinger equation within the atomic spheres as well as the distance of the first mesh point from the respective atomic center (usually "standardized" values are used here).

Only the data about the unit cell and the atoms have to be provided by the user.

3.3.3 Input/Output in WIEN2k

Input/**O**utput (IO) handling in WIEN2k is set up in a way that every program reads in program specific computational parameters from dedicated input files: For instance, `LAPW0` gets the chosen functional from `case.in0` or `LAPW1` reads in e.g. energy parameters and basis set size necessary for the eigensolving step from `case.in1`. These files can be modified to change computational parameters.
Shared IO is read from/written to files that are accessed by multiple programs. The following example nicely illustrates the way that such shared IO is handled in WIEN2k: The potential calculated by `LAPW0` is written to two files (`case.vsp` and `case.vns`) which are then read in by `LAPW1` in order to calculate the eigenvalues and -vectors. These values are in turn written to `case.energy` and `case.vector` by `LAPW1` and, subsequently, used by `LAPW2` to calculate the electronic valence density ρ_{val} (stored in

[vi] The naming scheme of WIEN2k requires that all "active" files, for instance used during an SCF cycle, are called the same as the working directory of the calculation. As stand-in for this name, the general "`case`" is used from now on.

`case.clmval`).
While this IO handling has the clear drawback of needing a substantial amount of file accesses, two advantages are apparent as well:

- The memory requirements are reduced, as it is, for instance, not necessary to keep all eigenvalues and -vectors in memory when the SCF cycle progresses from solving the eigenvalue problem (solved by `LAPW1`) to calculating the electronic density (done by `LAPW2`).
- A crashed calculation can be restarted from the last "save point", as in most cases WIEN2k programs do not overwrite their own input (provided the crash was caused by a hardware or system problem – e.g. a network failure – and not a user-related error).

3.4 Solving the Generalized Eigenvalue Problem

As described in Section 2.5, expanding the KS-orbitals into any basis set and applying the variational principal leads to a generalized eigenvalue problem (Eq. 2.46) that can be solved algebraically. The setup of the matrices H and S and the subsequent solution of the problem (both are performed within the program `LAPW1`) are routinely the most expensive steps in an SCF cycle – taking up to 90% of the total time (depending on the problem size).

$$HC = ESC\,, \tag{2.46 revisited}$$

with

$$H_{m,n} = \int \phi_m^* \hat{\mathcal{H}} \phi_n d\vec{r} \tag{2.46a revisited}$$

and

$$S_{m,n} = \int \phi_m^* \phi_n d\vec{r}\,. \tag{2.46b revisited}$$

Therefore, it is crucial that the computations involved in these steps are performed as efficiently as possible. WIEN2k uses algebraic libraries (i.e. collections of optimized and constantly improved algorithms) to perform the necessary matrix operations and calculations.
The following open-source libraries are used in the eigensolvers of WIEN2k:

- **B**asic **L**inear **A**lgebra **S**ubprograms (BLAS) [107] and
- **L**inear **A**lgebra **PACK**age (LAPACK) [108] for local single-processor computations (also called "sequential").

- **P**arallel **BLAS** (PBLAS) [109],
- **Sca**lable **LAPACK** (ScaLAPACK) [110], and

- **E**igenvalue so**L**vers for **P**etaflop-**A**pplications (ELPA) [111, 112] for parallel multi-processor computation (both with distributed or shared memory)[vii].

Moreover, WIEN2k can make use of the proprietary Intel® libraries **M**ath **K**ernel **L**ibrary (MKL) and **V**ector **M**ath **L**ibrary (VML), which are now part of the open programming model openAPI[viii].

The following gives a brief overview of the two eigensolvers implemented in WIEN2k. Both are entirely based on the routines and algorithms contained in the libraries listed above. Note, that only the (main) sequential algorithms are mentioned (i.e. BLAS and LAPACK routines). With one exception, the analogous routines of PBLAS and ScaLAPACK are used instead in case of parallel computations (see also Section 3.5).

3.4.1 Set-Up of H and S

In every SCF iteration, the first step during the solution of the respective eigenvalue problem is the setup of the Hamilton matrix H and the overlap matrix S according to Equations 2.46a and 2.46b. H and S are both either symmetric (i.e. $h_{i,j} = h_{j,i}$ and $s_{i,j} = s_{j,i}$), if the modelled unit cell contains an inversion center, or Hermitian (i.e. $h_{i,j} = h^*_{j,i}$ and $s_{i,j} = s^*_{j,i}$), if no inversion symmetry is present.
The setup of these matrices is a two step procedure, during which the `LAPW1` subroutine `HAMILT` computes the spherical matrix elements H_{sp} and S, and subroutine `HNS` sets up the non-spherical matrix elements H_{nsp}.
Both programs make use of BLAS routines (mostly `dgemm` and `zhemm`) and, if available, Intel® VML cos-functions (`vdcos` and `vzcis`). This is also true for parallel computations – as no other routines are used – the necessary communication across processors was implemented "manually" using MPI (see Section 3.5.1).

3.4.2 Standard ("Full") Diagonalization

The default diagonalization scheme is used to solve for the eigenpairs (i.e. eigenvalues and associated eigenvectors) of the general eigenvalue problem. It is also called "full diagonalization", as it calculates the eigenvalues exactly (for the given SCF iteration) and uses the whole (full) matrices to do so.

The general scheme of the full diagonalization scheme is shown in Algorithm 3.1.

The used library algorithms for the main steps are listed below:

- Finding the Cholesky factorization of S (`dpotrf`/`zpotrf`). This is required for the next step.
- Reducing the general eigenvalue problem (Eq. 2.46) to standard form $H'C' = EC'$ (`dsygst`/`zhegst`).

[vii] ELPA was implemented as part of this thesis.
[viii] `https://www.oneapi.io/`

Algorithm 3.1: Pseudo-code representation of a "full" diagonalization during the eigensolving step of `LAPW1`.

Input: read matrices H and S that were set up previously
perform a Cholesky factorization on S;
transform the general eigenvalue problem into standard form;
compute lowest eigenvalues up to a cut-off;
transform eigenvectors C' back to original problem;
Output: write eigenvalues to `case.energy`-file
Output: write eigenvectors to `case.vector`-file

- Computing the eigenvalues. Routinely in WIEN2k simulations, only eigenvalues below a specified energy cut-off (up to 10% of all eigenvalues) are computed. Therefore, a LAPACK algorithm based on inverse iteration [113] (called "expert driver") is used (`dsyevx/zheevx`)[ix]. For applications that require all eigenvalues (e.g. the simulation of NMR shifts), two alternative routines have been implemented during this thesis (cf. Section 4.2.1).
  This step is the exception mentioned above regarding parallel computation: During parallel calculations, `pdsyevr` and `pzheevr` – algorithms based on relatively robust representations [115] – are called instead of the parallel analogous of the algorithm used in sequential computations.
- The resulting eigenvectors in C' are transformed back into the original problem (`dtrsm/ztrsm`).

3.4.3 Iterative Diagonalization

A second scheme (called "iterative diagonalization") has been implemented [116] as alternative for larger systems that need only "few" eigenvalues relative to the dimension of the associated matrices.
This method makes use of the fact that during an SCF cycle the KS equations are solved iteratively. This means, in short, that it is not necessary to solve for the exact eigenvalues during each iteration. Instead, approximate solutions[x] are calculated using the eigenvectors of the previous iteration as the respective starting point. This works based on the (sensible) assumption that the change of the eigenvectors going from one SCF iteration to the next is only small.

The general scheme of the iterative scheme is shown in Algorithm 3.2.

The four main steps are (more details can be found in [116]):

[ix] In case of sequential computation and symmetric matrices, which occur in materials that exhibit inversion symmetry, a modified version of `dsyevx` by Kvasnicka et al. [114] is used.

[x] As long as the approximations converge as the SCF cycle, this approximative approach is satisfactory.

Algorithm 3.2: Pseudo-code representation of a "iterative" diagonalization during the eigensolving step of `LAPW1`. Note that the first step of an SCF cycle using iterative diagonalization has to be done within the "full" diagonalization scheme, as eigenvectors are needed.

```
Input: read matrices H and S that were set up previously
Input: read eigenvectors C from the previous step
generate search space Z:
if search space Z is calculated on the fly then
    perform LU-factorization of H;
    solve resulting system of linear equations
else if inverse H_0^{-1} will be stored on disk then
    if first iteration within the iterative diagonalization scheme then
        calculate the inverse H_0^{-1};
        Output: write inverse H_0^{-1} to case.storeHinv
        calculate Z via matrix-matrix multiplication
    else
        Input: read inverse H_0^{-1} from case.storeHinv
        calculate Z via matrix-matrix multiplication
    end if
end if
set up an eigenvalue problem with reduced dimension;
solve the eigenvalue problem of reduced dimension "fully";
Output: write eigenvalues to case.energy-file
Output: write eigenvectors to case.vector-file
```

- Finding starting estimates for the eigenvalues – these estimates are based on the eigenvectors of the previous SCF iteration:
  The vectors (stored in a matrix) are multiplied with the Hamilton matrix H and the overlap matrix S, respectively. To set up the estimated eigenvectors, these products are used. Only Level-3 BLAS routines (`dsymm/zhemm` and `dsyr2k/zher2k`, respectively) are used.
- Calculation of the search space in one of two different ways:
  1. The search space can be calculated "on the fly" by performing a LU-factorization of the H (`dgetrf/zgetrf`) followed by the solution of the corresponding system of linear equations (`dgetrs/zgetrs`).
  2. More efficiently, it is possible to calculate and store the inverse of H once in during first step, which is then read in all subsequent iterations. The search space is then generated by simple matrix-matrix multiplication. However, this procedure requires sufficient disk space and causes additional IO operations.
- Setting up an eigenvalue problem of reduced dimension (two times the number of looked-for eigenvalues, usually about 20% of the original dimension). Again, only

the same level-3 BLAS routines as above are used.

- Finding the solution of the reduced eigenvalue problem – this is done as described in the previous section.

Another key difference to the full diagonalization scheme (aside from faster computation in "standard" cases) lies in the dependence on the number of eigenvalues: It is small for the time spent in the full scheme, while the iterative diagonalization exhibits linear dependence [116]. That means that in cases in which more (or possibly all) eigenvalues are needed, the full scheme might be faster again.

3.5 Parallelization in WIEN2k

3.5.1 Parallel Computing

Parallel computing as well as **H**igh-**P**erformance **C**omputing (HPC) are terms in computer science that are somewhat self-explanatory and do not have rigorous definitions at the same time. Parallel computing simply relates to computations during which multiple calculations or processes are performed; the main motivation usually is achieving faster execution times. Parallel computations can be performed on any hardware with more than one core, which nowadays includes common laptops. On the other hand, HPC utilizes supercomputers or computer clusters for parallel computations on a much larger scale, i.e. computer systems with up to hundreds of thousands of **C**entral **P**rocessing **U**nits (CPUs), combined memory up to hundreds of TB (if not PB), specialized file systems (to handle data access and storage), and sophisticated interconnects to facilitate communication between the CPUs. [117]

Here, only two important types of parallelism in parallel computing – namely "data parallelism" and "task parallelism" – shall be discussed in brief, as both are relevant for parallelism in WIEN2k. A few other notions important to (parallel) computing (e.g. speedup) will be introduced were they are needed in Chapter 4. Further details and in-depth discussions about parallel computing and HPC are readily available in textbooks (for instance [117–119]).

3.5.1.1 Data Parallelism

Data parallelism uses multiple processors to work on different subsets of the same data: The data (for instance a matrix that has to be diagonalized) is split up, distributed across multiple processors (which can be part of the same CPU but can also be on a different node, provided communication between nodes can be done efficiently), and finally processed. Two widespread computing models used to do this are **O**pen **M**ulti-**P**rocessing (OpenMP) [120] and **M**essage **P**assing **I**nterface (MPI) [121][xi], that are aptly described

[xi] While OpenMP itself is an application programming interface, MPI is just a standard for message passing – hence the name. There are various implementations of MPI, for instance from Intel (used to

as “shared-memory” and “distributed-memory” multiprocessing. In a nutshell, the key differences are:

- OpenMP (as shared-memory model) runs on a single node without any requirements with respect to network capabilities – the available memory must, of course, be sufficiently large. For MPI parallel jobs this is the other way around: They can run across different hosts (e.g. a computer cluster) – hence distributed memory – and, therefore, need network connections to handle the communication. The memory requirements are much lower, as the data is split up.
- There is no direct communication in OpenMP, data exchange is handled via access to the same memory space. In MPI computations, on the other hand, data is exchanged via explicit messages between two processes.
- When executed, a OpenMP program runs as single instance during which multiple so-called “threads” are spawned that perform a given instruction (for instance, a loop over matrix elements) concurrently. These threads can be active for the whole runtime or be joined again after a parallel task has been completed (this is called the “fork-join model”). MPI programs are run with multiple instances: Every participating process runs the entire program – the difference between the instances (e.g. the specific communication pathways or the data subset) are determined hierarchically (each instance is assigned a “rank”).
- OpenMP programs can be started directly by the user, while MPI programs have to be executed via a specific command (usually called “`mpirun`” that is part of every MPI implementation): This command launches all necessary instances of the MPI-parallel program, assigns the ranks, and sets up communication.
- Coding errors can cause wrong results due to parallelization in case of OpenMP (even if the actual computations are all correct), as “race conditions” might occur. Read and write accesses to the shared data must be handled in a way that ensures that stored values are not changed out of order: Input values must be read correctly by all threads that need them before they get overwritten; values that are updated during iterations must not be read for the start of the next step before they have been updated, and so on. Otherwise, the results will change when repeatedly running the program (depending on the speed of the respective threads). This does not happen with MPI programs; errors in the parallelization simply lead to “deadlocks” – program freezes – that are, for instance, caused by all instances waiting for messages at the same time.

3.5.1.2 Task Parallelism

In task parallel computation, processors either perform the same task on different data sets (this is done in WIEN2k’s k-point parallelization, see below) or different tasks

be commercial, no part of the openAPI programming model), MPICH (`www.mpich.org`), or openMPI (`www.open-mpi.org`).

altogether. For instance, automated WIEN2k calculations (during which a list of solids is simulated in parallel) could be considered task parallelism: Every processor runs a different solid until the calculation has finished and then starts another calculation on the next solid from the list. During the work for this thesis, this was done during the calculation of a large number of benchmark cases (see Chapter 7).

3.5.2 Parallelization of WIEN2k

All of the main programs of WIEN2k have been parallelized (provided the time consumption of the respective program is large enough for parallelization to make sense). The program offers three levels of parallelism that can be employed to speed up calculations. [7, 8]

Depending on the size of the problem (mostly determined by the number of atoms in a unit cell) and the available hardware, one (or a combination) of the following parallelization schemes can be utilized – the first two of which make use of data parallelism, while the last one serves as a prototypical example for task parallelism:

- ***Shared memory processing (OpenMP)***:
  All programs of WIEN2k that use libraries (like BLAS and LAPACK[xii]) – for instance, most of the core programs mentioned in section 3.3.1 – can make use of shared memory parallelization via the usage of the OpenMP versions of the respective libraries.
  This form of parallelization is controlled by environment variables of the operating system (`OMP_NUM_THREADS` which specifies how many threads should be used in threaded parts of any given program). Alternatively, WIEN2k specific parameters can be chosen in the `.machines` file (see example at the end of this chapter).

- ***Distributed memory processing (MPI)***:
  MPI-parallel versions of most of the core programs exist (`DSTART`, `LAPW0`, `LAPW1`, and `LAPW2`). In fact, parallelization is necessary for large systems (more than 50 atoms per unit cell), as the associated memory requirements grow beyond single-system memory sizes. For MPI parallelization to be effective, the interconnect between the parts of the network must be strong enough to facilitate fast communication (the small cluster of the WIEN2k group uses an Infiniband network with transfer rates of 40 GB/s). Moreover, sufficiently large number of cores are needed for MPI to be efficient (the specifics depend on the problem sizes and the hardware).

  MPI parallelization in WIEN2k works on two levels: (i) Some programs are parallelized over atoms or basis functions (e.g. `LAPW0` or `DSTART`), which yields excellent speedups, as hardly any communication (aside from the distribution)

[xii] Aside from the algebraic libraries heavily used in WIEN2k, FFTW [122] is also used when Fourier transforms are needed, e.g. in `LAPW0` or `3DDENS`.

is necessary. (ii) Other programs (e.g. `LAPW0` or `LAPW1`) make use of parallel versions of libraries (ScaLAPACK, ELPA, **F**astest **F**ourier **T**ransform in the **W**est (FFTW)). This type of MPI parallelism was benchmarked as part of this thesis (cf. Section 4.1).

- ***k-point parallelization***:
  WIEN2k programs that operate on a k-mesh (e.g. `LAPW1`, `LAPW2`, `NMR`, or `OPTIC`, can be executed in k-parallel. That means, that the k-point list is split up and multiple processors work on their own subset of k-points. This is then followed by a summation step that combines the partial results.

  While OpenMP and MPI parallelization will be discussed in more detail in Chapter 4, k-point parallelization was not a major focus of this thesis; therefore, a simple performance test was conducted to illustrate this scheme.

  The program `LAPW1` was used on a small test case ($LiBH_4$) with computational parameters chosen such that the matrices of the associated eigenvalue problem have dimensions around 9300. A series of test-runs was done, where the same steps were performed for up to 6 k-points at the same time[xiii]. The recorded execution times are summarized in Table 3.1 and visualized in Figure 3.3.

Table 3.1: Results of a performance test of k-point parallelization in WIEN2k: `LAPW1` was run on a small test case ($LiBH_4$) with matrix dimensions of around 9300, performing the same steps for up to six k-points in parallel.

	Execution Time (in s) of Processor					
	#1	**#2**	**#3**	**#4**	**#5**	**#6**
1 k-point	140	—	—	—	—	—
2 k-points	142	139	—	—	—	—
3 k-points	144	140	144	—	—	—
4 k-points	147	146	146	139	—	—
5 k-points	154	152	151	147	148	—
6 k-points	159	157	159	158	158	158

The test does not reveal any significant performance drops due to the concurrent execution of multiple k-points up to three parallel k-points on the given hardware. The slight deviations are likely a combination of random fluctuations and minute differences in the size of the respective matrices – which vary in the order of 0.5 % in this case. That means a list of k-points could be tackled using up to three processors (i.e. splitting the list and sharing the k-points between multiple processors). Due

[xiii] For a system this small in terms of atoms per unit cell, 6 k-points are not enough to expect physical results; however, the actual results are not of interest here.

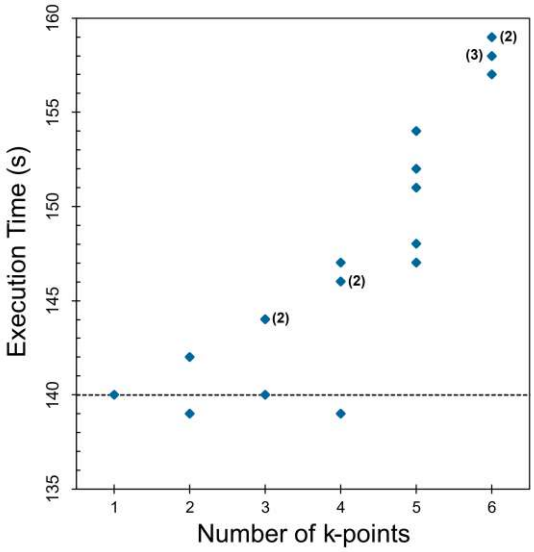

Figure 3.3: Performance test of k-parallelism in WIEN2k. Shown is the execution time of the program `LAPW1` on a hexacore CPU when solving the eigenvalue problem for a small test case ($LiBH_4$) with matrix dimensions of around 9300 (slightly varying from k-point to k-point) with up to 6 k-points solved in parallel on a single workstation with a 6-core Intel i7-4930K processor.

to the virtually unchanged average k-point solution time, perfect scaling behaviour compared to a sequential run would be expected (i.e. only half the sequential execution time in case of two and a third of the execution time in case of three k-points in parallel.)

Going beyond three tasks (in this particular test case), the (perfect) scaling begins to break down, as the average execution time per k-point starts to increase: for 6 k-points in parallel the average completion time for a k-point is increased by more than 10 %.

However, taking the initial k-point execution time of 140 s as basis, sequentially solving 6 k-points in this test case would take a single processor about 840 s, while sixfold k-point parallelization would lead to all k-points being completed after 159 s – which is no longer "perfect" but is still a reduction of the execution time by a factor of 5.3. This result (albeit on a very simple test case) impressively demonstrates, how straightforwardly and efficiently k-point parallelization can be used.

In practice, the "optimal" parallelization scheme depends on the size of the problem as well as the available hardware. For example, if WIEN2k calculations are run a computer cluster consisting of multi-core hosts without fast interconnect that shares a file system, a combination of k-point and OpenMP usually is recommended: The k-points are distributed to the different hosts of the cluster, where multi-threaded calculations

can be executed. If a supercomputer or the required network capabilities are available, distributed parallelization (via MPI) can be considered as well.

In WIEN2k, the parallelization is controlled with a single configuration file (`.machines`). One such file to handle all three types of parallelization could look like this (assuming three hosts with 16 cores each and 24 k-points in an SCF run):

```
# .machines file
# OpenMP related directives
omp_global:16
omp_lapw1:4
# k-list splitting used in lapw1 (with additional MPI)
1:host1:4
1:host2:4
# k-list splitting continued (no MPI)
1:host3
1:host3
1:host3
1:host3
```

Using a `.machine` file like this, all programs that make use of multi-threaded libraries will use 16 threads, except `LAPW1` which will only use 4. During the execution of `LAPW1`, the 24 k-points will be split in 6 subsets of 4 k-points each, four of which will be concurrently processed by host3 (i.e. four instances of `LAPW1` will be started using 4 threads each). Hosts 1 and 2 will process one subset each utilizing hybrid parallelization – each k-point will be distributed to 4 cores (i.e. each host will start 4 instances of `LAPW1_mpi`) that additionally use 4 threads each.

CHAPTER 4

Performance and Optimization

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable.
There is another theory which states that this has already happened.

Douglas Adams, The Restaurant at the End of the Universe (1980)

This chapter deals with different optimization approaches that can be employed to improve the performance of a computer code. The main focus is on speeding up the solution of the eigenvalue problem that is at the core of any WIEN2k calculation. The chapter is divided into four parts:

1. The results of a benchmark of the status of MPI parallelization as of the beginning of the work for this thesis are presented first.
2. The potential of choosing task-appropriate or more efficient algorithms for a given task is demonstrated with two examples in the second part.
3. In the third part, parameters that can be tuned to optimize the performance of MPI parallel computation will be discussed. Moreover, comparisons between two libraries – ScaLAPACK and ELPA, which has been newly implemented as part of this thesis – will be presented.
4. Lastly, the results of a small test of mixed parallelization are presented to demonstrate that choosing the "correct" type of parallel computation is hardly straight forward.

Unless stated otherwise, all results in this chapter have been obtained for real symmetric matrices (derived from structures with inversion) using the third generation[i] of the **V**ienna **S**cientific **C**luster (VSC), the Austrian super computer[ii]. The configuration of the compute nodes was as follows:

- CPU: Intel Xeon E5-2650v2 (2 CPUs per node with 8 cores each), with 2.60 GHz and 20 MB Cache
- Memory: 64 GB per node (larger memory nodes were available but were not used for the presented computations)
- Network: Infiniband (QDR-80) in a fat tree.
- Relevant software: Linux (Scientific Linux 6.6) on both log-in and compute nodes, Intel MKL 11.3 (with LAPACK 3.5.0 and ScaLAPACK 2.0.2), Intel Fortran Compiler 16.0.1, Intel MPI 5.1

4.1 Benchmark of WIEN2k 14.2

This benchmark of the program `LAPW1` has been presented as poster paper at a HPC conference and been published in the conference proceedings [123], therefore only a summary will be given here.

A relatively small (albeit too large for sequential computation – the upper limit for matrix dimensions is 20 000 due to memory limitations) real eigenvalue was chosen as example, that still reveals interesting scaling behavior. The involved matrices were of the size 24920×24920 and 1620 eigenvalues were computed using 2 up to 512 cores. $LiBH_4$ was used as test structure, however, as this is merely a small unit cell with only 5 atoms per unit cell, a $2 \times 2 \times 2$ supercell (48 atoms per cell) was used to somewhat "artificially" increase the problem size[iii].

Wall-clock times (i.e. the real-world time elapsed between start and end of the run) were obtained with a WIEN2k routine (based on `walltime.c`) that is used by default to time programs and sub-steps. All runs have been done four times each, with the found times usually agreeing within 3 %. Instead of using the usual speedup ($S = t_{\text{sequential}}/t_{\text{parallel}}$) to evaluate the scaling of a given routine, here the parallel time is related to the time taken on two cores ($S = t_{2\text{ cores}}/t_{\text{n cores}}$). This is done because the test case is too large for sequential computation (WIEN2k limits the maximum matrix size in sequential runs). Moreover, a different algorithm is used on a single core.

The mean values of all runs are summarized in Table 4.1 – total wall-clock times as well as timings for the main setup steps (setting up the spherical and non-spherical matrix elements of H, and S, setup in the programs `HAMILT` and `HNS`, respectively), the

[i] Which has been decommissioned since.
[ii] `https://www.vsc.ac.at`
[iii] The resultant eigenvalues were not checked for correctness, as only the timings were of interest.

duration of the diagonalization (both full and iterative), and the detailed sub-steps of both schemes.

The full diagonalization is clearly the dominant sub-step across all core numbers – it takes about 75 % of the total time at 2 cores. Its share continuously increases due to the better scaling of `HAMILT` and `HNS` (at 512 cores, the setup routines only take up about 10 % of the total time). The iterative diagonalization, on the other hand, is faster then the (combined) setup at smaller core number, however, due to its worse scaling, it becomes the dominating contribution between 32 and 64 cores.

In terms of total speedup (shown in Figure 4.1), the eigensolvers scale really badly: While the iterative variant follows the linear speedup until 16 cores and only deviates from that as soon as inter-node communication starts, the speedup of the full scheme almost immediately breaks down – already at 16 cores only roughly 50 % of the expected speed increase are achieved. Notably, the full diagonalization stagnates between 64 and 192 cores. In fact, the speed decreases between 64 and 128 cores, which might be due to unfavorable processor grid dimension (cf. Section 4.3.1.1). The performance surges once more at 256 cores and remains more or less constant from then on. The iterative diagonalization scales better than the full scheme, despite also reaching 50 % of the linear speedup above 32 cores. At least, the performance increases up until 192 cores. In contrast to the full scheme, iterative solver shows a drop in performance at 256 cores. After that, basically no gain is noticeable for higher core numbers.

The setup routines `HAMILT` and `HNS` scale quite nicely until 128 cores and reasonably well beyond that. Still, given the immediate speedup breakdown of both solvers, using more than 256 cores would not be meaningful (results of benchmarks with larger cases not discussed here suggest that 512 cores might still be reasonable for larger cases).

The speedups of the sub-steps are shown in Figure 4.2: As mentioned before, the speedups of all sub-steps immediately break down as soon as inter-node communication comes into play (i.e. going beyond 16 cores). At 512 cores, all routines are below 50 % (most quite significantly) of the theoretical speedup.

The performance drop of the full diagonalization scheme around 128 cores is caused by the eigenvalue computation step (as at smaller core numbers it is the dominating step). In conjunction with the significant improvement at 256 cores, this suggests some influence of the processor grid (as 256 cores decompose into a square 16×16 processor grid which might be beneficial for this particular routine). On the other hand, both the eigenvalue estimation step as well as the setup of the reduced eigenvalue problem (`pdgemm/pdsyr2k` and `pdsymm/pdsyr2k`, respectively) show a distinct speedup drop at 256 cores, and a less pronounced one at 64 cores, indicating that square processor grids might not be beneficial for these algorithms.

The curious feature of the solution of the reduced eigenvalue problem exhibiting super-linear speedup (visible in the inset of Figure 4.2) is probably connected to the smallness of the test case: At some core number, the reduced problem gets small enough to be kept in cache, thus eliminating memory accesses. From 32 cores onward, the speedup drops

Table 4.1: "*Times of different steps of the WIEN2k subprogram* `lapw1` *- dealing with the setup and the solution of a generalized eigenvalue problem* [...]. *The dimension of the involved matrices in this example was* 24920×24920. 1620 *eigenvalues were computed with the iterative diagonalization scheme. Given are wall-clock times in seconds.*" Initially published in [123] © 2016 IEEE.

	Number of cores												
Task	2	4	8	16	32	64	128	192	256	320	384	448	512
	Total time spent in program `lapw1`												
Full	3581.6	2123.8	1172.0	728.1	409.3	254.7	253.4	201.2	88.9	92.6	87.6	82.2	73.6
Iterative	1486.9	821.0	426.0	245.8	123.8	76.4	43.1	34.3	29.9	27.6	26.7	24.7	25.3
	Setup times (same for both procedures)												
H_{sp}, S	496.3	257.7	126.6	66.1	33.0	16.7	10.0	6.7	5.1	4.1	3.7	3.2	2.8
H_{nsp}	350.5	194.7	103.2	71.9	30.9	17.1	7.8	5.7	4.6	3.8	3.2	2.8	2.5
	Sub-step times for full diagonalization												
Total	2729.2	1667.0	938.4	586.5	342.0	217.6	232.3	185.5	75.9	81.4	77.3	72.9	64.9
Cholesky	154.8	85.4	44.1	25.8	17.4	11.9	5.5	4.9	3.9	3.4	2.9	2.6	2.0
Reduction	665.5	396.2	211.6	123.8	90.9	57.8	47.8	44.4	35.1	36.3	38.1	32.8	32.1
Eigenvalues	1899.1	1177.3	678.5	434.5	232.0	146.7	178.1	135.9	36.6	41.5	36.2	37.2	30.5
	Sub-step times for iterative diagonalization												
Total	635.3	368.9	191.0	71.8	56.2	39.0	21.5	18.1	16.3	15.9	15.7	14.8	15.9
Estimate for eigenvalues	145.6	90.5	48.9	26.2	13.8	11.9	4.2	3.1	3.5	2.2	2.0	1.7	1.7
LU-Factorization	278.3	144.6	75.6	39.2	23.0	13.2	9.5	7.3	6.4	6.5	6.4	6.2	6.2
Solution linear equations	64.3	52.8	29.0	16.6	6.8	3.6	1.7	1.3	1.0	0.8	0.8	0.7	0.6
Setup reduced problem	87.5	53.3	28.6	15.7	8.1	6.7	2.5	1.8	2.0	1.3	1.1	1.0	1.0
Solution reduced problem	28.8	20.2	4.0	2.5	2.2	2.0	2.0	1.8	1.5	1.7	1.8	1.6	1.8

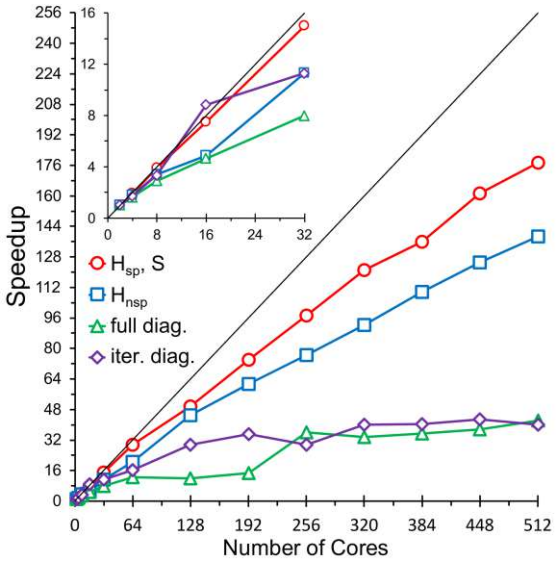

Figure 4.1: Benchmark of WIEN2k - Setup and Diagonalization: Shown are the speedups of both setup steps (setting up H_{sp}, H_{nsp}, and S) as well as full diagonalization ("full diag."), and iterative diagonalization ("iter. diag") for a real matrix (24920×24920, 1620 eigenvalues computed). Modified from [123] © 2016 IEEE.

below super-linear again (most likely due to overhead) and remains constant beyond that number of cores.

Further tests with additional openMP parallelization (see Figure 4.3) reveal that combining MPI and openMP parallelization in WIEN2k 14.2 is at best useless and in the worst case detrimental to the performance – going to 16 threads increases the execution time. The only time reduction that was found (when increasing the number of openMP threads) appeared at 128 cores and 2 threads. This lends further credence to the assumption that square grids are beneficial for some algorithms (as using 2 threads with 128 cores means that 64 cores make up a 8×8 grid).

4.2 Choice of Algorithm

4.2.1 Solving for All Eigenvalues

Routinely, calculations performed with WIEN2k only need the lowest eigenvalues – a "standard" SCF run computes between about 10 to 15 % of the eigenvalues (depending on the system). Therefore, a highly optimized version of LAPACKs `dsyevx` (which is based on inverse iteration [113]) is used in non-MPI parallelized runs [114].

However, when a larger number or all eigenvalues are needed – as is the case, for instance, when simulated NMR shifts are calculated (details on that can be found in [124–126]), the modified `dsyevx` no longer works efficiently. The expected scaling for the basis of

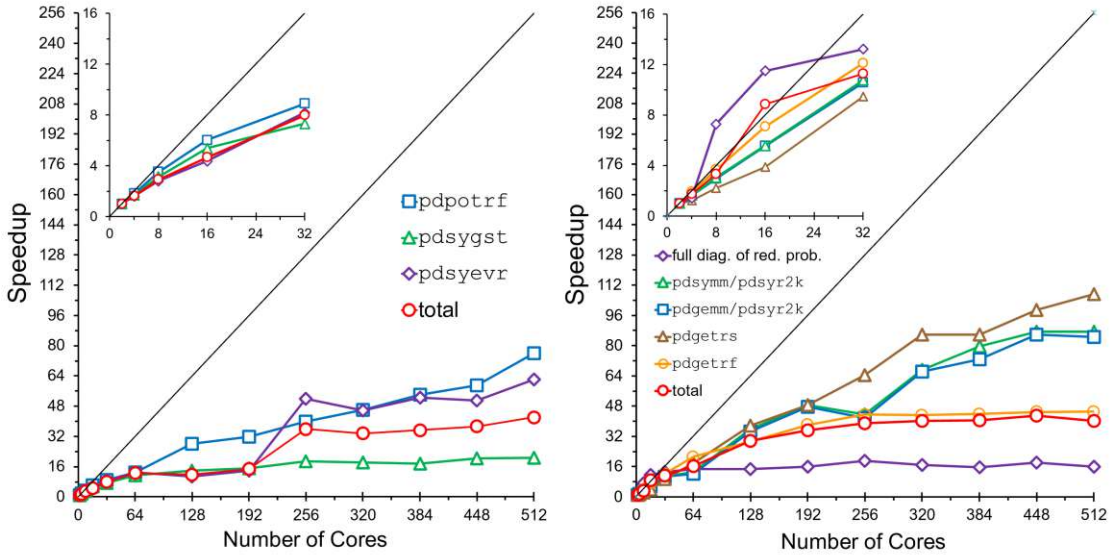

Figure 4.2: Benchmark of WIEN2k - Detailed comparison of full and iterative diagonalizations: Shown are the speedups of a real matrix (24920 × 24920, 1620 eigenvalues computed) diagonalized with (left) full diagonalization and (right) iterative diagonalization. Modified from [123] © 2016 IEEE.

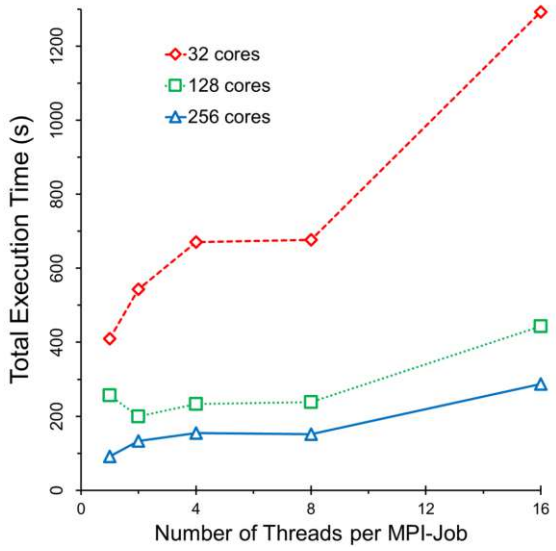

Figure 4.3: Combination of MPI and OpenMP parallelism: Shown are total execution times for different numbers of threads per MPI job for the full diagonalization of a real matrix (24920 × 24920, 1620 eigenvalues computed). Modified from [123] © 2016 IEEE.

the modified version used in WIEN2k, is $\mathcal{O}(nk)$, where n is the dimension of the matrix and k the number of eigenvalues, at best, but can increase to $\mathcal{O}(n^3)$ depending on the separation of the eigenvalues [127].

Therefore, two algorithms as implemented in LAPACK have been made available (as options for the user to choose) for computation of all eigenvalues and tested:

- `dsyevr` uses the **M**ultiple **R**elatively **R**obust **R**epresentations (MR^3) algorithm [128], which shifts the original problem (by subtracting multiples of the identity matrix) close to an eigenvalue and uses that shifted matrix to compute it. `dsyevr` is reported to scale as $\mathcal{O}(n^2)$, however, if some eigenvalues are not well separated, additional operations are needed ([127, 129].
- `dsyevd` makes use of the "Divide & Conquer" approach [130, 131], where the eigenvalue problem gets divided into two smaller problems (roughly half of the original), which are divided further until remaining small matrices can be solved, e.g. via the QR algorithm [132, 133]. Scaling is reported to be $\mathcal{O}(n^3)$ "*in the worst case*" [127], however, in benchmarks exponents below 3 have been found [127]

In practice, one has to keep in mind that the memory requirements of `dsyevd` are (potentially much, depending on n) higher than for all other eigensolvers in LAPACK.

Both above-mentioned algorithms were implemented in the sequential version of WIEN2k and tested locally on a "standard" workstation (Intel i7-4930K with 3.40 GHz and 6 cores, 32 GB RAM). The test case was again an "artificially blown-up" supercell of a small system, for which the basis set size was tuned to achieve the desired matrix dimension. The result of this test is shown in Figure 4.4.

These results clearly show that the modified `dsyevx` algorithm only is competitive below 3000 computed eigenvalues, beyond that it is drastically slower. `dsyevd` is the fastest algorithm across the tested eigenvalue range, but the difference between `dsyevd` and `dsyevr` decreases going to more eigenvalues. Scaling was found to be $\mathcal{O}(n^{2.98})$ for `dsyevd` (only slightly below the expected worst case according to [127]) and $\mathcal{O}(n^{2.60})$ – worse than the expected n^2 [127] but still significantly better than the other algorithms. It is therefore assumed that `dsyevr` will outperform `dsyevd` for even larger matrix dimensions (and eigenvalue numbers). The test was aborted at 10 000 computed eigenvalues for `dsyevx`, as both the performance and the scaling (until that point scaling behavior of $\mathcal{O}(n^{3.45})$ was found) are inferior to the other algorithms (`dsyevx` took 50 % longer to compute 10 000 eigenvalues than `dsyevr` and `dsyevd` took for 16 000). The worse than reported worst-case scaling of `dsyevx` is very likely to be due to the optimization for "few" eigenvalues.

4.2.2 Plotting Electron Density on a 3D Grid

As part of this thesis, a new program called `3DDENS` capable of providing the electronic density ρ on a 3D grid was introduced into the collection of WIEN2k.

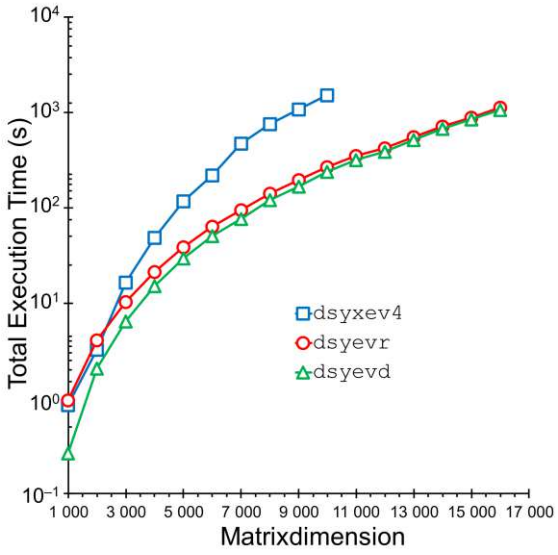

Figure 4.4: Total execution times when computing all eigenvalues as function of the number of those eigenvalues. Compared were the performances of three algorithms: the default `dsyxev4` (blue squares), `dsyevr` based on MR^3 (red circles), and the "Divide & Conquer" algorithm `dsyevd` (green triangles). Note the logarithmic time scale.

For computational reasons (e.g. to make use of symmetry as much as possible), ρ is stored via expansion coefficients (of the expansion into lattice harmonics) [7]: Analogously to the basis set decomposition into plane waves outside the atomic spheres and "atom-like" basis functions inside, the electronic density is expanded in a similar way:

$$\rho(\vec{r}) = \begin{cases} \sum_{\vec{K}} \rho_{\vec{K}} e^{\mathrm{i}\vec{K}\vec{r}} & \vec{r} \in I \\ \sum_{LM} \rho_{LM}(r) Y_{LM}(\hat{r}) & \vec{r} \in S \end{cases} \tag{4.1}$$

Here, $\vec{r}$ is the position vector and $\vec{K}$ are reciprocal vectors, $\rho_{\vec{K}}$ are the Fourier coefficients of the density expansion in the interstitial, $\rho_{LM}(r)$ is the radial contribution and $Y_{LM}(\hat{r})$ are spherical harmonics of the degree L and the order M according to [134]. Note the capitalization of L and M – this is done to avoid confusion with the quantum numbers l and m that appear in the basis set description.

However, when using non-local van der Waals functionals in WIEN2k, ρ is needed on a 3D grid (details can be found in reference [87]). Moreover, one might need 2D or 3D grids for visualization of the electronic density.

The program `LAPW5` is still used to get a 2D grid representation of the electronic density. The general procedure is summarized in Algorithm 4.1 and consists of the following main steps:

- setting up of the 2D grid

- looping over all grid points:
  - calculating interstitial density with a discrete Fourier sum
  - calculating density in atomic spheres from lattice harmonics
- writing output

Algorithm 4.1: Pseudo-code representation of the procedure used in the program `lapw5` to obtain the electron density on a 2D grid.

```
Input: read computational parameter for lapw5
Input: read plane on which electron density will be plotted
set up grid in specified plane;
Input: read Fourier coefficients
Input: read lattice harmonics
forall grid points in y-direction do
    forall grid points in x-direction do
        if (x,y) lies inside atomic sphere then
            calculate density from lattice harmonics;
        else
            calculate density from Fourier coefficients with discrete Fourier sum;
        end if
        store calculated value for corresponding grid point (x,y);
    end forall
end forall
Output: write density data to case.rho-file
```

Before the introduction of `3DDENS`, `LAPW5` was used to provide ρ on a 3D grid as well: The script `prepare_xsf` would run `LAPW5` n times to add n layers in the third dimension. Subsequently, the output of every singly run would be compiled in a file (`case.xsf`), which in turn could be used together with a program able to visualize electronic densities (for instance XCrysden[iv] [135] or VESTA)[v] [136]. Computation of discrete Fourier sums is costly, however, thus making this method very slow – especially for cases that have lots of grid points within the interstitial (e.g. molecules in a large box of vacuum that can have more than 99 % of grid points in the interstitial, see benchmark below). The Fourier sums cannot be avoided for 2D grids of the density, but seeing that those usually contain in the order of 10^4 grid points this is manageable. 3D grids usually contain about 10^6 points, which leads to large compute times.

To alleviate this, `3DDENS` (summarized in Algorithm 4.2) makes use of fast Fourier transforms (as implemented in [122]) to obtain the density on all grid points of the cell in one step. That means, `3DDENS` reads in the Fourier coefficients and uses them to

[iv] `http://www.xcrysden.org`
[v] `https://jp-minerals.org/vesta/en`

calculate the interstitial density of the cell as if no atoms were present. The second step then is – again – to loop over all grid points, check whether they fall within an atomic sphere, and, if so, replace the wrong electron density from the Fourier transform with the correct one calculated from the lattice harmonics (this step is the same as in `LAPW5`).

Algorithm 4.2: Pseudo-code representation of the procedure used in the newly implemented program `3ddens` to obtain the electron density on a 3D grid.

```
Input: read computational parameters for 3ddens
set up grid;
Input: read Fourier coefficients
perform FFTW and calculate density in whole unit cell;
store current density values on corresponding grid points;
Input: read lattice harmonics
forall grid points in z-direction do
    forall grid points in y-direction do
        forall grid points in x-direction do
            if (x,y,z) lies inside atomic sphere then
                calculate density from lattice harmonics;
                replace density at (x,y,z) with new value;
            else
                do nothing;
            end if
        end forall
    end forall
end forall
Output: write density data to case.xsf-file
```

Not only is this new method much faster than the previous method, it is also OpenMP-parallelized: The used FFTW library has a threaded version. Moreover, `3DDENS` itself parallelizes over z-values (i.e. the grid points in the third direction) showing somewhat satisfactory speedup up to 4 threads).

In the following, a brief benchmark of the new method is presented: A selection of materials with small unit cells (around 5 atoms in the unit cell) where chosen. The aim was to test all cell and lattice types. In Table 4.2 all test cases are listed.

The electronic densities were computed for all test cases with the old, `LAPW5`-based method and with `3DDENS` (both with and without OpenMP-parallelization). The results are given in Table 4.3 and visualized in Figure 4.5.

The benchmark shows impressively, how much faster the new `3DDENS` is compared to the old method. The execution times of all test cases were reduced by at least 90 %, with the exception of Ne. This is due to the fact that the old method took only 1.6 s. The largest improvement was found for H_3PO_4, where the execution time reduced by more

Table 4.2: Test set members of the 3DDENS benchmark. "Cell" is the shape of the unit cell,"Lattice" specifies the Bravais lattice (P...primitive, C...base-centered, R...rhombohedral, H...hexagonal, B...body-centered, F...face-centered), "grid" gives the number of grid points in the respective direction, and "Ratio" is the percentage of grid points within the interstitial.

	Testcase					
Parameter	**TeI**	**$CaSb_2$**	**MoF_6**	**H_3PO_4**	**Ne**	**OsO_4**
Cell	triclinic	monoclinic	orthorombic	cubic	cubic	monoclinic
Lattice	P	P	P	P	P	C
Inversion	yes	yes	yes	no	yes	yes
x-grid	144	72	144	200	48	144
y-grid	120	64	128	200	48	128
z-grid	120	144	80	200	48	72
Ratio	0.828	0.663	0.873	0.998	0.866	0.908
	Testcase					
Parameter	**MnO**	**h-BN**	**TiO_2**	**Sm_3S_4**	**$MgCu_2$**	**MgO**
Cell	trigonal	hexagonal	tetragonal	cubic	cubic	cubic
Lattice	R	H	B	B	F	F
Inversion	yes	no	yes	yes	yes	yes
x-grid	216	120	120	100	100	100
y-grid	216	120	120	100	100	100
z-grid	216	300	288	100	100	100
Ratio	0.873	0.829	0.684	0.775	0.658	0.744

Table 4.3: Comparison of the benchmark results of the new program 3DDENS. "prepare_xsf" is the old method based on LAPW5. 3DDENS was run sequentially and in parallel using 2 ("omp2") or 4 ("omp4") threads.

	Testcase					
Program	**TeI**	**$CaSb_2$**	**MoF_6**	**H_3PO_4**	**Ne**	**OsO_4**
prepare_xsf	111.7	17.5	163.4	10310.5	1.6	92.5
3ddens	6.6	1.5	6.7	16.3	0.1	4.6
3ddens (omp2)	4.0	0.9	3.8	11.6	<0.1	2.7
3ddens (omp4)	2.6	0.7	2.3	9.4	<0.1	1.7
	Testcase					
Program	**MnO**	**h-BN**	**TiO_2**	**Sm_3S_4**	**$MgCu_2$**	**MgO**
prepare_xsf	179.8	112.1	118.3	111.8	24.5	34.2
3ddens	18.7	6.0	12.0	4.4	4.1	1.9
3ddens (omp2)	11.7	4.1	7.1	2.5	2.4	1.2
3ddens (omp4)	8.4	2.9	4.5	1.5	1.4	0.8

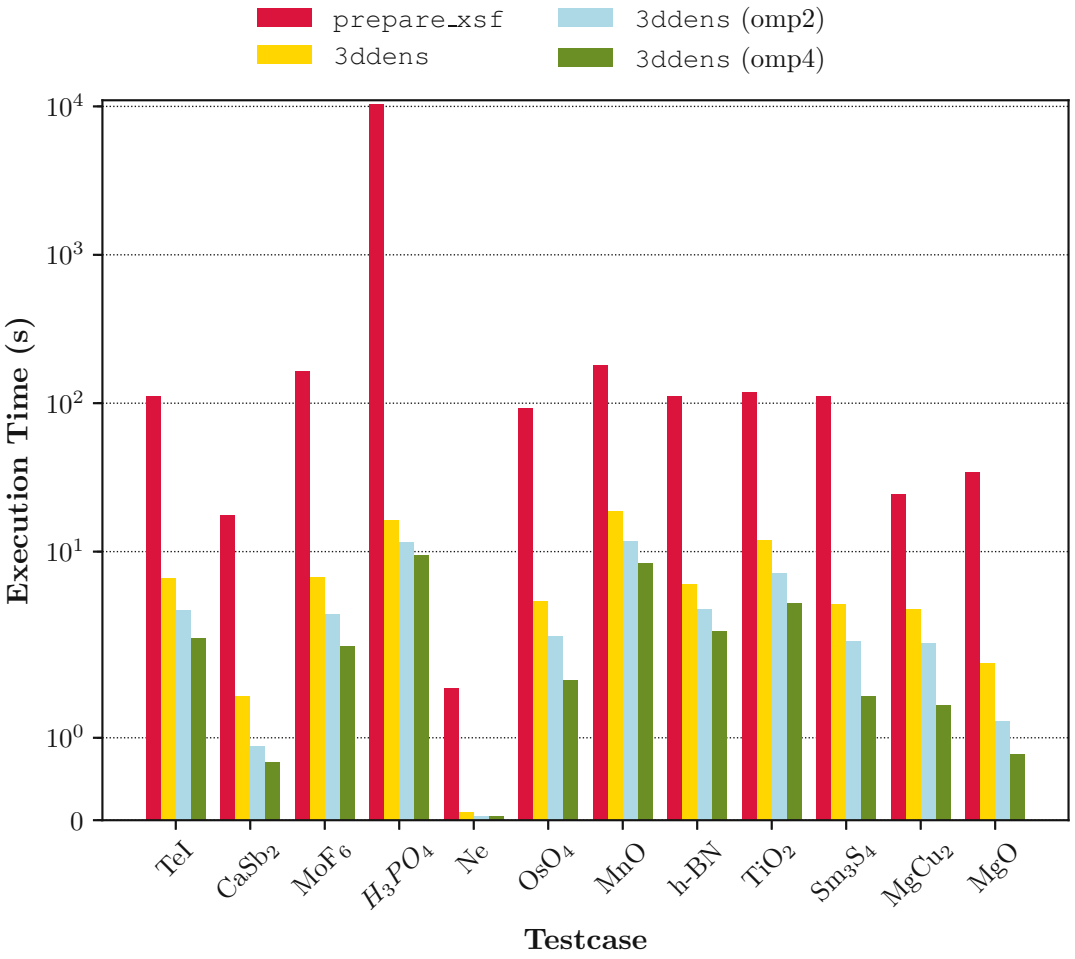

Figure 4.5: Performance benchmarks of the new program `3ddens` for calculating electronic densities on 3D grids. The times of the previously used method `prepare_xsf` (which uses layer-by-layer application of `lapw5`) is represented by the red columns. Times of three different versions of `3ddens` are shown: non-threaded (yellow) and threaded with 2 ("omp2", lightblue) and 4 threads ("omp4", green). Note the logarithmic scale on thy y-axis.

than 99 % (from almost 3 hours to 17 seconds) – which is due to the large proportion of grid points being part of the interstitial.

4.3 Using Optimized Libraries for Parallel Calculations

4.3.1 MPI Parallelization with Libraries

A couple of general considerations have to be taken into account when applying MPI parallelization to a given problem. The foremost of which is the size of the involved matrices: Parallel computing in general always introduces overhead (for example, spawning and joining threads in OpenMP, setting up the communication (MPI), and – of course – the MPI communication itself). Ideally, this should be small relative to the computation of the actual task; however, for too small problem sizes overhead might become dominant. It is perfectly possible for an over-parallelized job to actually take longer than the sequential task would. As a rule of thumb used in WIEN2k [7] states that the distributed matrices should be no smaller than 1500×1500 for parallelization to make sense.

To check WIEN2k's parallel efficiency, a one-shot benchmark was performed (the results are published in [137]) in which the time spent in MPI-specific tasks during a run of `LAPW1` (using a 21756×21756 test case and both ScaLAPACK and ELPA) was tracked with the instrumentation software Allinea (now Linaro) MAP[vi]. The results are shown in Figure 4.6.

The fraction of times spent in MPI tasks almost immediately jumps up – at 64 cores it already surpassed 50 %. The increase levels off beyond 160 cores (75 % and remains almost constant around that value. For 416 and 512 cores, another increase to about 85 % was recorded. In terms of execution times of `LAPW1`, it has to be noted that in this particular case no performance gain could be achieved beyond 64 cores, and for the two largest core numbers, the times began to grow again. The same behavior was observed both for ScaLAPACK and ELPA.

This result showcases quite impressively that one always has to keep in mind how to best parallelize any given problem. Two further necessary considerations (matrix decomposition and making sure that the parallel system is configured properly) are presented below.

4.3.1.1 Influence of Matrix Decomposition

The question of matrix decomposition actually consists of two interconnected parts:

- How is the matrix itself decomposed?
- How are the sub-matrices (however many there are) distributed among the cores?

[vi] The 2016 version 6.0.6 of Allinea Map was used. The current Linaro MAP can be found here: `https://www.linaroforge.com/linaroMap`

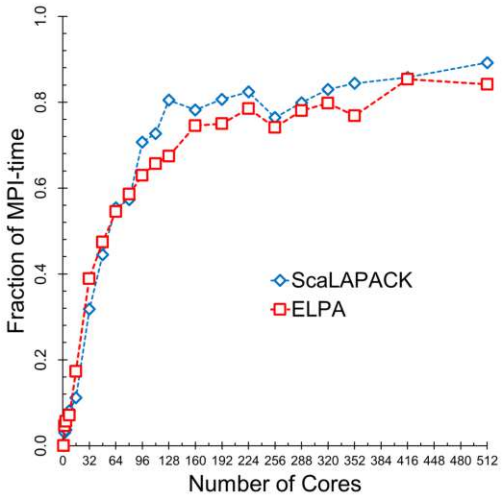

Figure 4.6: Fraction of time spent in MPI communication.

The first question is compounded by the fact that LAPACK and ScaLAPACK make use of the symmetric or Hermitian properties of the matrices used in WIEN2k calculations (i.e. $a_{i,j} = a_{j,i}$ and $a_{i,j} = a^*_{j,i}$, respectively), because in this way only half of the entries above or below the diagonal and the diagonal itself have to be stored. Figure 4.7 schematically shows four possible ways to distribute a $M \times M$ matrix: The first option (Figure 4.7a) comes with massive load imbalances – CPU2 will only get zero values, while CPU1 will only get non-zeroes. Options b and c (Figure 4.7b,c) are only marginal improvements (and both actually equivalent, as the matrix is square. There is still a large disparity between the amount of non-zero values every CPU is getting. The solution to these load balancing issues is what is called "block-cyclic distribution" (shown in Figure 4.7d), which is used in ScaLAPACKto handle, for instance, load balancing in diagonal matrices. The matrix is divided in smaller blocks, characterized by their "block size" (which is the number of matrix elements in each block and not necessarily a square number). These blocks are subsequently distributed to the processors in a cyclic manner. One way to visualize this process is to think of covering the divided matrix with the processor grid $P \times Q$ and assigning the blocks accordingly.

Now that the matrix is decomposed in blocks, the question of the optimal distribution to the processors remains open: Both block size (block size tests will be discussed in Section 4.3.3) and layout of the processor grid can be tuned for optimal performance. The grid shape of the processor grid is determined at runtime in WIEN2k and set up such, that the grid is "as square as possible" (i.e. $P + Q$ is minimal). That means for 32 cores, a 8×4 grid is chosen over a 16×2 configuration. For non-square grids both $P \times Q$ and $Q \times P$ are possible. Table 4.4 shows tests of the effect of the grid shape ($P \times Q$, which is the WIEN2k default, vs. $Q \times P$) for `pdsyevr` and `pdsygst` (`pdsyevr` displayed a very bad performance for 128 cores in the WIEN2k benchmark in the previous section). The same test case as for the Benchmark was used.

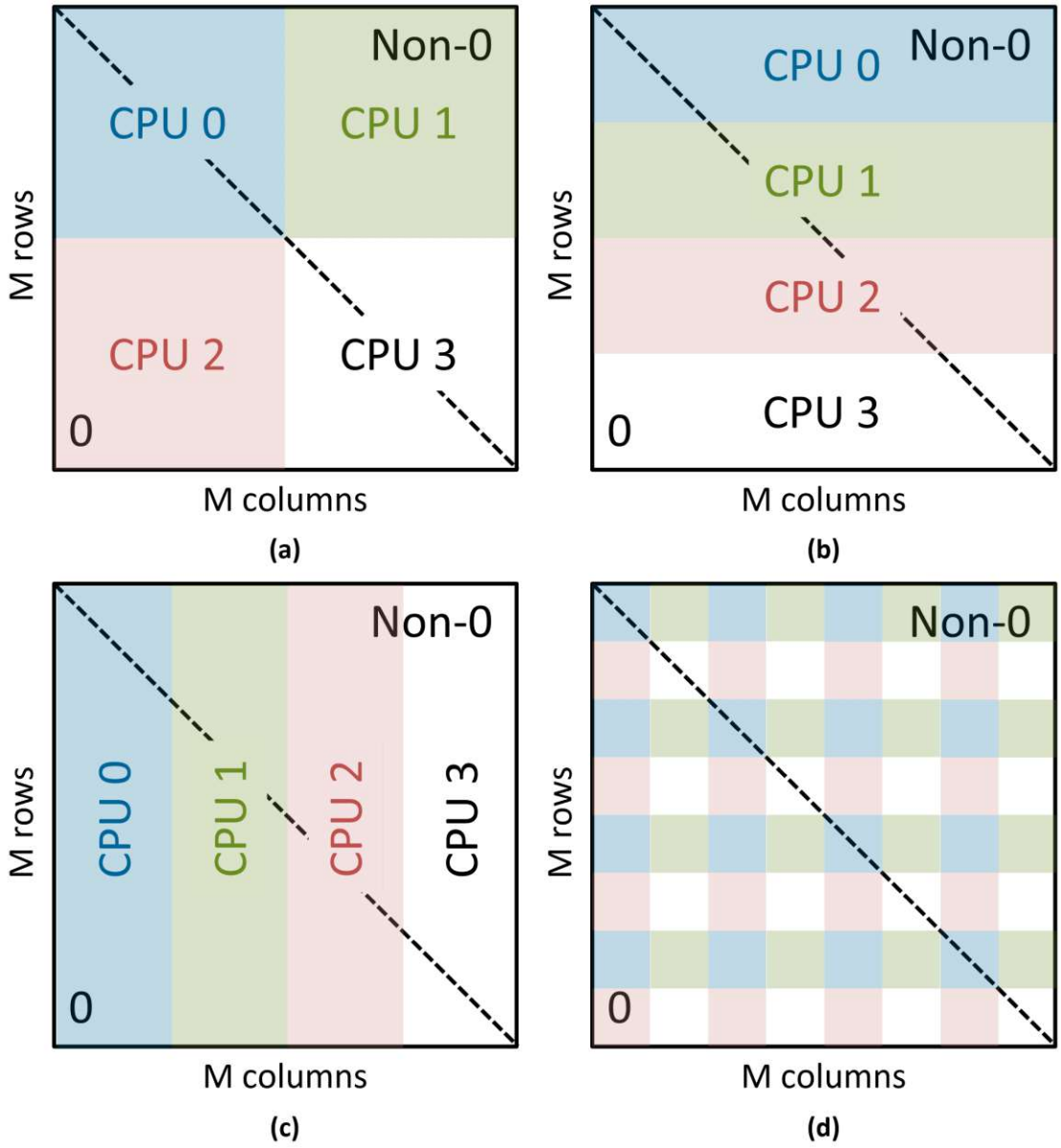

Figure 4.7: Possible ways of distributing a symmetric or hermitian upper diagonal $M \times M$ matrix among 4 CPUs: The distribution to a 2×2 grid shown in (a) has severe load balancing issues – all entries of the subset of CPU1 are non-zero, while CPU2 has only zeroes. 4×1 (b) and 1×4 (c) grids are only slightly better, as CPU0 has only a small fraction of non-zero matrix elements. The best way to distribute matrix elements among multiple processors is what is called a "block-cyclic distribution": A "block size" is defined as new computational parameter that is used to subdivide the whole matrix. These smaller blocks are then distributed cyclically (according to the chosen processor grid) among all CPUs. Note that the blocks do not have to be square.

Table 4.4: Execution times of `pdsyevr` and `pdsygst` for different grid geometries: The grid dimension in WIEN2k are chosen at runtime in a way, that $P + Q$ (P is the larger and Q the smaller grid dimension) is minimized, making the resulting grid "as square as possible".

	pdsyevr			pdsygst		
# of Cores	**P×Q**	**Q×P**	**diff.**	**P×Q**	**Q×P**	**diff.**
2	1942.9 s	2036.8 s	5%	665.4 s	778.7 s	7 %
8	697.6 s	721.4 s	3%	217.0 s	208.0 s	-4%
32	238.5 s	256.3 s	8%	88.4 s	78.6 s	-11%
128	170.1 s	65.7 s	-61%	46.7 s	39.2 s	-16%
512	30.0 s	80.7 s	169%	31.5 s	25.9 s	-18%
2048	36.2 s	69.5 s	92%	24.1 s	20.6 s	-14%

The measured times reveal that the performance drop of `pdsyevr` at 128 cores is most likely caused by an unfavorable processor grid. Using a 6×18 grid instead of the default decreases the time spent in `pdsyevr` by 61 %. However, at all other core numbers the default grid shape is beneficial. `pdsygst` on the other hand shows inverted behavior: For core number larger than 2 the $Q \times P$ grid was found to be beneficial and reducing the time between -4 and -18 %. That means there is a necessarily a trade-off, as the different algorithms of the eigensolver scheme show inverted behavior. To account for this, a new option to WIEN2k's `run_lapw` script has been added. If switched on, WIEN2k will use both grid geometries in the first two iterations of the SCF cycle and automatically pick the faster one for the following iterations.

4.3.1.2 Influence of Proper CPU Binding

After preliminary tests on VSC3 (`LAPW1`, matrix size 26 252, 16 cores, MPI only) of the influence of the block size exhibited sharp and non-reproducible spikes in the compute time, repeat calculations revealed an unusually large spread of execution times up to 50 % (left side of Figure 4.8). It turned out that these deviations had two causes not produced by WIEN2k:

- Processes jumping between CPU cores, thus leading to longer memory access times
- Cores being assigned more than one core, which hampers performance for obvious reasons

Usually, proper CPU-process assignment is handled by the operating system, however, sometimes configurations can be off. To fix this problem, CPU binding can be enforced at run-time (`--cpu_bind=map_cpu`), however, the downside is that this has to be tuned specifically for the used hardware. After ensuring pinning, the same repetition test was re-run with excellent reproducibility of the observed execution times (a variance of about 1 % was found).

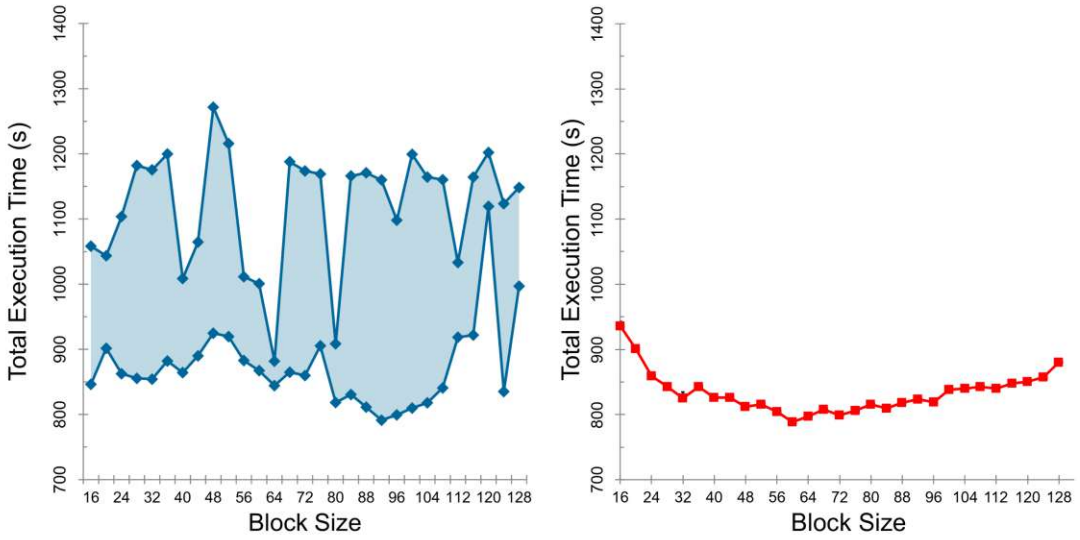

Figure 4.8: Influence of CPU binding: The left side shows the spread between repeated measurements of the execution time of `LAPW1` (matrix size 26 252, 16 cores, MPI only) – deviations up to 50 % are observed. The right plot shows the same repeat measurement after explicit CPU binding was turned on with hardly any variance visible (about 1 %).

4.3.2 Eigenvalue SoLvers for Petaflop-Applications (ELPA)

The open-source library ELPA (**E**igenvalue So**L**vers for **P**etaflop-**A**pplications)[111, 112][vii] was implemented in WIEN2k as part of this thesis.

Nowadays, ELPA is widely used in many electronic structure codes. It is used on top of ScaLAPACK, which is actually a prerequisite, as in WIEN2k, for example, only the computation of the eigenvalues is handled by ELPA, which also means that it uses block-cyclic distribution as well, thus making it quite easy to integrate into existing codes. All other algorithms used in the eigensolvers of WIEN2k still are part of ScaLAPACK. That means, the general strategy of using ELPA remains the same as laid out in the description of the full diagonalization scheme of WIEN2k (Section 3.4.2):
A Cholesky factorization of the general eigenvalue problem followed by the transformation into standard form are performed. As first part for the eigenvalue step, ELPA performs a tridiagonalization, as do all other solvers presented so far. The difference lies in the specific implementation of this step. ELPA uses Householder transform [138] to arrive at a tridiagonal matrix – either directly in one step (this algorithm is called elpa1) or in a 2-step procedure by way of a band matrix in between (elpa2) [112] – this is illustrated schematically in Figure 4.9. To do this, the complete matrix is needed instead of only a upper or lower diagonal matrix. Therefore, the implementation of ELPA included an additional step to setup the full matrix.

[vii] `https://elpa.mpcdf.mpg.de`

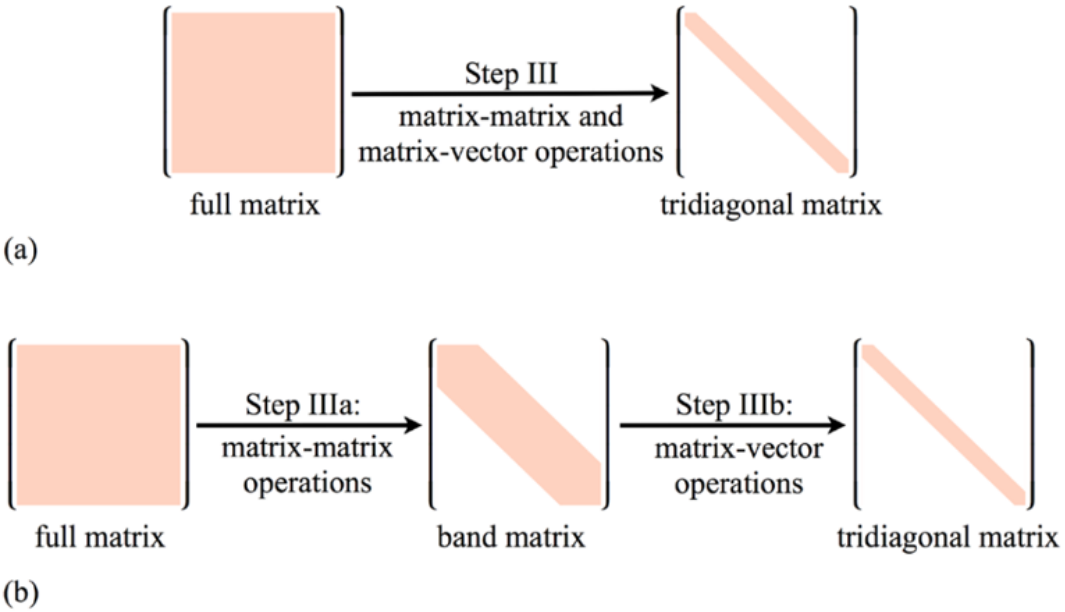

Figure 4.9: Schematic representation of the difference between elpa1 and elpa2. Householder transformations are used in both cases, however, elpa1 (a) uses a 1-step approach; elpa2 (b) uses two steps with a band matrix as intermediary. Reproduced with permission from [112], © IOP Publishing.

Figure 4.10 shows a first test of ELPA (both elpa1 and elpa2) compared to `pdsyevr` up to 2048 cores (the same test case as for the initial benchmark was used again): While elpa1 starts with similar timings up to the performance irregularity of ScaLAPACK at 128 cores, it performs better from then on. elpa2 immediately outperforms both ScaLAPACK and elpa1 almost across the whole range of core numbers – only at 2048 does elpa1 catch up. Both ELPA alternatives reduce the execution time until 1024 cores, elpa1 even beyond. Technically, elpa1 scales better (it shows higher speedups) compared to elpa2, however, in absolute times elpa2 remains faster. For that reason, ELPA (more specifically, elpa2) is the new parallel default in WIEN2k (provided it is installed on the respective system).

The additional matrix completion step is negligibly fast (it was found to be below 0.5 % of the total runtime). It was realized with the level 3 PBLAS routines `pdtran` and its complex analogue `pdtranc`. The only slight drawback here is the fact, that this completion imposes higher memory requirements, since an additional array of the size of H is needed[viii].

[viii] An alternative hand-coded method that utilizes a much smaller array h_{help} and explicit MPI communication was implemented but is currently not functional due to the switch from upper to lower diagonal matrices for the Cholesky factorization in newer WIEN2k versions.

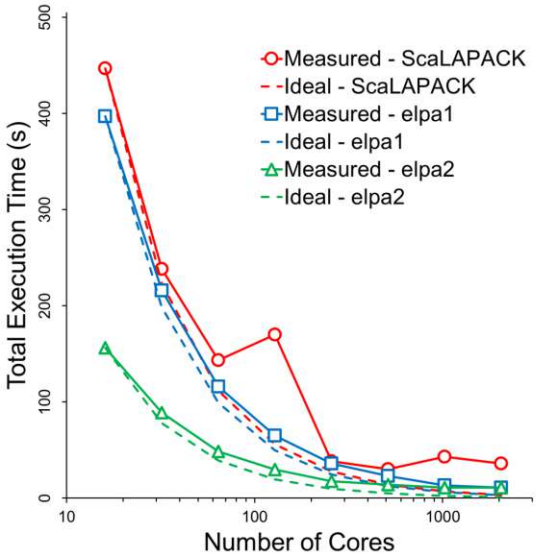

Figure 4.10: Scaling comparison of ScaLAPACK and ELPA up to 2048 cores. The "ideal" times (dashed lines) are the times corresponding to linear speedup related to the time at 16 cores. The performance irregularity at 128 cores in case of ScaLAPACK, where the algorithm is actually slower than before, is the reproduced behavior described in the benchmark (Section 4.1).

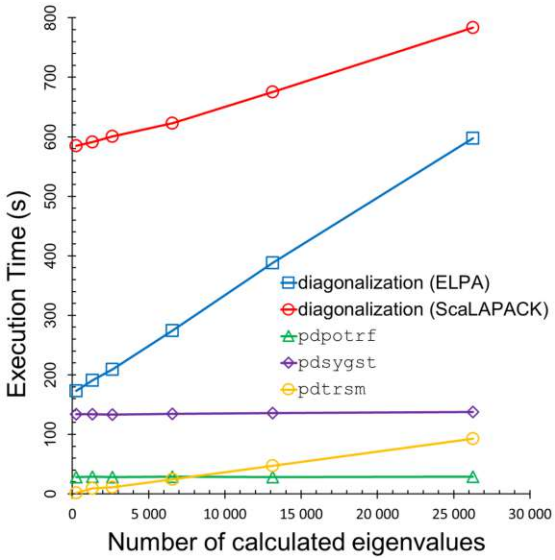

Figure 4.11: Execution time of the main algorithms of the full diagonalization dependent on the number of calculated eigenvalues. The dependence on the number of eigenvalues of the eigensolving steps (both ELPA and ScaLAPACK), `pdpotrf` (Cholesky factorization), `pdsygst` (transformation to standard form), and `pdtrsm` (back transformation of the eigenvectors) are compared for a test matrix of size 26252×26252 on 16 cores.

4.3.3 Performance Benchmarking with ELPA

After the implementation of ELPA, different tests and comparisons with the ScaLAPACK have been performed to investigate performance improvements and effects of computational parameters like core number, number of computed eigenvalues, and block-size. The results of those tests are summarized below.

A test (with matrix sizes 26252×26252 on 16 cores) of the effect the number of computed eigenvalue has on the performance of the algorithms involved in the full diagonalization scheme shows no dependence in case of the Cholesky factorization (`pdpotrf`) and the transformation to standard form (`pdsygst`) (Figure 4.11). Which is as expected, as both of these algorithms act on entire matrices independently from the number of eigenvalues needed. The time consumption of ELPA grows almost perfectly linearly – and while the gap closes, as the ScaLAPACK diagonalization time grows more slowly, ELPA remains faster up to 100 % of computed eigenvalues.

The influence of the block size was tested as well for three different matrix sizes of 21756×21756, 26252×26252, and 31304×31304 on 16 and on 32 cores (Figures 4.12 and Figure 4.13). Both ScaLAPACK and ELPA scale nicely going from 16 to 32 cores. In terms of block size dependence, both libraries show similar behavior: Both seem to perform better for smaller block sizes with estimated optima around 64 and 96, respectively. Especially the curves for ELPA are very shallow, which makes it hard to assign a "real optimum". The spread of the different block size dependent execution times are very comparable as well: Both libraries exhibit differences between the highest and the lowest time in the order of 20 to 30 %, which means that via choosing the block size correctly, enhancing the performance would be possible relatively easily.

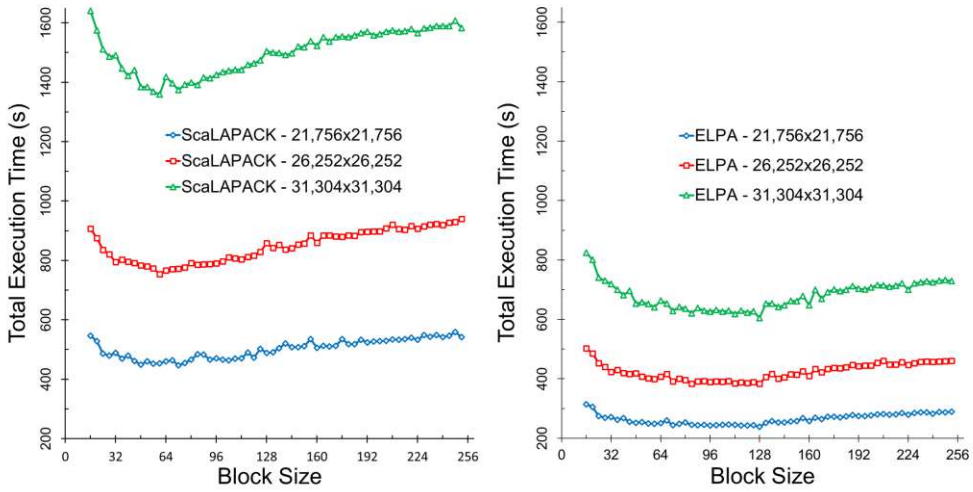

Figure 4.12: Block size test on 16 cores. Timings of ScaLAPACK (left) and ELPA (right) for different matrix sizes are compared over a block size range from 16 to 256.

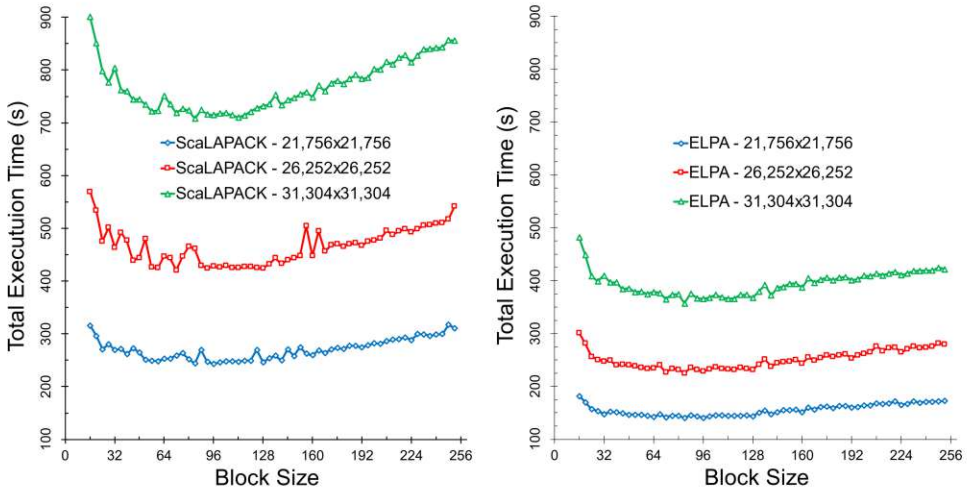

Figure 4.13: Block size test on 32 cores. Timings of ScaLAPACK (left) and ELPA (right) for different matrix sizes are compared over a block size range from 16 to 256.

4.4 Hybrid Parallelization

Inspite of the initial benchmark presented in this thesis (Section 4.1) showed no benefits in using OpenMP and MPI parallelization at the same time, a small benchmark was performed with a newer version of WIEN2k (WIEN2k 18), as OpenMP improvements have been provided by the community (credit goes to P. Ondracka). The results of this new benchmark are summarized in Tables 4.5 and 4.6. Note, that this benchmark uses the MPI version of `LAPW1` with ScaLAPACK routines – that means that even the single-processor run was performed with parallel algorithms from this library.

When using either pure OpenMP or pure MPI, an interesting observation can be made: The setup of the spherical matrix elements scales fairly well with both methods (albeit worse with OpenMP). While the non-spherical setup and the diagonalization show almost identical scaling in case of MPI, inverse behavior is found for OpenMP: The diagonalization scales better with OpenMP than with MPI, the non-spherical setup shows a pronounced decrease in speed up with OpenMP.

For mixed parallelization, no clear trends emerge: 16 mixed cores (either 2 threads and 8 MPI processes or 4 of each) compare reasonable well to the timings found for 16 MPI processes. The same is true for both setup steps and 8 mixed cores, however using 4 threads and 2 MPI processes yields a very curious timing for the diagonalization (at almost twice the execution time of 8 cores used purely with MPI.

According to this benchmark, hybrid parallelization is still not recommended (at least for small cases). In principle, pure OpenMP parallelization could be used as an alternative to MPI, as the most time consuming diagonalization step scales better in this benchmark. However, the sequential version of `LAPW1`, which should rather be used for non-MPI

calculations, does not scale as well with the number of threads (4 threads usually give good performance gains, any number larger than 4, however, are usually not reasonable)[7].

Table 4.5: Parallel execution times of a small hybrid parallelization benchmark. Given are the wall-clock times of the matrix setup (spherical and non-spherical elements, respectively) as well as the diagonalization step within the full diagonalization scheme. A test case of matrix dimensions 14404×14404 was used.

OMP	MPI	$H_{\mathbf{sp}}$, S	$H_{\mathbf{nsp}}$	`diag`	Total
1	1	137.6 s	122.2 s	936.7 s	1198.8 s
1	2	67.7 s	65.3 s	559.2 s	696.0 s
1	4	35.6 s	39.8 s	288.7 s	365.6 s
1	8	20.0 s	22.8 s	188.5 s	232.8 s
1	16	10.9 s	16.7 s	133.2 s	162.4 s
2	1	73.7 s	67.2 s	538.4 s	681.6 s
2	4	20.1 s	24.5 s	184.4 s	230.7 s
2	8	11.4 s	13.9 s	143.1 s	169.9 s
4	1	42.1 s	41.3 s	349.1 s	434.7 s
4	2	22.9 s	25.6 s	315.1 s	365.4 s
4	4	12.0 s	15.3 s	145.1 s	174.0 s
8	1	20.9 s	26.8 s	109.4 s	159.2 s
16	1	15.2 s	28.6 s	91.6 s	137.6 s

Table 4.6: Parallel execution speedup of the benchmark in Table 4.5

OMP	MPI	$H_{\mathbf{sp}}$, S	$H_{\mathbf{nsp}}$	`diag`	Total
1	1	1	1	1	1
1	2	2.03	1.87	1.68	1.72
1	4	3.87	3.07	3.24	3.28
1	8	6.88	5.36	4.97	5.15
1	16	12.62	7.32	7.03	7.38
2	1	1.87	1.82	1.74	1.76
2	4	6.85	4.99	5.08	5.20
2	8	12.07	8.79	6.55	7.05
4	1	3.27	2.96	2.68	2.76
4	2	6.01	4.77	2.97	3.28
4	4	11.47	7.99	6.46	6.89
8	1	6.58	4.56	8.56	7.53
16	1	9.05	4.27	10.23	8.71

To also compare hybrid parallelization with the sequential version, a second benchmark with matrix dimensions of 17808 × 1708 was performed on a compute node of the local cluster of the WIEN2k group (Intel i7-7820X with 3.60 GHz and 8 cores). The execution times are listed in Table 4.7), the speedup is summarized in Table 4.8 – here, the runtime of the unthreaded sequential run was used as reference for the speedup. "MPI = 0" calculations have been run with the sequential version of `LAPW1`, all other calculations used the MPI-version with the respective parallel library. Hyperthreading was used for the 16-core cases.

For the sequential version, satisfactory scaling with the number of thread was observe until 4 threads, going beyond that is not sensible. The "MPI pure" runs with ELPA and ScaLAPACK nicely illustrate the advantage of ELPA – it is faster by at least 30 % in all cases and scales better as well (albeit on a rather low level). However, using up to 8 cores still leads to a performance gain for both libraries. Irregular timings of ScaLAPACK with only one MPI process were observed, which could be connected to issues with the configuration or be a bug of the local implementation, additional testing would be necessary to investigate this further.

When using a second thread, all times grow worse compared to the MPI-only times with the same total number of cores (and the irregularity of ScaLAPACK with only one MPI process gets compounded further), with the exception of ELPA with two threads and 4 MPI processes (this combination is slightly faster than ELPA with 8 MPI cores), however, this would mean super-linear speedup which is unlikely. Further tests would be required here as well.

A comparison of threaded and MPI runs (leaving aside the mixed parallelization) reveals a slight edge of the sequential `LAPW1` over ELPA at fewer nodes, ELPA takes over at 8 cores. ScaLAPACK is slower up until 8 cores. Another strange observation with this respect is the fact that the setup of H_{nsp} appears to be faster when using one MPI job, which is counter intuitive, because it would mean that `pdsyr2k`, which is the algorithm mainly used in the program `HNS` that sets up H_{nsp}, is faster than its sequential counterpart.

Hyperthreading (i.e. using 16 cores on a system with only 8 physical cores, which leads to every CPU running two processes at once) slightly decreases performance.

This second benchmark confirms the conclusion of the one above: Hybrid parallelization is currently not recommended.

Table 4.7: Execution times of a second benchmark performed on a compute node of the local cluster of the WIEN2k group (Intel i7-7820X with 3.60 GHz and 8 cores). Given are the wall-clock times of the matrix setup (spherical and non-spherical elements, respectively) as well as the diagonalization step within the full diagonalization scheme. "MPI = 0" calculations have been run with the sequential version of `LAPW1`, all other calculations used the MPI-version with the respective parallel library. Hyperthreading was used for the 16-core cases. A test case of matrix dimensions 17808×1708 was used. Data courtesy of Peter Blaha.

OMP	**MPI**	$H_{\mathbf{sp}}$, S	$H_{\mathbf{nsp}}$	**diag**	**Total**
Sequential					
1	0	113.3 s	80.4 s	297.6 s	493 s
2	0	60.9 s	40.0 s	176.9 s	280 s
4	0	34.0 s	27.5 s	116.3 s	180 s
6	0	25.6 s	23.8 s	94.3 s	145 s
8	0	23.4 s	22.4 s	84.7 s	132 s
ELPA					
1	1	120.1 s	57.5 s	336.5 s	518 s
1	2	63.1 s	29.9 s	186.8 s	283 s
1	4	33.8 s	18.7 s	107.0 s	132 s
1	8	18.6 s	10.0 s	74.9 s	111 s
2	1	114.9 s	84.4 s	581.0 s	784 s
2	2	55.5 s	34.0 s	203.9 s	297 s
2	4	20.0 s	14.1 s	71.7 s	109 s
2	8	18.3 s	14.4 s	83.6 s	124 s
Scalapack					
1	1	120.5 s	57.8 s	1379.1 s	1561 s
1	2	63.2 s	29.9 s	291.0 s	387 s
1	4	32.8 s	17.8 s	166.5 s	220 s
1	8	19.4 s	10.1 s	114.6 s	148 s
2	1	116.1 s	84.7 s	1528.2 s	1737 s
2	2	55.4 s	34.1 s	310.9 s	404 s
2	4	20.3 s	14.0 s	116.8 s	154 s
2	8	18.3 s	14.2 s	125.6 s	162 s

Table 4.8: Speedup of the benchmark in Table 4.7

OMP	**MPI**	$H_{\mathbf{sp}}$, S	$H_{\mathbf{nsp}}$	**diag**	**Total**
Sequential					
1	0	1	1	1	1
2	0	1.86	2.01	1.68	1.76
4	0	3.33	2.92	2.56	2.74
6	0	4.43	3.38	3.16	3.40
8	0	4.84	3.59	3.51	3.73
ELPA					
1	1	0.94	1.40	0.88	0.95
1	2	1.80	2.69	1.59	1.74
1	4	3.35	4.30	2.78	3.73
1	8	6.09	8.04	3.97	4.44
2	1	0.99	0.95	0.51	0.63
2	2	2.04	2.36	1.46	1.66
2	4	5.67	5.70	4.15	4.52
2	8	6.19	5.58	3.56	3.98
Scalapack					
1	1	0.94	7.96	0.22	0.32
1	2	1.79	4.52	1.02	1.27
1	4	3.45	2.69	1.79	2.24
1	8	5.84	1.39	2.60	3.33
2	1	0.98	0.95	0.19	0.28
2	2	2.05	2.36	0.96	1.22
2	4	5.58	5.74	2.55	3.20
2	8	6.19	5.66	2.37	3.04

CHAPTER 5

Optimizing Atomic Positions

"Ah, I've got an idea!" said the Dean, beaming. "We can get Hex to reverse the thaumic flow in the cthonic matrix of the optimized bi-direction octagonate, can't we?"

Terry Pratchett, Ian Stewart, and Jack Cohen, The Science of Discworld (1999)

The following chapters deal with three exemplary use cases of WIEN2k. All DFT calculations were performed using the PBE functional [52]; all other important computational parameters, for instance the radii of the atomic spheres (R_{MT}), the k-mesh size for Brillouin zone sampling, and the parameter $R_{\mathrm{MT}}^{\mathrm{min}} K_{\mathrm{max}}$ (RK_{max} in short) that is the product of the smallest atomic sphere and the largest reciprocal vector K_{max} and that determines the size of the used basis set, will be given for each case.

When evaluating **X**-**R**ay **D**iffraction (XRD) data of chlorothionite, Berthold Stöger found diffuse diffraction (streaked reflections – which appear as consequence of stacking faults in the material) that could not entirely be explained with the structures extracted from experimental data. Figure 5.1 shows the observed reflections of the crystallographic layer hk3: Two different types of streaks can be seen. The first type (highlighted with the dotted box) derives from stacking faults in the layered crystalline material and can be simulated using XRD data alone. The second type (dashed box in Figure 5.1), however, arises due to local distortions that are not visible in XRD measurement as they are averaged out due to the large number of layers. Here, DFT offers an opportunity to find these local distortions: The optimization of atomic positions[i] is a crucial first step of many DFT calculations and can be used to find local distortions. This first use case demonstrates the importance of this step.

[i] Often also called "structure minimization", as the goal is to minimize residual forces acting upon atoms that arise from atoms not occupying their equilibrium positions.

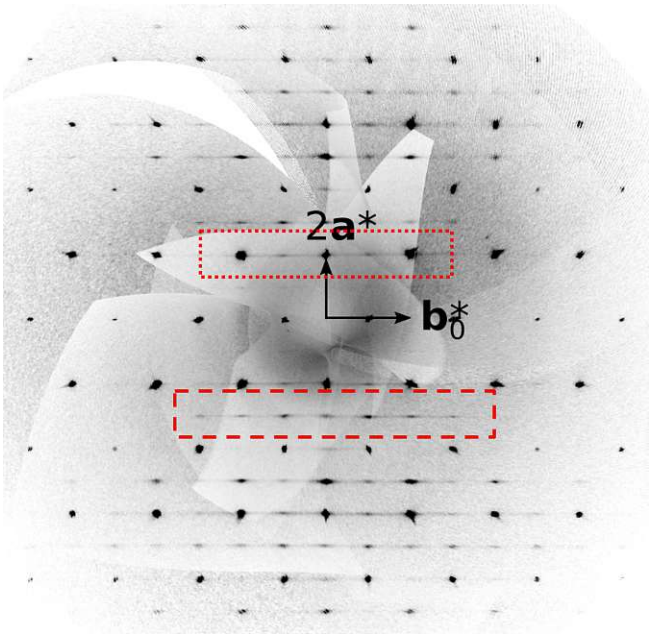

Figure 5.1: Diffraction pattern (hk3 layer) of chlorothionite with two types of streaks: The streaks marked with the dotted box arise due to stacking faults in the crystal structure, the second type (in the dashed box) are caused by local distortions that cannot be resolved directly from an XRD experiment [139].

5.1 Experimental Background

5.1.1 Order-Disorder (OD) Structures

Figure 5.2: A piece of chlorothionite of roughly 8 cm, found at the Tolbachik Volcano, Kamchatka Oblast, Russia. Image taken from `www.mineralienatlas.de` with permission of the copyright holder Luigi Chiappino.

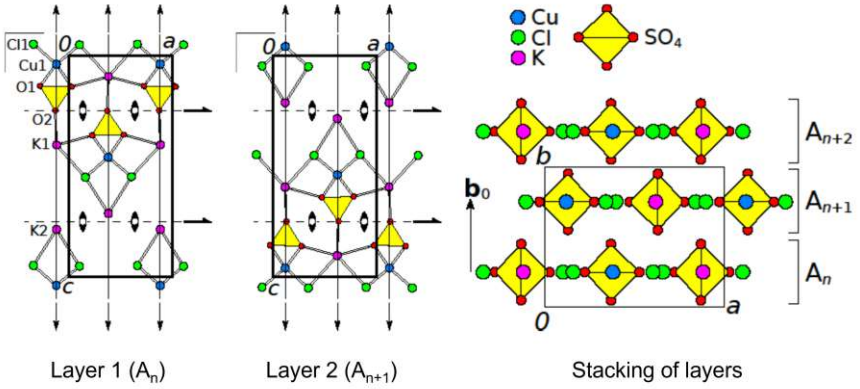

Figure 5.3: The unit layers of the layered material chlorothionite (Layer 1 and Layer 2) and the stacking order (1-2-1-...). The pseudo-symmetry elements (2-fold rotation axes and mirror planes) of both layers 1 and 2 are shown as well. [139]

Chlorothionite is a naturally occurring lightblue to greenish mineral that is classified as a sulfate (Figure 5.2). Chemically, it is a double salt with the formula $K_2Cu(SO_4)Cl_2$.

The structure of chlorothionite has been known since the 1970s [140] – chlorothionite is a layered material and an **O**rder-**D**isorder (OD) structure [141, 142]. That means that the individual layers have higher symmetry than the overall structure. In Figure 5.3, the two different unit layers ("Layer 1" and "Layer 2") and the stacking are shown. The pseudo-symmetry elements (2-fold rotation axes and mirror planes) of layers 1 and 2 are displayed as well – these symmetry elements are valid only for the layer, but not for the entire structure.

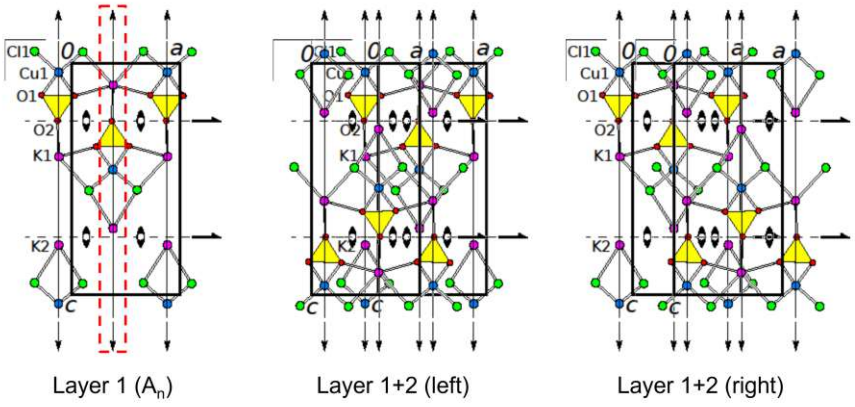

Figure 5.4: Two possibilities of stacking layer 2 of chlorothionite on layer on 1. The highlighted mirror plane (red dashed box) in layer 1 is the reason for the diffuse diffraction, as two inequivalent double layers can be set up: The unit cell of layer 2 can go to the left or to the right of the mirror plane of layer 1 [139].

A direct consequence of the layer symmetry being higher than the overall symmetry of the stacking is visualized in Figure 5.4. Starting from the first layer, there are two possibilities of adding the second layer on top: Either to the left of the highlighted mirror plane or to the right. That means that indefinitely many different stacking orders are possible.

5.1.2 X-Ray Diffraction (XRD)

X-Ray **D**iffraction is a powerful tool that is widely applied to study crystalline samples (e.g. powders or single crystals). At its core, it uses the phenomenon of diffraction which occurs if the distance between diffracting planes is in the order of the wave length of the incoming beam. This makes X-rays ideally suited to be used with crystalline materials, as they are ordered (therefore form potentially diffracting planes) and the wave length of X-rays matches the typical distances in crystalline materials of about 10^{-10} m.

When X-rays interact with materials (specifically the electron density of it), they get scattered, producing a secondary wave that emanates in all directions. This happens multiple times at neighboring atoms when a samples is hit by X-rays, however, most of the scattered secondary waves interfere destructively and cancel each other out. Only under certain incident angles in combination with matching atomic distances can constructive interference lead to scattered X-rays that can be detected. Bragg's law gives the condition for this to happen (Equation 5.1).

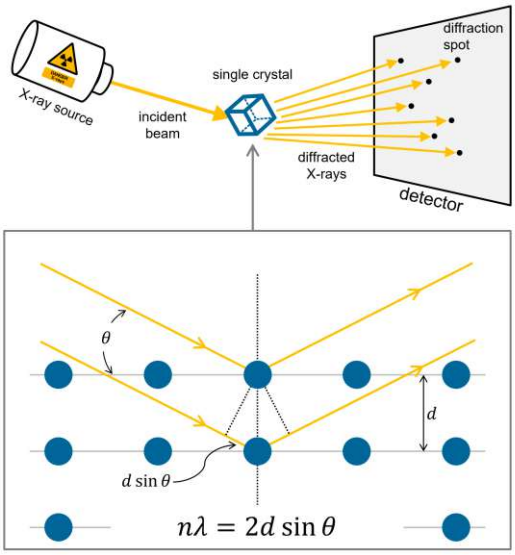

Figure 5.5: Schematic representation of the basic principle of XRD. X-rays from hit a sample (here a single crystal, but powders are possible as well), get diffracted according to Bragg's law and get detected. The angles under which constructive interference occurs give information about the structure of the sample.

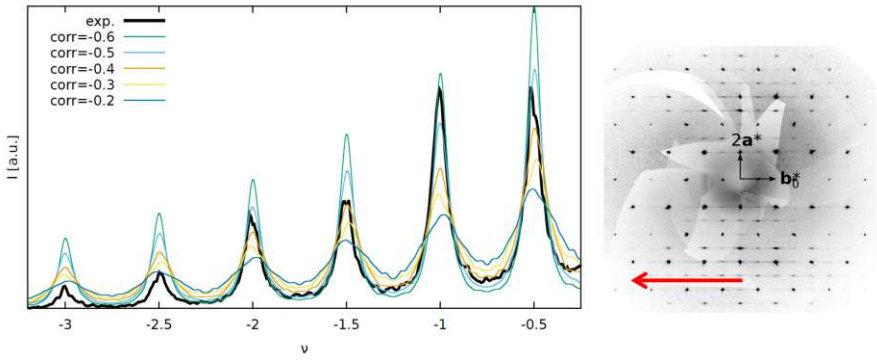

Figure 5.6: Simulation of the disorder related streaks in chlorothionite. Simulated peaks appear above the red arrow in the diffraction pattern (right), going from the center to the left (and from right to left in the simulated peak profile) [139].

$$n\lambda = 2d \sin\theta \,, \tag{5.1}$$

where λ is the wavelength of the X-rays, d is the distance between lattice planes at which diffraction occurs, θ is the incident angle, and n is the diffraction order. Figure 5.5 shows a simple schematic of the basic setup and the diffraction process at lattice planes.

5.2 Simulating the Diffraction Peaks

Using just the data from the experiment, the extracted unit cell (for a single layer – the two layers in Figure 5.3 are translationally equivalent) can be used to simulate the streaks that appear in the diffraction pattern due to stacking faults in the layered structure (cf. Section 5.1.1). Figure 5.6 shows such a simulation: The simulated peak profile goes along the red arrow shown in the diffraction pattern to the right. The streaks appear as the intensity does not go down in between two reflections completely, which is caused by diffuse diffraction. This can be simulated with the experimental structure, taking into account stacking faults. The parameter "corr" is a measure of disorder, 0 would mean completely random stacking, corr $= -1$ would lead to alternating stacking, as after stacking one layer to the left of the mirror plane of the unit layer, the next would always go to the right.

The second type of streaks, which cannot be simulated with only one unit cell, are caused by local distortions in the layers that get averaged out during an XRD experiment. They are probably caused by non-negligible interatomic interactions across layers due to the fairly thin layers of chlorothionite. This phenomenon (of local distortion) is typical for OD structures and is called "desymmetrization" [143]. DFT can find the distorted positions via force minimization (as described above). By performing such structure optimizations on four different stacking fragments (shown in Figure 5.8 at the end of

this chapter), it is possible to extract the necessary layer geometries and subsequently successfully model the streaks caused by desymmetrization as well.

5.2.1 Computational Details

The optimization of atomic positions is among the most important tasks performed with WIEN2k. Routinely, structures have internal degrees of freedom (i.e. atoms that do not occupy positions fixed by symmetry). This is especially important, if substitutions are undertaken (and the substituent atom causes local distortions) or when a calculation is started with an experimental structure. Due to the approximations inherent in the density functionals or limitations in experimental methods leading to differences regarding the equilibrium positions. That means that after the initial calculation, (potentially large) forces acting upon the atoms are to be expected. These forces have to be minimized in order to find a ground state energy. The idea is simply to move atoms in some direction (prescribed by the forces) and recalculate the forces on the new position. This is done by using a modified version of WIEN2k's `MIXER` program (called "MSR1a" [144, 145], that moves the atoms on the fly (instead of changing the atomic positions after fully converging and starting a new SCF cycle), which is a very efficient and usually quite fast method.

For the structure optimization of the chlorothionite fragments, different k-meshes [146] had to be used due to the different unit cell dimension ($4 \times 4 \times 4$ for Fragment 1, $5 \times 7 \times 2$ for Fragment 2, and $2 \times 1 \times 3$ for Fragments 3 and 4); convergence checks with respect to the k-meshes were performed and virtually the same positions were found when the mesh size was increased. The following parameters were used for all four cases: atomic radii of 1.92 bohr for Cu, 1.86 bohr for Cl, 2.10 bohr for K, 1.34 bohr for S, and 1.27 bohr for O; $RK_{\max}$ was set to 7, and a Hubbard U of 6 eV was used to treat the delocalized 3d electrons of Cu in the LDA+U scheme as implemented in WIEN2k [65, 99]. Moreover, Cu was treated spin-polarized (with antiferromagnetic ordering). All forces were relaxed until the residual forces were below 1 mRy/bohr.

5.2.2 Results

Figure 5.9 (at the end of this chapter) showcases some examples of geometry changes due to atomic position optimization. Finally, Figure 5.7 shows a successful simulation of the peak profile along the red arrow inserted into the diffraction pattern – both simulations, with purely experimental input and with DFT-based input, are shown for comparison.

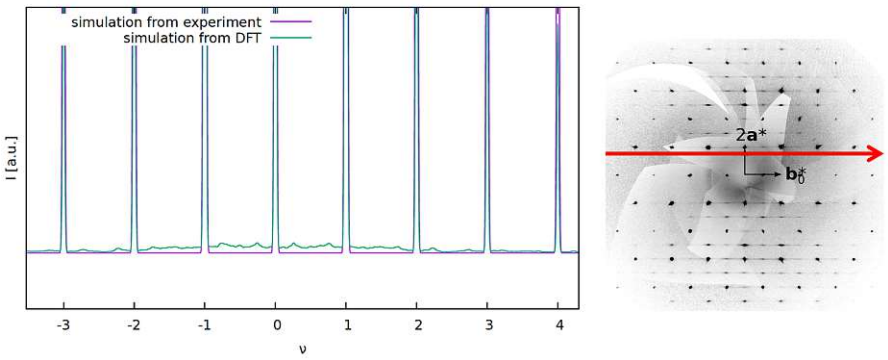

Figure 5.7: Simulation of the distortion related streaks in chlorothionite diffraction patterns. Simulated peaks appear over the red arrow in the diffraction pattern (right) going from left to right [139].

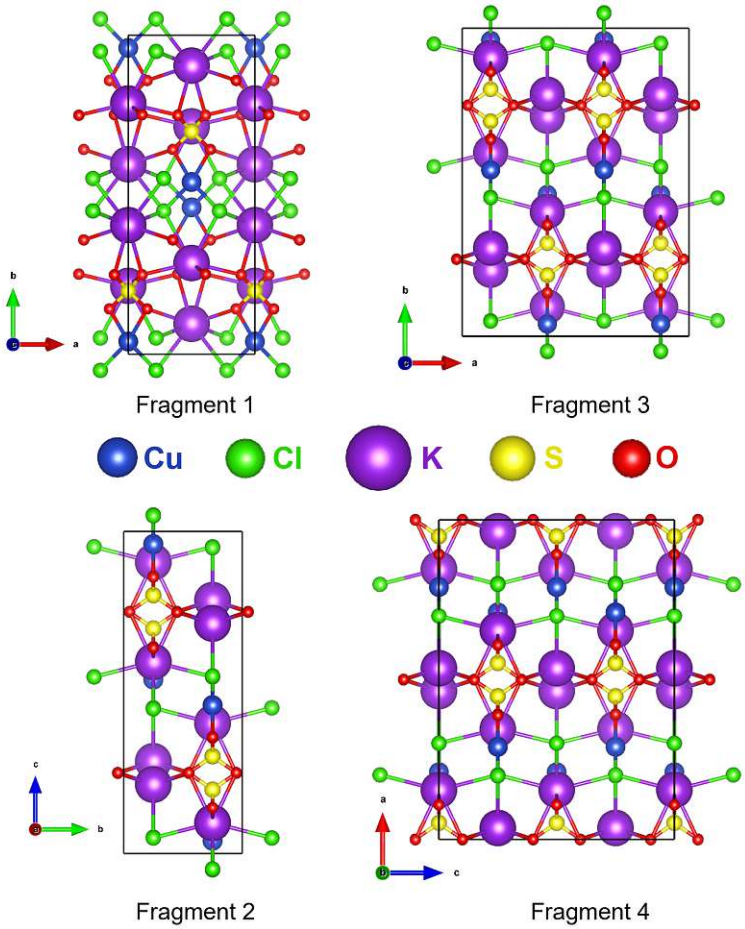

Figure 5.8: For different layer fragments (with different stacking order) for which the atomic positions were optimized.

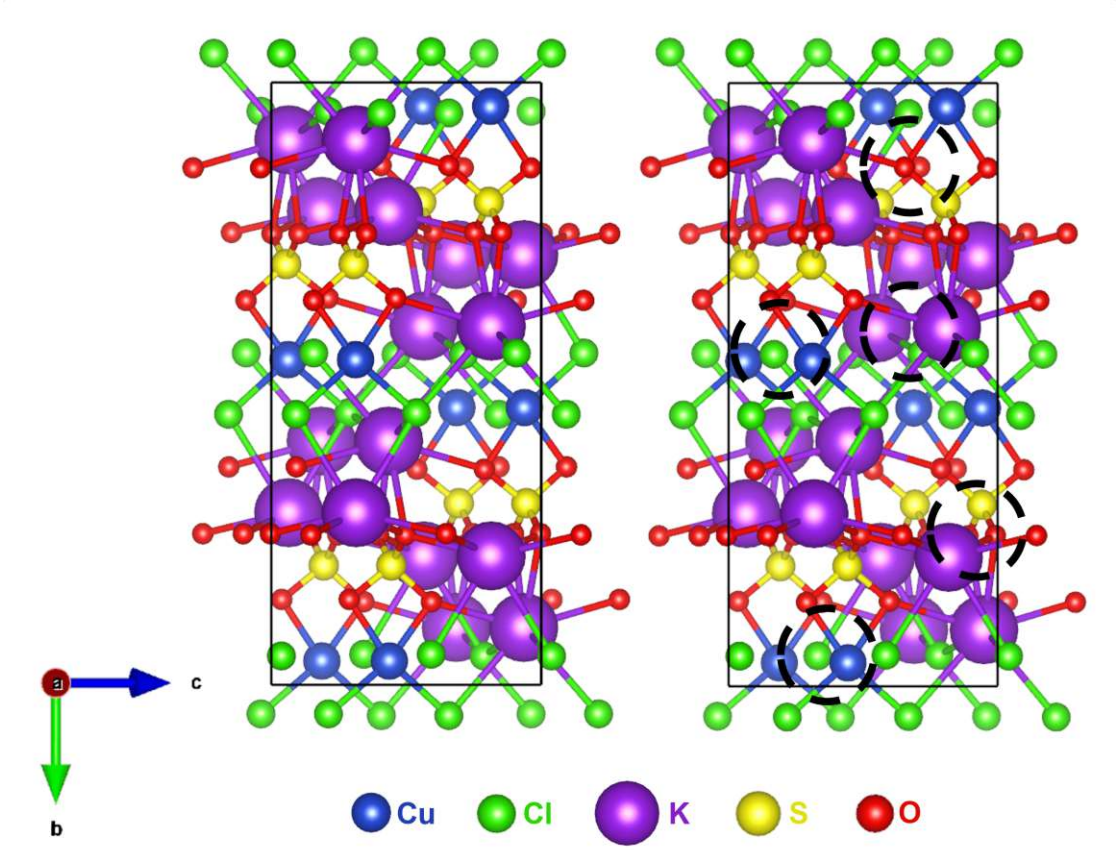

Figure 5.9: Fragment 3 before (left) and after (after) atomic position optimization. 5 examples of structural changes are highlighted (dashed circles).

CHAPTER 6

Adsorption on Surfaces

It is up to you, how "thick" (how many layers) you want to make your slab. The thicker the better, but soon you will run out of computer power.

Peter Blaha, WIEN2k Mailing List (2013-06-26)

When investigating the $MgO/Mg(OH)_2$ system as potential candidate for thermochemical energy storage, Danny Müller and coworkers found that the hydration of MgO with steam is slow and incomplete, but can be much improved by doping Ca (leading to mixed oxides $Mg_{(1-x)}Ca_xO$), as can be seen in Figure 6.1.

A reasonable hypothesis would be to assume that this has to do with adsorption of water on the respective oxide surfaces, as adsorption is the first step of the reaction of the solid and the molecules in the gas phase. Therefore, in this second use case of WIEN2k, the adsorption behavior of water on mixed (Mg,Ca)O surfaces is investigated. Particular focus is put on the question whether adsorbed water remains intact or dissociates to form hydroxyl groups.

Understanding the properties of and processes on surfaces is crucially important when studying materials in general. All interactions of solid materials with matter happen (initially) at the surface. Adsorption on surfaces plays a particularly important role for these interactions, as properties of both the solid as well as the adsorbed species change due to adsorption. Such changes could affect, for instance, the electronic structure and thereby chemical bonding thus making a material catalytically active (e.g. in heterogeneous catalysis).

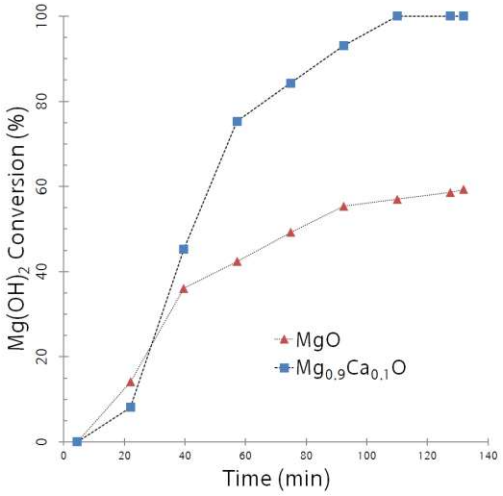

Figure 6.1: Hydration of MgO and Ca-doped MgO with water vapor. Pure MgO (red triangles) reacts slowly and incompletely, Ca-doping (here 10 %) enables complete and fast conversion (blue squares). Courtesy of Danny Müller.

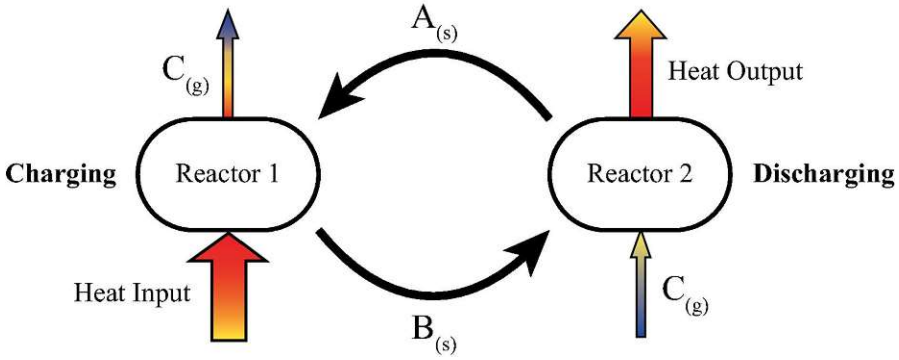

Figure 6.2: Schematic representation of the principle of thermochemical energy storage using a reversible chemical reaction $A_{(s)} \rightleftharpoons B_{(s)} + C_{(g)}$: During charging, solid A is decomposed with waste heat in reactor 1, which leads to the creation of solid B and gas C. B can then be stored until heat is needed or transported. During discharging, B is reacted with C, releasing the stored energy and recovering solid A that can be used in a next cycle. Reprinted from [147], © (2016) with permission from Elsevier.

6.1 Thermochemical Energy Storage

Thermochemical energy storage is a promising approach to address the issue of energy loss via waste heat [148].

The fundamental principle – utilizing reversible reactions to turn heat into chemical energy and vice versa – is simple yet elegant (cf. schematic representation in Figure 6.2): A solid material A is decomposed into two products (another solid B and gas C), thus storing the heat of the reaction in the two products. This process is called charging. B can then be transported and/or stored until energy is needed. Then, during discharging, B can react with C to form A again, while releasing the previously stored energy.

$$A_{(s)} \rightleftharpoons B_{(s)} + C_{(g)} \tag{6.1}$$

The choice of system depends on the amount of waste heat to be utilized; however, a large number of systems over a wide charging temperature range are studied for potential usage [147], making thermochemical energy storage highly flexible and adaptable.

6.2 Simulating Mixed-Oxide Surfaces

Simulating surfaces using a DFT code with periodic boundary conditions requires the setup of special unit cells: In principle, a surface is a two-dimensional defect that terminates a solid in one direction – that means periodicity remains intact parallel to the surface. To model such a system, the unit cell of the bulk material (in this case MgO or CaO-doped MgO) is used as basis to create a supercell with a vacuum layer along one lattice vector. The thickness of the vacuum layer has to be simultaneously large enough that neighboring cannot interact and small enough that computational cost remains reasonable. The vacuum layer necessitates larger basis sets; and the thicker the vacuum layer is the more APWs are needed. Thus, modeling surfaces greatly benefits of MPI parallelization – in fact, in larger cases it is actually inevitable as memory requirements grow.

MgO (in a cubic, rock salt structure with $a_{\text{exp}} = 4.214\,\text{Å}$ [149]) and CaO (cubic, rock salt structure, $a_{\text{exp}} = 4.8152\,\text{Å}$ [150]) were used as a starting point to determine DFT lattice parameters and to create surface slabs. The slabs used for the presented calculations (shown in Figure 6.3) have 5 MgO layers (with 40 atoms in per unit cell) along the c-direction and 15 Å between the slabs. This is achieved by first setting up a $1 \times 1 \times 2$ supercell to get the 5 layers perpendicular to the lattice vector c; the vacuum layer can be added at this point already. Finally, a $\left(\sqrt{2} \times \sqrt{2}\right) R45$ cell was created using the 5-layer model with vacuum. According to this notation, the final cell has new lattice vectors a' and b' that are a factor of $\sqrt{2}$ larger than a and b, additionally, both vectors are rotated by 45°. In other words, the lattice vectors are rotated into the diagonals of the original cell and scaled.

Surface slab models for pure MgO (as reference), $Mg_0.9Ca_{0.1}O$, and $Mg_0.8Ca_0.2O$ were set up. Since it seems reasonable to assume that changes in the electronic structure

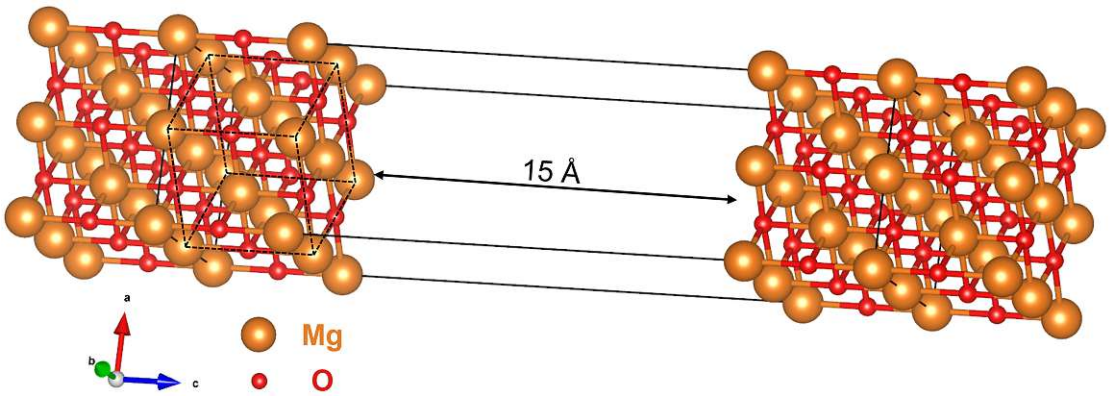

Figure 6.3: Surface slab model of with 5 layers of MgO with inversion symmetry and 15 Å of vacuum between neighboring slabs. Shown are two slabs – the supercell is set up in way, that the vacuum is "in the middle", i.e. the unit cell contains half a slab at small coordinates of the lattice vector perpendicular to the surface, and another half slab at large coordinates. The original cubic unit cell that was used to create the slab is drawn with dashed black lines.

of the surface caused by Ca-doping would also affect the adsorption behavior of water, the Ca atoms were placed at the surface. Vegard's law (Equation 6.2) [151] was used to estimate the expanded lattice parameters due to mixing, which is expected to occur due to the lattice mismatch of the constituent oxides (the DFT lattice parameters were determined to be $a_{\mathrm{MgO}} = 4.26\,\text{Å}$ and $a_{\mathrm{CaO}} = 4.84\,\text{Å}$, respectively).

$$a_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x\mathrm{O}} = (1-x)\, a_{\mathrm{MgO}} + x a_{\mathrm{CaO}}\,, \tag{6.2}$$

where $a_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x\mathrm{O}}$, a_{MgO}, and a_{CaO} are the cubic lattice constants of the mixed and pure constituent oxides $(\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x)\mathrm{O}$, MgO, and CaO, respectively. x is the fractional content of CaO.

Another possible cause of the changed hydration behavior by doping might be related to the expansion of the lattice (via introducing larger Ca ions): To investigate the influence of size effects, all three types of slab were simulated within a range of different lattice parameters (the respective Vegard values for $x = 0$ to $x = 0.5$).

As first step, the surface energies ΔE_{surf} for all surface models were compared. The surface energy is a measure for the energy cost per unit area that is needed to create the surface. It can be calculated according to:

$$\Delta E_{\mathrm{surf}} = \frac{1}{2A}\left(E^{\mathrm{total}}_{\mathrm{slab}} - n_{\mathrm{fu}} E^{\mathrm{total}}_{\mathrm{bulk}}\right)\,, \tag{6.3}$$

where A is the unit surface area, $E^{\mathrm{total}}_{\mathrm{slab}}$ is the total energy of the surface slab, $E^{\mathrm{total}}_{\mathrm{bulk}}$ is the total energy per formula unit of the bulk material, and n_{fu} gives the number of formula units contained in the slab. The factor $\frac{1}{2}$ arises due to the fact that a surface slab has

two surfaces. In mixed surface slabs, the second term of the difference is repeated for all species occurring in the mixture.

The following computational parameters were used: For the surface slabs, a $2 \times 2 \times 1$ Monkhorst-Pack k-mesh [146] was chosen for Brillouin zone sampling. The R_{MT} values of Mg/Ca, O, and H were set to 1.7 bohr, 1.1 bohr, and 0.55 bohr, respectively. An RK_{max} value of 3.5 was used. For the bulk calculations, a denser k-mesh ($5 \times 5 \times 5$) was sampled, the R_{MT} values remained the same and an RK_{max} of 8 was chosen. All structures were relaxed to residual forces of below 1 mRy/bohr.

Figure 6.4 displays the calculated surface energies. The pure MgO (001) surface with its equilibrium lattice parameter gives a value of about 55 meV/Å^2, which agrees with values found in literature [152, 153][i]. For larger values of a, the surface energy increases, which is to be expected, as the slab moves away from the equilibrium lattice parameter. The $Mg_{0.9}Ca_{0.1}O$ surfaces behave similarly, even though the surface energy increases more slowly. Furthermore, the values at lattice parameters corresponding to $x = 0$, $x = 0.05$, and $x = 0.1$ lie within 1.1 meV/Å^2 of each other. The progression of ΔE_{surf} in the case of $Mg_{0.8}Ca_{0.2}O$ matches expectations as well: The lattice parameter of the undoped MgO surface deviates by more than 1 %, while the minimum of ΔE_{surf} appears at the equilibrium value of a according to Vegard's law.

In the second step, water molecules were adsorbed on both side of slab (to keep inversion symmetry and to avoid polarization across the slab). According to Hu et al., mainly four adsorption structures of water occur on (001) surfaces of MgO and CaO (Figure 6.5) that are mediated by hydrogen bonds [154]: In the Type I adsorption structure, the water is oriented planar with respect to the surface; the O atom sits above a cation site and the O-H-bonds point in the direction of the cation-anion bonds of the oxide. This is energetically the most favorable orientation on MgO. Type II has the water oxygen above a hollow site, while the H atoms point downwards in the direction of oxygen sites. This is the most stable orientation of CaO. Type III (only one O-H-bond oriented towards the surface with the other pointing away from the surface) is reported to be the most stable for fluorides (and was, therefore, not considered in this work). Lastly, the Type IV adsorption structure is the preferred configuration for dissociated water – one hydroxyl is located above and slightly off-center of a hollow cite, while the second H atom forms a second hydroxyl with a lattice O.

From the total energies of these surface slabs, the adsorption energy of water on the surface can be calculated as follows:

$$E_{\mathrm{ads,H_2O}} = \frac{1}{n_{\mathrm{ads,H_2O}}} \left[E^{\mathrm{total}}_{\mathrm{slab+H_2O}} - E^{\mathrm{total}}_{\mathrm{slab}} - n_{\mathrm{ads,H_2O}} E^{\mathrm{total}}_{\mathrm{H_2O}} \right] , \tag{6.4}$$

where $E_{\mathrm{ads,H_2O}}$ is the adsorption energy of a water molecule, $E^{\mathrm{total}}_{\mathrm{slab+H_2O}}$ and $E^{\mathrm{total}}_{\mathrm{slab}}$ are the total energies of the surface slab with and without adsorbed water, and $E^{\mathrm{total}}_{\mathrm{H_2O}}$ is the

[i] The cited sources give values of 0.90 J m^{-2} and 0.92 J m^{-2}, respectively. Converting the value found in this work results in 0.88 J m^{-2}.

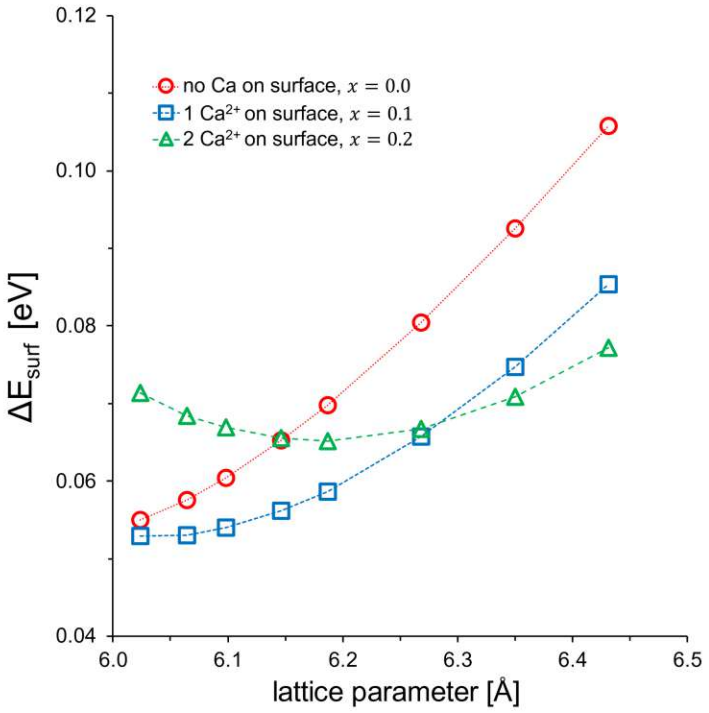

Figure 6.4: Surface energies for MgO, $Mg_{0.9}Ca_{0.1}O$, and $Mg_{0.8}Ca_{0.2}O$ as function of lattice parameter a.

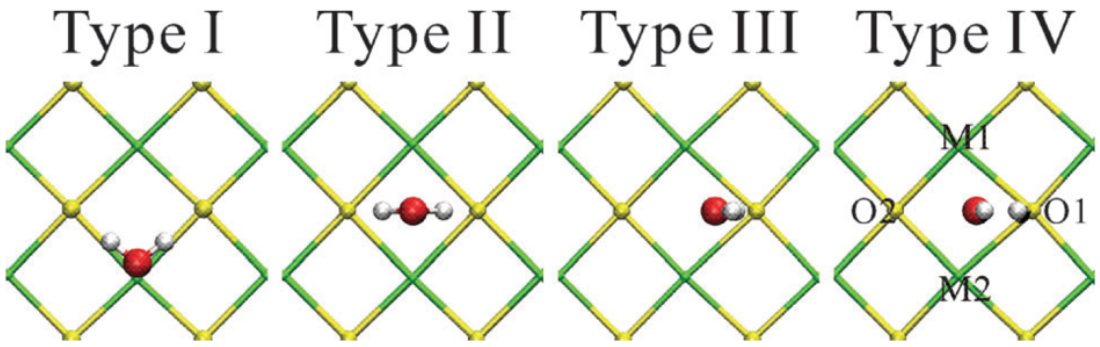

Figure 6.5: Different possible adsorption structures of water molecules on binary cubic materials. Used with permission of the Royal Society of Chemistry. Reprinted from [154].

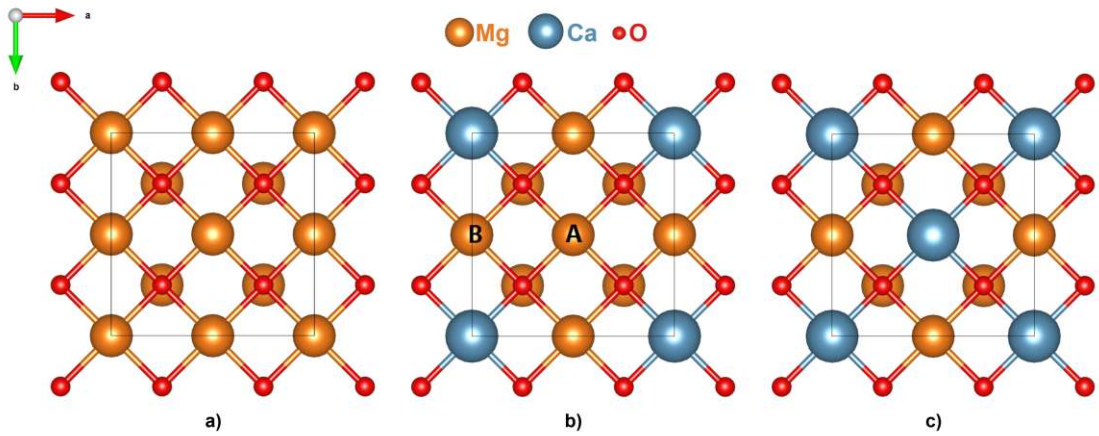

Figure 6.6: Top view of the three slab models. a) MgO (001) surface with four equivalent adsorption sites. b) $Mg_{0.9}Ca_{0.1}O$ (001) surface with three different adsorption sites: one Ca site in the corner, two equivalent Mg sites B on the edges, and one Mg site in the center of the unit area. c) $Mg_0.8Ca_{0.2}O$ (001) surface with two pairwise equivalent adsorption sites (Ca sites in the corners and in the center, Mg sites on the edges).

total energy of a water molecule[ii]. n_{ads,H_2O} denotes the number of adsorbed molecules. The computational parameters remained the same as for all other slab calculations.

To obtain values for $E^{\mathrm{total}}_{\mathrm{slab}+H_2O}$ with intact water, a molecule was adsorbed in the geometries of Type I and Type II. Dissociation was triggered "manually": After the adsorption structure was relaxed, one hydrogen was moved to a neighboring lattice oxygen and the simulation re-run. Figure 6.6 shows the top view of the unit areas of the surface slabs used here: All slabs have 4 cationic adsorption sites, one and two of which are occupied by Ca instead of Mg in case of $Mg_{0.9}Ca_{0.1}O$ and $Mg_{0.8}Ca_{0.2}O$, respectively. In case of MgO, all those sites are equivalent by symmetry (in case of mono-molecular adsorption); in case of $Mg_{0.9}Ca_{0.1}O$, two different Mg sites occur (a twofold site on the edges of the square, and the center atom (labeled "B" and "A" in Figure 6.6, respectively); in case of $Mg_{0.8}Ca_{0.2}O$, the two Mg sites and the two Ca sites are equivalent.

6.3 Results[iii]

For the MgO surface with the correct lattice parameter the reported behavior [154] could be reproduced. The Type I structure is energetically favored, while Type II is stable but less favorable. Type IV was not observed on MgO (the hydrogen which was split off manually, recombined with the hydroxyl group to re-form molecular water). All other adsorption calculations were started from structure Type I for simplicity.

[ii] For consistency, this energy is also calculated with WIEN2k, where a single water molecule sits in a large "box" to make sure it does not interact with its periodic images.

[iii] The results presented here have been published in [155].

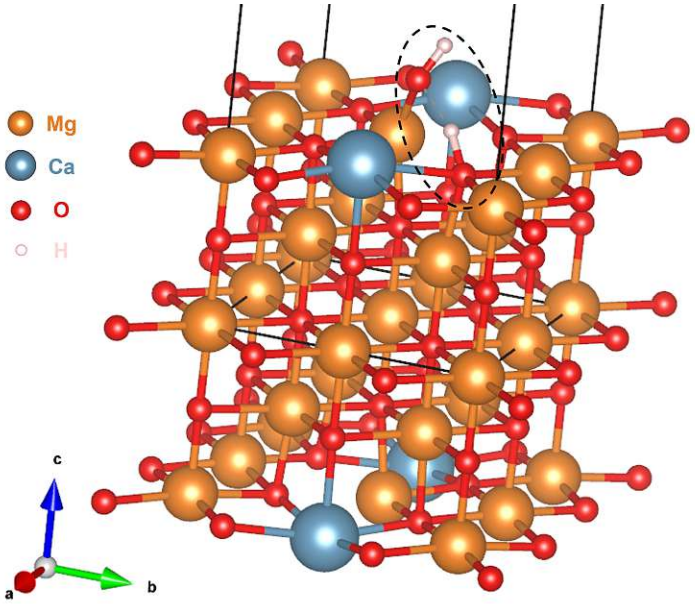

Figure 6.7: Surface slab of $(Mg_{0.9}Ca_{0.1})O$ with one adsorbed and dissociated water molecule: H_2O was initially adsorbed on top of a Mg atom, relaxed, and manually split (i.e. one H was moved to a bulk O). After a second relaxation, the adsorbed H_2O remained dissociated – the newly formed OH-groups can be seen within the dashed black ellipse.

Increasing the lattice parameter moves the optimal adsorption structure away from Type I – during relaxation the oxygen atom moves towards the hollow site and the whole water molecule starts to tilt. Moreover, hydroxyl groups are stable even on a pure MgO surface for the two largest values of a (and for the largest cell, the hydroxyls are energetically favorable). This confirms the initial assumption that size effects (due to lattice expansion) influence adsorption, as the expansion of the lattice was caused "manually" and no other effects come into play.

In case of both Ca-doped surfaces, stable hydroxyl structures were observed over the whole range of lattice parameters – Figure 6.7 shows an example: The adsorbed oxygen is located above a hollow site and one of the hydrogen atoms has formed a hydroxyl group with a lattice O. This confirms the second initial assumption of the electronic structure of the surface having an effect as well.

These electronic effects of Ca doping become even more apparent when the most stable adsorbed species is plotted against the lattice parameter (see Figure 6.8): For pure MgO (red curve), intact water (full symbols) is the most stable configuration over almost the whole size range – only at the very largest slab, hydroxyl groups are more stable (open symbols). The point at which this switch between preferential adsorption of molecular and dissociated water occurs shifts towards smaller sizes with increasing Ca content: In

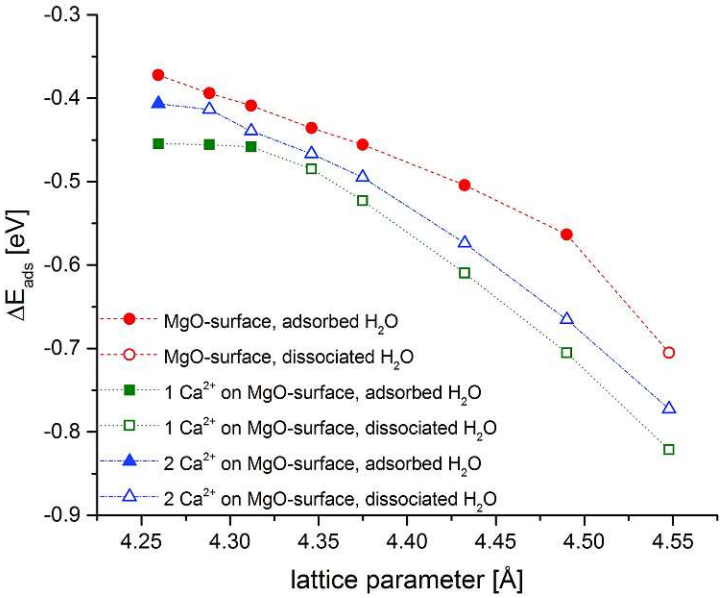

Figure 6.8: Stability of adsorbed water on mixed (Mg,Ca)O surfaces: Full symbols denote adsorbed (but not dissociated) water molecules are more stable, empty symbols denote the slabs, where dissociation was energetically favorable and two OH groups occupy the surface after adsorption. Reprinted from [155], © (2017) with permission from WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim.

case of the surface with one Ca site – $Mg_{0.9}Ca_{0.1}O$ – this happens at the lattice parameter nominally corresponding to $x = 0.15$ according to Vegard's law. For $Mg_{0.8}Ca_{0.2}O$, the switch happens at even smaller unit cells – only the smallest slab (with the lattice parameter of a pure MgO surface) dissociation of the adsorbed water is less favorable.

Additionally, the DFT formation energetics of mixed (Mg,Ca)O phases were modeled utilizing a regular solution model as suggested by Peelaers et al. [156]: The model given in Equation 6.5 assumes a parabolic relation between the Ca content x and the enthalpy of formation $\Delta_f H$.

$$\Delta_f H_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x\mathrm{O}} = 4x(1-x)\Delta H_0 \tag{6.5}$$

To find the parameter ΔH_0, the formation enthalpies of mixed oxides with Ca contents from 0 to 100 % had to be calculated. A primitive $2 \times 2 \times 2$ supercell of MgO was used as starting point (a larger $3 \times 3 \times 3$ cell was used for the smallest Ca content) in which random Mg atoms were substituted for Ca. These random mixed structures were created with the program "supercell" [157] – the number of substitutions was prescribed by the target Ca content. For the calculation of $\Delta_f H_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x\mathrm{O}}$, R_{MT} radii of 1.7 bohr for all atoms were chosen. A k-mesh of $5 \times 5 \times 5$ and a basis set determined by $R_{\mathrm{MT}}^{\mathrm{min}} K_{\mathrm{max}} = 8$ were used. The lattice parameters of the respective mixed oxides were optimized (the starting value again determined according to Vegard's law) and all structures were relaxed (until residual forces below 1 mRy/bohr were reached). $\Delta_f H_{(\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x)\mathrm{O}}$ was then derived

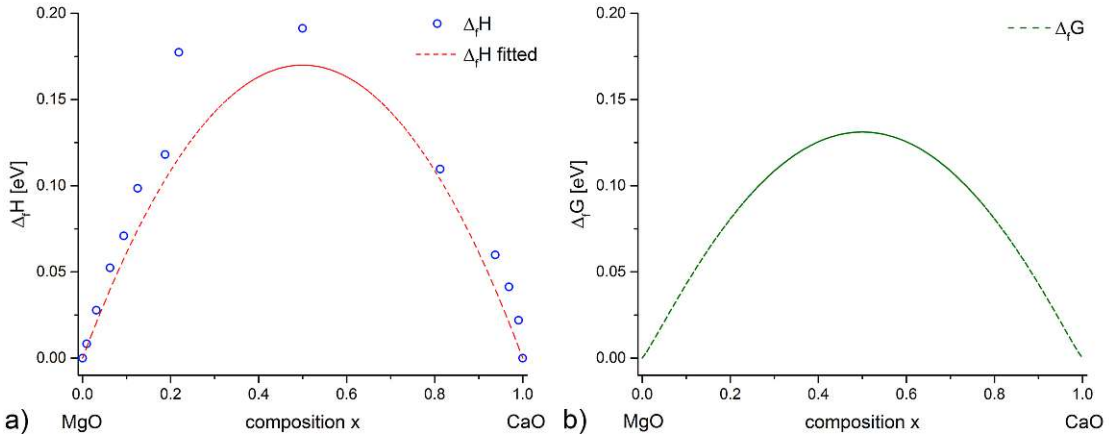

Figure 6.9: a) Simulated enthalpy of formation ($\Delta_f H$) as laid out by [156]: $\Delta_f H$ was calculated for randomly generated mixed (Mg,Ca)O structures with varying Ca content (blue circles). These values were parabolically fitted (see Equation 6.5) such, that all of them lie on or above the fit parabola. A $\Delta_f H_0$ of 170 meV per unit formula. b) $\Delta_f G$ as function of x at 650 K, the entropic contribution was estimated using Equation 6.7. Reprinted from [155], © (2017) with permission from WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim.

according to:

$$\Delta_f H_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x\mathrm{O}} = E^{\mathrm{total}}_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x\mathrm{O}} - (1-x)\, E^{\mathrm{total}}_{\mathrm{MgO}} - x E^{\mathrm{total}}_{\mathrm{CacO}}\,, \tag{6.6}$$

where $E^{\mathrm{total}}_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x)\mathrm{O}}$ is the energy of the mixed oxide and $E^{\mathrm{total}}_{\mathrm{MgO/CaO}}$ are the respective energies of the bulk oxides.

The calculated values for $\Delta_f H_{\mathrm{Mg}_{(1-x)}\mathrm{Ca}_x)\mathrm{O}}$ were fitted such that all enthalpies fall on or above the fit – they are only upper bounds, as the randomly chosen configuration is not necessarily the energetically most favorable (see Figure 6.9a). Thus, a value for ΔH_0 of 170 meV per unit formula was obtained. To account for entropic effects as well, the mixing entropy was modeled with the entropy of a random mixture according to:

$$S(x) = -k_{\mathrm{B}}\left[x \ln x + (1-x)\ln(1-x)\right]\,, \tag{6.7}$$

where k_{B} is Boltzmann's constant. Using the values for $\Delta H(x)$ and $S(x)$ the Gibbs free energy can be calculated as function of x for a given temperature $T = 650\,\mathrm{K}$ according to

$$\Delta_f G(x) = \Delta H(x) - TS(x)\,. \tag{6.8}$$

The obtained $\Delta_f G$ as function of x is shown in Figure 6.9b. As both $\Delta_f H$ and $\Delta_f G$ are positive over the whole range of x, MgO and CaO should not be miscible according to thermodynamics at 650 K, which agrees with experimental findings in literature [158].

Müller et al. [155] propose the origin of materials as reason for the existence of mixed (Mg,Ca)O phases: They are synthesized via co-precipitation of hydroxides and subsequent calcination at mild temperatures (around 650 K). For the hydroxides, the authors find miscibility up to 10 % $Ca(OH)_2$ (which was confirmed by XRD). Thus, DFT results and experimental findings are in agreement that Ca-rich (Mg,Ca)O phases have to be formed via the detour of hydroxides for thermodynamic reasons.

CHAPTER 7

Error Estimation in DFT Codes – The Delta-Project

+++ Divide By Cucumber Error. Please Reinstall Universe And Reboot +++

Terry Pratchett, Hogfather (1997)

Solid-state DFT codes have been in use for decades and continue to be improved and further developed. In recent years, they are gaining more and more traction as work horses in condensed matter physics and materials science [159], and majorly contribute to large-scale material research oriented data collection projects like the Materials Project[i] [160] and the NOMAD Laboratory[ii] [161]. However, systematic assessments with respect to reliability and precision of DFT have not been done until fairly recently. Such assessments are particularly important, because even when codes implement the same approximations (e.g. KS-DFT with a given density functional), the specifics still vary (for instance with respect to basis sets or treatment of core electrons). The Delta Project[iii] was among the first projects to attempt such an assessment on a larger scale.

7.1 The Original Delta-Benchmark

In 2013, Lejaeghere et al. proposed a metric to quantify the agreement between different electron codes [162], that was subsequently used in their seminal work in 2016 [163], in which they compared 40 different codes (including all-electron, PAW, and pseudopotential codes). The test set then consisted of 71 experimentally known crystals.

[i] https://materialsproject.org
[ii] https://nomad-lab.eu
[iii] https://molmod.ugent.be/deltacodesdft

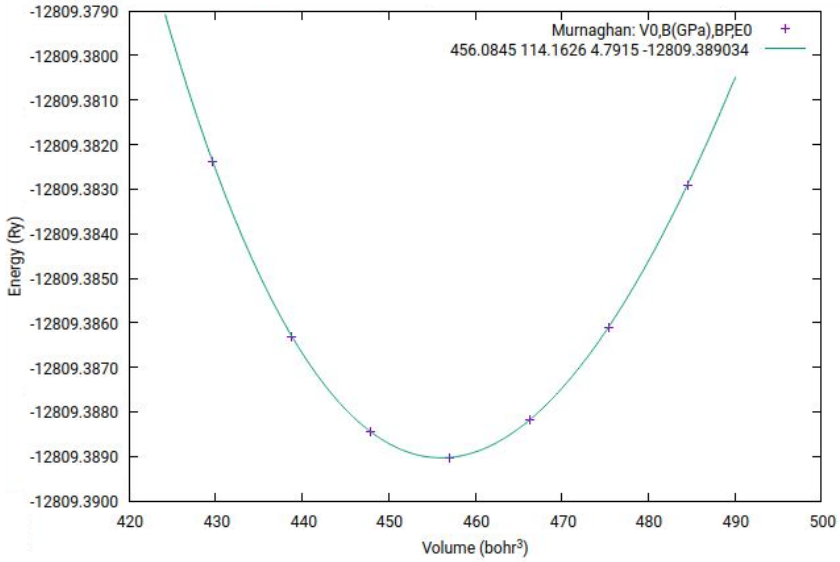

Figure 7.1: Example of an energy-volume curve, shown is the curve of SnO_3. The plus symbols denote the measured energies. The fit was obtained from the Birch-Murnaghan density of state.

The metric Δ is defined according to Equation 7.1 and compares energy-volume curves in a pre-defined volume range (going from $-6\,\%$ to $6\,\%$) with 7 calculated energies in equidistant steps of $2\,\%$. The smaller the value of Δ is, the better the agreement between two codes.

$$\Delta_{(\mathrm{a,b})} = \sqrt{\frac{1}{0.12V_0}\int_{0.94V_0}^{1.06V_0} \left[E_a(V) - E_b(V)\right]^2 dV}\,, \tag{7.1}$$

where V_0 is the equilibrium volume obtained by fitting a energy-volume curve with an equation of state that – in case of the Delta approach, the Birch-Murnaghan [164] equation of states is used.

$$E(V) = E_0 + \frac{9}{16}V_0B_0\left\{\left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^3 B_1 + \left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^2 \left[6 - 4\left(\frac{V_0}{V}\right)^{\frac{2}{3}}\right]\right\}, \tag{7.2}$$

where V_0 and B_0 are the equilibrium volume and bulk modulus, respectively, and B_1 is the first derivative of B_0 with respect to pressure. Figure 7.1 shows an example of an energy-volume curve (here SnO_3 is shown).

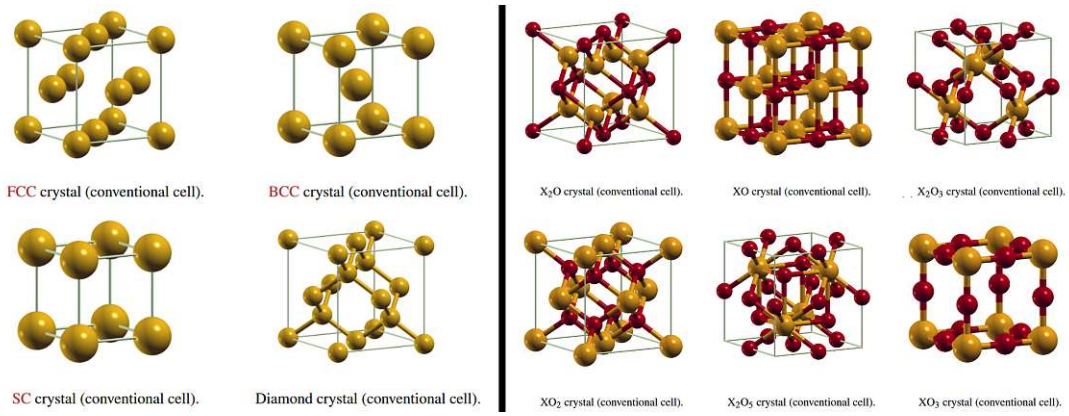

Figure 7.2: Test set structures. (left) Elemental crystals to account for different coordination numbers (tier 1). (right) Oxide structures to account for different chemical environments (different "oxidation numbers" of the cation from +1 to +6, tier 2). Reproduced from [159] under CC BY 4.0.

7.2 Extending the Test Set – Elemental Crystals and Oxides

Another interesting issue (beyond simple agreement of codes) is whether pseudopotentials are transferable to different solids. To study this question further, more comprehensive test sets are necessary to systematically test different pseudopotentials. In a recently submitted study [159], an extended test set of 960 structures in total, was proposed. It consists of two subsets: the tier 1 set containing elemental crystals to systematically account for different coordination numbers, and the tier 2 set to do the same for oxidation numbers (Figure 7.2 shows the used test set structures). Moreover, a reference set of high-accuracy (i.e. highly converged) were obtained from WIEN2k and Fleur [165], two all-electron codes. Calculations of the 570 tier 2 crystals, that were used as starting point for the WIEN2k references were conducted as part of this thesis.

Results produced by 9 pseudopotential codes were compared to the all-electron reference [159]. A visual summary of these comparisons is given in Figure 7.3: The all-electron codes WIEN2k and Fleur show excellent agreement with each other over the whole test set. The pseudopotential codes, on the other hand, show significant deviations. Moreover, not all elements of the test set could be calculated with all codes, as the corresponding pseudopotentials are not available in all codes (for instance due to difficulties in describing localized f-states). Some codes implemented systematic improvements to the underlying pseudopotentials (which shall not be discussed here) during this study and achieved improvements by adding new or adapting existing pseudopotentials. This highlights the importance of systematic studies like the one presented by Bosoni et al., as they advance progress of electronic structure calculations in general.

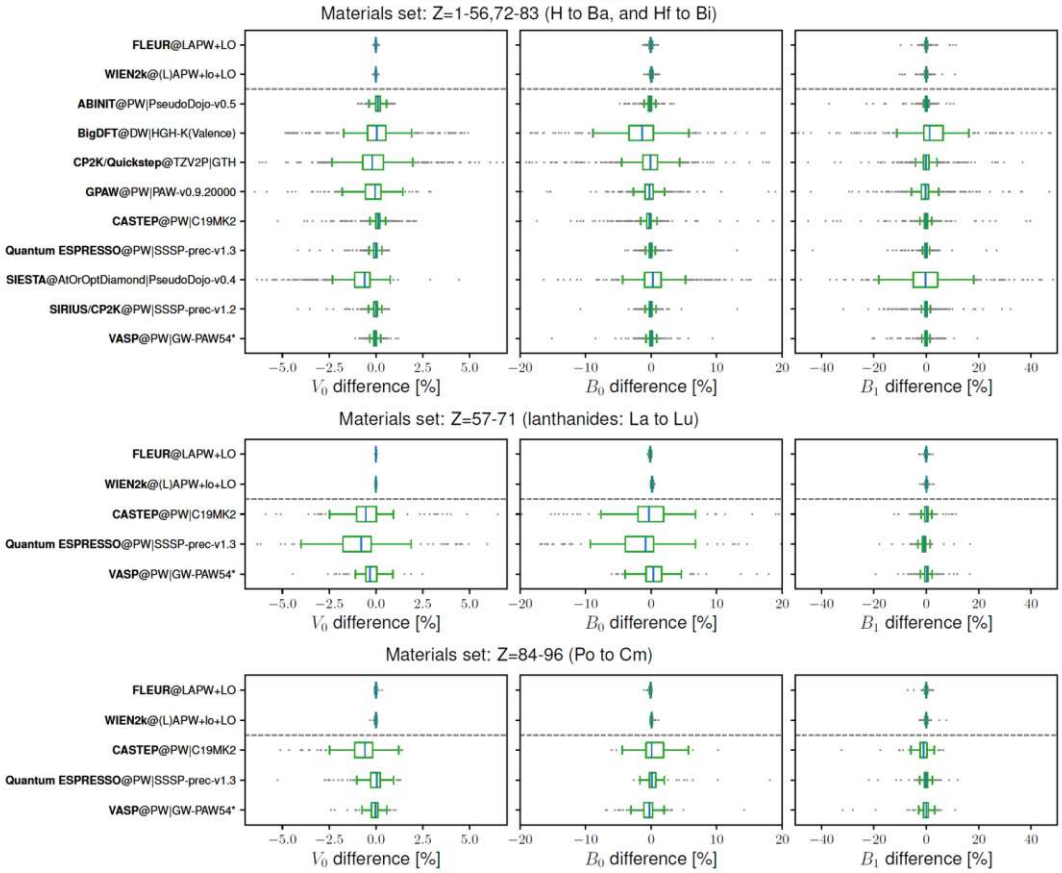

Figure 7.3: Results of a comparison of 9 pseudopotential codes with the reference test set presented in [159]. Box-and-whisker plots show the calculated deviations of V_0, B_0, and B_1 from the reference data set. Reproduced from [159] under CC BY 4.0.

7.3 Extending the Test Set Further – Real Binaries

One issue with the test set layout as described so far is the fact that while coordination numbers (tier 1) and oxidation numbers (tier 2) are covered extensively and systematically, a systematic treatment of chemical bonds is still missing. Moreover, most of the tier 1 and tier 2 test cases are purely hypothetical structures (only a little over 10 % of the oxides are known experimentally). Therefore, the preliminary work of creating a third tier was done as part of the work for this thesis. This included the following steps:

- scanning the **I**norganic **C**rystal **S**tructure **D**atabase (ICSD) [iv] [166] for suitable candidates of experimentally known cubic binary crystals without internal degrees of freedom (i.e. all atomic positions are fixed by symmetry),

[iv] www.https://icsd.fiz-karlsruhe.de

- setting up a classification system and find as objective criteria as possible to classify the candidates (which was iterated a couple of times during the process),
- classifying the candidate crystals, and
- selecting test cases (5 per element.)

As of January 2018, the ICSD contained experimental structures of 32 163 binaries, 11 840 of which were cubic. After the removal of duplicates, 1457 candidates in 23 different structure types[v] remained to be classified.

The classification system is based on electronic difference densities: The superposed densities of free atoms filling the lattice was subtracted from the calculated crystal density. The main idea was to have test set members that represent the prototypical bond types ionic, covalent, and metallic, and include a possible band gap as additional criterion. According to these specifications, every element should occur in 5 binaries. The finalized assessment criteria were as follows:

- A binary was defined as "ionic" if the following criteria were met:
  - area around the atom in the difference density plots almost spherical
  - interstitial region mostly shows electron depletion or neutral
  - both atoms with electron accumulation and depletion present
- The criteria for a "metallic" binary were:
  - area around the atoms rather spherical
  - large portions of interstitial with electron accumulation
  - both atom types electronically depleted
- A "covalent" binary exhibited clearly visible localized electronic accumulation/depletion between the atoms (in bond directions).
- If there are features of more than one of the other three classes, the crystal will be considered "mixed".

Figure 7.4 shows three examples of difference density plots: Electrons are accumulated in red areas (positive values) and depleted in blue areas (negative values). Shown are (from left to right): CsI (rock salt structure, "ionic"), MnZn (CsCl structure), "metallic", and $AuSb_3$ ($AuSb_3$ structure type, "covalent").

After all 1457 binaries were manually classified, the test set members were chosen. The goal was to choose as few crystals as possible while containing as many elements with 4 or 5 different bond types as possible. In the end, 241 cubic binary structures were chosen to generate tier 3. Figure 7.5 shows how often each element is represented.

[v] The 5 most common structure types are (in that order) rock salt, CsCl, Laves, $AuCu_3$, and fluorite.

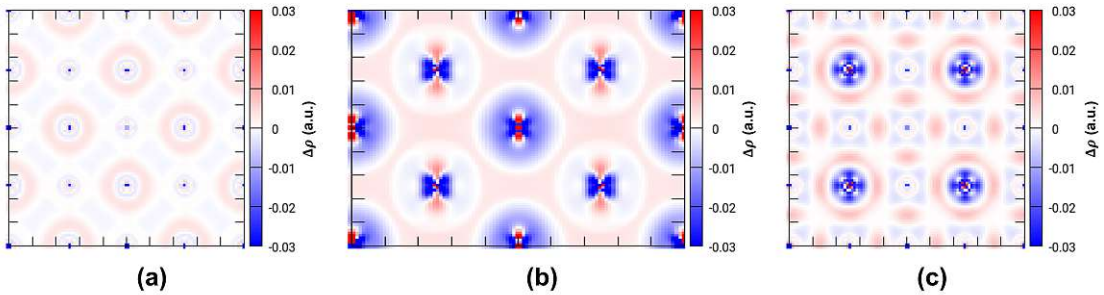

Figure 7.4: Difference density plots of three prototypical cases: (a) CsI (rock salt structure, (100) plane) as representative of ionic binaries, (b) MnZn (CsCl structure, (110) plane) as representative of metallic binaries, (c) $AuSb_3$ ($AuSb_3$ structure, (100) plane) as representative of covalent binaries. Red areas exhibit electron accumulation compared to the density free atoms sitting on lattice sites would induce, blue areas are electron depleted.

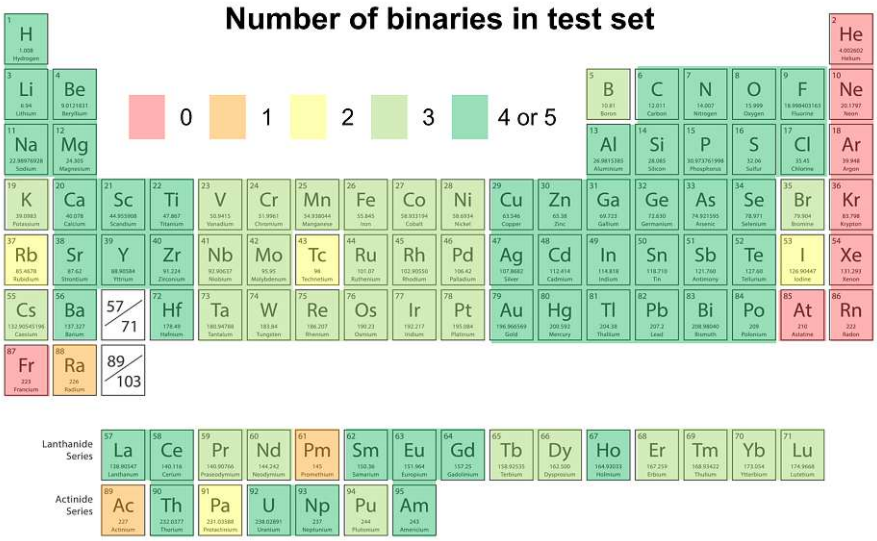

Figure 7.5: Elemental map of the number of binaries contained in the binary testset per element

Only 7 elements could not be included: At, Fr, and the noble gases. Ra, Pm, and Ac could be included only once; Rb, Tc, I, and Pa only twice, and all other elements are contained at least three times in the tier 3 test set.

CHAPTER 8

Conclusion

"The Answer to the Great Question ... Of Life, the Universe and Everything ... Is ... Forty-two," said Deep Thought, with infinite majesty and calm.

Douglas Adams, The Hitchhiker's Guide to the Galaxy (1979)

In this thesis, the performance of the DFT code WIEN2k has been assessed with a focus on the solution of the general eigenvalue problem, which is needed to obtain energies of the electronic states and, subsequently, ground state energies modeled systems.

Two questions regarding performance of the eigensolvers in WIEN2k were investigated:

1. How can the eigensolving step (as part of the program `LAPW1`) perform efficiently in parallel?
2. How does the (sequential) performance change, if not only the lowest eigenvalues (about 10 to 15 %) are needed but all of them?

To answer the first question, a variety of computational parameters necessary to set up parallel calculations (e.g. number of processors, block size of the matrix distribution, shape of the processor grid) were tested and shown to have at least some influence. Moreover, testing alternative algorithms (e.g. ELPA) and using the faster ones is always a possible approach to increase parallel efficiency.

The answer to the second question was quite clearly that the default algorithm used in sequential calculations is not suited to calculate all eigenvalues for larger matrices. Alternative algorithms have been implemented as replacement for such calculations.

However, the quest for faster and/or more efficient parallel computations does not stop, as for instance "old" algorithms and libraries are continuously developed and improved (both

(sca)LAPACK and ELPA continue to evolve) and entirely new methods and algorithms become available.

Furthermore, a new and much faster method (`3DDENS`) to obtain electronic density data on a 3D grid has been implemented and benchmarked against a small yet diverse set of crystal structures to excellent results: The time to finish setting up the grid was reduced by more than 80 % in all but one cases, the most extreme example was a reduction of 99.8 % for a molecule in a box.

The last three chapters of the thesis dealt with three different use cases, giving three examples of tasks that can be performed with WIEN2k. Beyond that, they served to showcase how different problem sizes profit from different parallelization schemes:

1. The calculation of optimized atomic positions, which cannot be found reliably by experiment (XRD in this particular case), is a simple but important application of solid state DFT in general.

   In this case, input data for a simulation of XRD diffraction patterns of an OD structure (chlorothionite) was created, which is a layered mineral that shows a large amount of stacking disorder. To obtain this data, the atomic positions in four fragments of different stackings needed to be optimized. Subsequent simulations of the diffraction pattern showed that input based on the DFT data succeeded to capture the proper streak behavior observed in experiment. A publication of the results is in preparation [139].

   The calculations for this use case were performed k-point and MPI parallel: As the unit cells of the stacking fragments were relatively small (with 10 to 20 atoms per cell), a larger number of k-points was needed for Brillouin zone sampling (hence the k-point parallelism), and the associated matrices in `LAPW1` were just large enough to run every k-point on 16 cores.

2. Studying the adsorption behavior of small molecules on surfaces is another prototypical example for DFT calculations – especially in connection with materials science.

   The example study presented investigated the adsorption behavior of water molecules on mixed magnesium/calcium oxide surfaces in order to shed light on the incomplete hydration reaction of MgO. On pure MgO surfaces, water adsorbs as intact molecule, which could explain low reactivity. By replacing Mg by Ca atoms, the preferred adsorption structure changed to two hydroxyl groups (i.e. dissociated water), thus achieving the experimentally found faster and complete hydration for the mixed surfaces. The results of these study have already been published [155].

   To model these systems, surface slabs with five MgO layers were modeled with a vacuum layer between neighboring slabs. These slabs need a larger number of APWs due to the vacuum layer, which in turn means large matrices in `LAPW1`. Therefore, these calculations could make efficient use of MPI parallelism.

3. The calculations for the Delta project were not routine, as a large number of small systems (most of them hypothetical) were simulated to create benchmark data for two newly created test sets.

   Even though the structures contained within these test sets are very small (less than 10 atoms per cell), both k-point parallelization and efficient task parallelism are crucially important, simply because of the large number of calculations needed. For instance, to provide high quality reference data for 570 oxides, seven calculations per oxide with very large ($33 \times 33 \times 33$) k-meshes had to be conducted.

   The purpose of the Delta project is the creation of an expansive test set that can be used to thoroughly assess the precision and transferability of DFT codes. This is still an ongoing and actually very active field of research, a large-scale test comparison of 9 pseudopotential codes using all-electron data of WIEN2k and FLEUR [165] as reference was recently submitted [159].

List of Figures

List of Tables

List of Algorithms

Abbreviations and Acronyms

a.u.	**a**tomic **u**nits
AFM	**A**tomic **F**orce **M**icroscopy
APW	**A**ugmented **P**lane **W**aves
BLAS	**B**asic **L**inear **A**lgebra **S**ubprograms
BO	**B**orn-**O**ppenheimer
CC	**C**oupled **C**luster
CI	**C**onfiguration **I**nteraction
CPU	**C**entral **P**rocessing **U**nit
DFT	**D**ensity **F**unctional **T**heory
DOS	**D**ensity **o**f **S**tates
ELPA	**E**igenvalue so**L**vers for **P**etaflop-**A**pplications
FFTW	**F**astest **F**ourier **T**ransform in the **W**est
GGA	**G**eneralized **G**radient **A**pproximation
GTO	**G**aussian-**T**ype **O**rbital
HDLO	**h**igh **d**erivative **L**ocal **O**rbital
HELO	**h**igh **e**nergy **L**ocal **O**rbital
HF	**H**artree-**F**ock
HPC	**H**igh-**P**erformance **C**omputing
ICSD	**I**norganic **C**rystal **S**tructure **D**atabase
IO	**I**nput/**O**utput
KS	**K**ohn-**S**ham

LAPACK	**L**inear **A**lgebra **PACK**age
LAPW	**L**inearized **A**ugmented **P**lane **W**aves
LDA	**L**ocal **D**ensity **A**pproximation
LO	**L**ocal **O**rbital of LAPW method
lo	local **o**rbital of APW+lo method
LSDA	**L**ocal **S**pin **D**ensity **A**pproximation
mGGA	**m**eta-**GGA**
MKL	**M**ath **K**ernel **L**ibrary
MP	**M**øller–**P**lesset
MPI	**M**essage **P**assing **I**nterface
MR^3	**M**ultiple **R**elatively **R**obust **R**epresentations
MRE	**M**ean **R**elative **E**rror
NMR	**N**uclear **M**agnetic **R**esonance
OD	**O**rder-**D**isorder
OpenMP	**O**pen **M**ulti-**P**rocessing
PAW	**P**rojector **A**ugmented **W**ave
PBE	**P**erdew-**B**urke-**E**rnzerhof
PBLAS	**P**arallel **BLAS**
RPA	**R**andom **P**hase **A**pproximation
ScaLAPACK	**Sca**lable **LAPACK**
SCF	**S**elf-**C**onsistent **F**ield
STM	**S**canning **T**unneling **M**icroscopy
STO	**S**later-**T**ype **O**rbital
TDSE	**T**ime-**D**ependent **S**chrödinger **E**quation
TFD	**T**homas-**F**ermi-**D**irac
TISE	**T**ime-**I**ndependent **S**chrödinger **E**quation
UEG	**U**niform **E**lectron **G**as
VML	**V**ector **M**ath **L**ibrary
VSC	**V**ienna **S**cientific **C**luster
XAS	**X**-ray **A**bsorption **S**pectroscopy
XC	e**x**change-**c**orrelation
XPS	**X**-ray **P**hotoelectron **S**pectroscopy

XRD **X-R**ay **D**iffraction

Bibliography

1. Nilsson, L. & Karplus, M. "Empirical energy functions for energy minimization and dynamics of nucleic acids". *Journal of Computational Chemistry* **7,** 591–616 (1986).
2. Patel, S. & Brooks, C. L. "CHARMM fluctuating charge force field for proteins: I parameterization and application to bulk organic liquid simulations". *Journal of Computational Chemistry* **25,** 1–16 (2003).
3. Hammersley, J. M. & Handscomb, D. C. *Monte Carlo Methods.* ISBN: 0-416-52340-4 (Methuen, London, 1975).
4. Binder, K. *The Monte Carlo Method in Condensed Matter Physics.* ISBN: 0-387-54369-4 (Springer, New York, 1995).
5. Jones, R. "Density functional theory: Its origins, rise to prominence, and future". *Reviews of Modern Physics* **87,** 897–923 (2015).
6. Mardirossian, N. & Head-Gordon, M. "Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals". *Molecular Physics* **115,** 2315–2372 (2017).
7. Blaha, P., Schwarz, K., *et al. WIEN2k: An Augmented Plane Wave plus Local Orbitals Program for Calculating Crystal Properties.* ISBN: 3-9501031-1-2 (Vienna University of Technology, Austria, 2019).
8. Blaha, P., Schwarz, K., *et al.* "WIEN2k: An APW+lo program for calculating the properties of solids". *The Journal of Chemical Physics* **152,** 074101 (2020).
9. Levin, F. S. *An Introduction to Quantum Theory,* 129–173. 808 pp. ISBN: 978-0521598415 (Cambridge University Press, 2001).
10. Berman, P. R. *Introductory Quantum Mechanics,* 77–114. 653 pp. ISBN: 978-3-319-68598-4 (Springer International Publishing, 2018).
11. Nottale, L. & Célérier, M.-N. "Derivation of the postulates of quantum mechanics from the first principles of scale relativity". *Journal of Physics A: Mathematical and Theoretical* **40,** 14471–14498 (2007).
12. Schrödinger, E. "Quantisierung als Eigenwertproblem - Erste Mitteilung". *Annalen der Physik* **79(4),** 361–376 (1926).

13. Schrödinger, E. “Quantisierung als Eigenwertproblem - Zweite Mitteilung”. *Annalen der Physik* **79(4),** 489–527 (1926).

14. Schrödinger, E. “Quantisierung als Eigenwertproblem - Dritte Mitteilung”. *Annalen der Physik* **80(4),** 437–490 (1926).

15. Schrödinger, E. “Quantisierung als Eigenwertproblem - Vierte Mitteilung”. *Annalen der Physik* **81(4),** 109–139 (1926).

16. Schrödinger, E. “An Undulatory Theory of the Mechanics of Atoms and Molecules”. *Physical Review* **28,** 1049–1070 (1926).

17. Born, M. & Oppenheimer, R. “Zur Quantentheorie der Molekeln”. *Annalen der Physik* **84(4),** 457–484 (1927).

18. Schwenke, D. W. “Beyond the Potential Energy Surface: Ab initio Corrections to the Born-Oppenheimer Approximation for H_2O”. *The Journal of Physical Chemistry A* **105,** 2352–2360 (2001).

19. Holka, F., Szalay, P. G., *et al.* “Accurate ab initio determination of the adiabatic potential energy function and the Born–Oppenheimer breakdown corrections for the electronic ground state of LiH isotopologues”. *The Journal of Chemical Physics* **134,** 094306 (2011).

20. Pack, R. T. & Hirschfelder, J. O. “Energy Corrections to the Born–Oppenheimer Approximation. The Best Adiabatic Approximation”. *The Journal of Chemical Physics* **52,** 521–534 (1970).

21. Tajti, A., Szalay, P. G. & Gauss, J. “Perturbative treatment of the electron-correlation contribution to the diagonal Born-Oppenheimer correction”. *The Journal of Chemical Physics* **127,** 014102 (2007).

22. Hartree, D. R. “The Wave Mechanics of an Atom with a Non-Coulomb Central Field. Part I. Theory and Methods”. *Mathematical Proceedings of the Cambridge Philosophical Society* **24,** 89–110 (1928).

23. Fock, V. “Näherungsmethode zur Lösung des quantenmechanischen Mehrkörperproblems”. *Zeitschrift für Physik* **61,** 126–148 (1930).

24. Slater, J. C. “Note on Hartree’s Method”. *Physical Review* **35,** 210–211 (1930).

25. Hartree, D. R. & Hartree, W. “Self-consistent field, with exchange, for beryllium”. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences* **150,** 9–33 (1935).

26. Slater, J. C. “A Simplification of the Hartree-Fock Method”. *Physical Review* **81,** 385–390 (1951).

27. Koopmans, T. “Über die Zuordnung von Wellenfunktionen und Eigenwerten zu den Einzelnen Elektronen Eines Atoms”. *Physica* **1,** 104–113 (1934).

28. Jensen, F. *Introduction to Computational Chemistry* 2nd Edition. 620 pp. ISBN: 978-0470011874 (Wiley, 2011).

29. Møller, C. & Plesset, M. S. "Note on an Approximation Treatment for Many-Electron Systems". *Physical Review* **46,** 618–622 (1934).

30. Kohn, W. "Nobel Lecture: Electronic structure of matter—wave functions and density functionals". *Reviews of Modern Physics* **71,** 1253–1266 (1999).

31. Strout, D. L. & Scuseria, G. E. "A quantitative study of the scaling properties of the Hartree–Fock method". *The Journal of Chemical Physics* **102,** 8448–8452 (1995).

32. Jin, Y. & Bartlett, R. J. "Perturbation Improved Natural Linear-Scaled Coupled-Cluster Method and Its Application to Conformational Analysis". *The Journal of Physical Chemistry A* **123,** 371–381 (2018).

33. Hohenberg, P. & Kohn, W. "Inhomogeneous electron gas". *Physical Reviews* **136,** B864–B871 (1964).

34. Sun, J., Furness, J. W. & Zhang, Y. in *Mathematical Physics in Theoretical Chemistry* (ed S. M. Binder, J. E. H.) 119–159 (Elsevier, 2019). ISBN: 978-0-12-813651-5.

35. Becke, A. D. "Perspective: Fifty years of density-functional theory in chemical physics". *The Journal of Chemical Physics* **140,** 18A301 (2014).

36. Thomas, L. H. "The calculation of atomic fields". *Mathematical Proceedings of the Cambridge Philosophical Society* **23,** 542–548 (1927).

37. Fermi, E. "Un metodo statistico per la determinazione di alcune prioprietà dell'atomo". *Rendiconti Academia Dei Lincei* **6,** 602–607 (1927).

38. Dirac, P. A. M. "Note on Exchange Phenomena in the Thomas Atom". *Mathematical Proceedings of the Cambridge Philosophical Society* **26,** 376–385 (1930).

39. Teller, E. "On the Stability of Molecules in the Thomas-Fermi Theory". *Reviews of Modern Physics* **34,** 627–631 (1962).

40. Karasiev, V. V. & Trickey, S. B. in *Advances in Quantum Chemistry* 221–245 (Elsevier, 2015).

41. Kohn, W. & Sham, L. J. "Self-consistent equations including exchange and correlation effects". *Physical Reviews* **140,** A1133–A1138 (1965).

42. Perdew, J. P. *Jacob's ladder of density functional approximations for the exchange-correlation energy.* in *AIP Conference Proceedings* (AIP, 2001).

43. Tran, F., Stelzl, J. & Blaha, P. "Rungs 1 to 4 of DFT Jacob's ladder: Extensive test on the lattice constant, bulk modulus, and cohesive energy of solids". *The Journal of Chemical Physics* **144,** 204120 (2016).

44. Ceperley, D. M. & Alder, B. J. "Ground State of the Electron Gas by a Stochastic Method". *Physical Review Letters* **45,** 566–569 (1980).

45. Vosko, S. H., Wilk, L. & Nusair, M. "Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis". *Canadian Journal of Physics* **58,** 1200–1211 (1980).

46. Perdew, J. P. & Zunger, A. "Self-interaction correction to density-functional approximations for many-electron systems". *Physical Review B* **23,** 5048–5079 (1981).

47. Perdew, J. P. & Wang, Y. "Accurate and simple analytic representation of the electron-gas correlation energy". *Physical Review B* **45,** 13244–13249 (1992).

48. Oliver, G. L. & Perdew, J. P. "Spin-density gradient expansion for the kinetic energy". *Physical Review A* **20,** 397–403 (1979).

49. Von Barth, U. & Hedin, L. "A local exchange-correlation potential for the spin polarized case. i". *Journal of Physics C: Solid State Physics* **5,** 1629–1642 (1972).

50. Becke, A. D. "Density functional calculations of molecular bond energies". *The Journal of Chemical Physics* **84,** 4524–4529 (1986).

51. Perdew, J. P., Ruzsinszky, A., *et al.* "Prescription for the design and selection of density functional approximations: More constraint satisfaction with fewer fits". *The Journal of Chemical Physics* **123,** 062201 (2005).

52. Perdew, J. P., Burke, K. & Ernzerhof, M. "Generalized Gradient Approximation Made Simple". *Physical Review Letters* **77,** 3865–3868 (1996).

53. Perdew, J. P., Ruzsinszky, A., *et al.* "Restoring the Density-Gradient Expansion for Exchange in Solids and Surfaces". *Physical Review Letters* **100,** 136406 (2008).

54. Zhang, Y. & Yang, W. "Comment on "Generalized Gradient Approximation Made Simple"". *Physical Review Letters* **80,** 890–890 (1998).

55. Filatov, M. & Thiel, W. "Exchange-correlation density functional beyond the gradient approximation". *Physical Review A* **57,** 189–199 (1998).

56. Sun, J., Ruzsinszky, A. & Perdew, J. "Strongly Constrained and Appropriately Normed Semilocal Density Functional". *Physical Review Letters* **115,** 036402 (2015).

57. Perdew, J. P., Kurth, S., *et al.* "Accurate Density Functional with Correct Formal Properties: A Step Beyond the Generalized Gradient Approximation". *Physical Review Letters* **82,** 2544–2547 (1999).

58. Tao, J., Perdew, J. P., *et al.* "Climbing the Density Functional Ladder: Nonempirical Meta–Generalized Gradient Approximation Designed for Molecules and Solids". *Physical Review Letters* **91,** 146401 (2003).

59. Perdew, J. P., Ruzsinszky, A., *et al.* "Workhorse Semilocal Density Functional for Condensed Matter Physics and Quantum Chemistry". *Physical Review Letters* **103,** 026403 (2009).

60. Sun, J., Haunschild, R., *et al.* "Semilocal and hybrid meta-generalized gradient approximations based on the understanding of the kinetic-energy-density dependence". *The Journal of Chemical Physics* **138,** 044113 (2013).

61. Wellendorff, J., Lundgaard, K. T., *et al.* "mBEEF: An accurate semi-local Bayesian error estimation density functional". *The Journal of Chemical Physics* **140,** 144107 (2014).

62. Becke, A. D. "Density-functional thermochemistry. III. The role of exact exchange". *The Journal of Chemical Physics* **98,** 5648–5652 (1993).

63. Perdew, J. P., Ernzerhof, M. & Burke, K. "Rationale for mixing exact exchange with density functional approximations". *The Journal of Chemical Physics* **105,** 9982–9985 (1996).

64. Harl, J., Schimka, L. & Kresse, G. "Assessing the quality of the random phase approximation for lattice constants and atomization energies of solids". *Physical Review B* **81,** 115126 (2010).

65. Anisimov, V. I., Zaanen, J. & Andersen, O. K. "Band theory and Mott insulators: Hubbard U instead of Stoner I". *Physical Review B* **44,** 943–954 (1991).

66. Grimme, S., Antony, J., *et al.* "A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu". *The Journal of Chemical Physics* **132,** 154104 (2010).

67. Dion, M., Rydberg, H., *et al.* "Van der Waals Density Functional for General Geometries". *Physical Review Letters* **92,** 246401 (2004).

68. Román-Pérez, G. & Soler, J. M. "Efficient Implementation of a van der Waals Density Functional: Application to Double-Wall Carbon Nanotubes". *Physical Review Letters* **103,** 096102 (2009).

69. Zhao, Y., Lynch, B. J. & Truhlar, D. G. "Doubly Hybrid Meta DFT: New Multi-Coefficient Correlation and Density Functional Methods for Thermochemistry and Thermochemical Kinetics". *The Journal of Physical Chemistry A* **108,** 4786–4791 (2004).

70. Grimme, S. "Semiempirical hybrid density functional with perturbative second-order correlation". *The Journal of Chemical Physics* **124,** 034108 (2006).

71. Perdew, J. P. & Levy, M. "Physical Content of the Exact Kohn-Sham Orbital Energies: Band Gaps and Derivative Discontinuities". *Physical Review B* **51,** 1884–1887 (1983).

72. Janak, J. F. "Proof that $\frac{\partial E}{\partial n_i} = \varepsilon$ in density-functional theory". *Physical Review B* **18,** 7165–7168 (1978).

73. Bachelet, G. B., Hamann, D. R. & Schlüter, M. "Pseudopotentials that work: From H to Pu". *Physical Review B* **26,** 4199–4228 (1982).

74. Vanderbilt, D. "Soft self-consistent pseudopotentials in a generalized eigenvalue formalism". *Physical Review B* **41,** 7892–7895 (1990).

75. Blöchl, P. E. "Projector augmented-wave method". *Physical Review B* **50,** 17953–17979 (1994).

76. Martin, R. M. *Electronic Structure: Basic Theory and Practical Methods* (Cambridge University Press, 2004).

77. Woods, N. D., Payne, M. C. & Hasnip, P. J. "Computing the self-consistent field in Kohn–Sham density functional theory". *Journal of Physics: Condensed Matter* **31,** 453001 (2019).

78. Hartree, D. R. "The Wave Mechanics of an Atom with a Non-Coulomb Central Field. Part II. Some Results and Discussion". *Mathematical Proceedings of the Cambridge Philosophical Society* **24,** 111–132 (1928).

79. Kresse, G. & Hafner, J. "Ab initiomolecular dynamics for liquid metals". *Physical Review B* **47,** 558–561 (1993).

80. Kresse, G. & Furthmüller, J. "Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set". *Computational Materials Science* **6,** 15–50 (1996).

81. Kresse, G. & Furthmüller, J. "Efficient iterative schemes forab initiototal-energy calculations using a plane-wave basis set". *Physical Review B* **54,** 11169–11186 (1996).

82. Gulans, A., Kontur, S., *et al.* "exciting: a full-potential all-electron package implementing density-functional theory and many-body perturbation theory". *Journal of Physics: Condensed Matter* **26,** 363202 (2014).

83. Marks, L. D. & Luke, D. R. "Robust mixing for *ab initio* quantum mechanical calculations". *Physical Review B* **78** (2008).

84. Blaha, P., Schwarz, K., *et al.* "Full-potential, linearized augmented plane wave programs for crystalline systems". *Computer Physics Communications* **59,** 399–415 (1990).

85. Schwarz, K., Blaha, P. & Madsen, G. "Electronic structure calculations of solids using the WIEN2k package for material sciences". *Computer Physics Communications* **147,** 71–76 (2002).

86. Schwarz, K. & Blaha, P. "Solid state calculations using WIEN2k". *Computational Materials Science* **28,** 259–273 (2003).

87. Tran, F., Stelzl, J., *et al.* "Simple way to apply nonlocal van der Waals functionals within all-electron methods". *Physical Review B* **96,** 054103 (2017).

88. Bagheri, M. & Blaha, P. "DFT calculations of energy dependent XPS valence band spectra". *Journal of Electron Spectroscopy and Related Phenomena* **230,** 1–9 (2019).

89. Tran, F. & Blaha, P. "Accurate Band Gaps of Semiconductors and Insulators with a Semilocal Exchange-Correlation Potential". *Physical Review Letters* **102,** 226401 (2009).

90. Karsai, F., Tran, F. & Blaha, P. "On the importance of local orbitals using second energy derivatives for d and f electrons". *Computer Physics Communications* **220,** 230–238 (2017).

91. Bloch, F. “Über die Quantenmechanik der Elektronen in Kristallgittern”. *Zeitschrift für Physik* **52,** 555–600 (1929).

92. Slater, J. C. “Wave Functions in a Periodic Potential”. *Physical Review* **51,** 846–851 (1937).

93. Andersen, O. K. “Simple approach to the band-structure problem”. *Solid State Communications* **13,** 133–136 (1973).

94. Andersen, O. K. “Linear methods in band theory”. *Physical Review B* **12,** 3060–3083 (1975).

95. Singh, D. “Ground-state properties of lanthanum: Treatment of extended-core states”. *Physical Review B* **43,** 6388–6392 (1991).

96. Sjöstedt, E., Nordström, L. & Singh, D. “An alternative way of linearizing the augmented plane-wave method”. *Solid State Communications* **114,** 15–20 (2000).

97. Madsen, G. K. H., Blaha, P., *et al.* “Efficient linearization of the augmented plane-wave method”. *Physical Review B* **64** (2001).

98. Michalicek, G., Betzinger, M., *et al.* “Elimination of the linearization error and improved basis-set convergence within the FLAPW method”. *Computer Physics Communications* **184,** 2670–2679 (2013).

99. Anisimov, V. I., Solovyev, I. V., *et al.* “Density-functional theory and NiO photoemission spectra”. *Physical Review B* **48,** 16929–16934 (1993).

100. Czyżyk, M. T. & Sawatzky, G. A. “Local-density functional and on-site correlations: The electronic structure of La_2CuO_4 and $LaCuO_3$”. *Physical Review B* **49,** 14211–14228 (1994).

101. Grimme, S., Ehrlich, S. & Goerigk, L. “Effect of the damping function in dispersion corrected density functional theory”. *Journal of Computational Chemistry* **32,** 1456–1465 (2011).

102. Caldeweyher, E., Bannwarth, C. & Grimme, S. “Extension of the D3 dispersion coefficient model”. *The Journal of Chemical Physics* **147,** 034112 (2017).

103. Caldeweyher, E., Ehlert, S., *et al.* “A generally applicable atomic-charge dependent London dispersion correction”. *The Journal of Chemical Physics* **150,** 154122 (2019).

104. Caldeweyher, E., Mewes, J.-M., *et al.* “Extension and evaluation of the D4 London-dispersion model for periodic systems”. *Physical Chemistry Chemical Physics* **22,** 8499–8512 (2020).

105. Desclaux, J. “Hartree Fock Slater self consistent field calculations”. *Computer Physics Communications* **1,** 216–222 (1970).

106. Desclaux, J. “A multiconfiguration relativistic DIRAC-FOCK program”. *Computer Physics Communications* **9,** 31–45 (1975).

107. Blackford, L. S., Demmel, J., *et al.* "An updated set of basic linear algebra subprograms (BLAS)". *ACM Transactions on Mathematical Software* **28-2,** 135–151 (2002).

108. Anderson, E., Bai, Z., *et al. LAPACK Users' Guide* Third. ISBN: 0-89871-447-8 (paperback) (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999).

109. Choi, J., Dongarra, J. & Walker, D. *Parallel Matrix Transpose Algorithms on Distributed Memory Concurrent Computers.* Tech. rep. (Center for Research on Parallel Computation, Rice University, Houston, TX, USA).

110. Blackford, L. S., Choi, J., *et al. ScaLAPACK Users' Guide.* ISBN: 0-89871-397-8 (paperback) (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997).

111. Auckenthaler, T., Blum, V., *et al.* "Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations". *Parallel Computing* **37,** 783–794 (2011).

112. Marek, A., Blum, V., *et al.* "The ELPA library - scalable parallel eigenvalue solutions for electronic structure theory and computational science". *Journal of Physics: Condensed Matter* **26,** 21320 (2014).

113. Ipsen, I. C. F. "Computing an Eigenvector with Inverse Iteration". *SIAM Review* **39,** 254–291 (1997).

114. Kvasnicka, D. F., Gansterer, W. N. & Ueberhuber, C. W. *A Level 3 Algorithm for the Symmetric Eigenproblem.* in *VECPAR'98 - 3rd International Meeting on Vector and Parallel Processing* (1998), 267–275.

115. Parlett, B. N. & Dhillon, I. S. "Relatively robust representations of symmetric tridiagonals". *Linear Algebra and its Applications* **309,** 121–151 (2000).

116. Blaha, P., Hofstätter, H., *et al.* "Iterative diagonalization in augmented plane wave based methods in electronic structure calculations". *Journal of Computational Physics* **229,** 453–460 (2010).

117. *Parallel Computing* (eds Trobec, R., Vajteršic, M. & Zinterhof, P.) ISBN: 978-1-84882-409-6 (Springer London, 2009).

118. *Parallel Computing: On the Road to Exascale* (eds Joubert, G. R., Leather, H., *et al.*) ISBN: 978-1-61499-621-7 (IOS Press, 2016).

119. Robey, R. & Zamora, Y. *Parallel and High Performance Computing.* ISBN: 978-1617296468 (Manning Publications, 2021).

120. Dagum, L. & Menon, R. "OpenMP: an industry standard API for shared-memory programming". *Computational Science & Engineering, IEEE* **5,** 46–55 (1998).

121. Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 4.0* (2021).

122. Frigo, M. & Johnson, S. “The Design and Implementation of FFTW3”. *Proceedings of the IEEE* **93,** 216–231 (2005).

123. Ruh, T. & Blaha, P. *Evaluating eigensolver schemes within the density functional theory package WIEN2k.* in *2016 International Conference on High Performance Computing & Simulation (HPCS)* (IEEE Conference Publications, 2016), 973–978.

124. Laskowski, R. & Blaha, P. “Calculations of NMR chemical shifts with APW-based methods”. *Physical Review B* **85,** 035132 (2012).

125. Laskowski, R. & Blaha, P. “Calculating NMR chemical shifts using the augmented plane-wave method”. *Physical Review B* **89,** 014402 (2014).

126. Laskowski, R. & Blaha, P. “NMR Shielding in Metals Using the Augmented Plane Wave Method”. *The Journal of Physical Chemistry C* **119,** 19390–19396 (2015).

127. Demmel, J. W., Marques, O. A., *et al.* “Performance and Accuracy of LAPACK’s Symmetric Tridiagonal Eigensolvers”. *SIAM Journal on Scientific Computing* **30,** 1508–1526 (2008).

128. Dhillon, I. S. & Parlett, B. N. “Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices”. *Linear Algebra and its Applications* **387,** 1–28 (2004).

129. Dhillon, I. S., Parlett, B. N. & Vömel, C. “The design and implementation of the MRRR algorithm”. *ACM Transactions on Mathematical Software* **32,** 533–560 (2006).

130. Cuppen, J. “A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem.” *Numerische Mathematik* **36,** 177–196 (1980/81).

131. Gu, M. & Eisenstat, S. C. “A Divide-and-Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem”. *SIAM Journal on Matrix Analysis and Applications* **16,** 172–191 (1995).

132. Francis, J. G. F. “The QR Transformation A Unitary Analogue to the LR Transformation–Part 1”. *The Computer Journal* **4,** 265–271 (1961).

133. Francis, J. G. F. “The QR Transformation–Part 2”. *The Computer Journal* **4,** 332–345 (1962).

134. Kara, M. & Kurki-Suonio, K. “Symmetrized multipole analysis of orientational distributions”. *Acta Crystallographica Section A* **37,** 201–210 (1981).

135. Kokalj, A. “Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale”. *Computational Materials Science* **28,** 155–168 (2003).

136. Momma, K. & Izumi, F. “VESTA: a three-dimensional visualization system for electronic and structural analysis”. *Journal of Applied Crystallography* **41,** 653–658 (2008).

137. Höfinger, S., Ruh, T. & Haunschmid, E. “Fast Approximate Evaluation of Parallel Overhead from a Minimal Set of Measured Execution Times”. *Parallel Processing Letters* **28,** 1850003 (2018).

138. Householder, A. S. “Unitary Triangularization of a Nonsymmetric Matrix”. *Journal of the ACM* **5,** 339–342 (1958).

139. Stöger, B. & Ruh, T., Manuscript in preparation.

140. Scordari, F., Scandale, E. & Giacovazzo, C. “The crystal structure of chlorotionite, $CuK_2Cl_2SO_4$”. *Zeitschrift für Kristallographie - Crystalline Materials* **144,** 226–237 (1976).

141. Dornberger-Schiff, K. & Grell-Niemann, H. “On the theory of order–disorder (OD) structures”. *Acta Crystallographica* **14,** 167–177 (1961).

142. Stöger, B., Weil, M., *et al.* “The Order-Disorder (OD) Polytypism of $[Cu_2ZnTeO_4]^{2+}$ $[SO_4 \cdot H2O]^{2-}$”. *Crystal Research and Technology* **55,** 1900182 (2020).

143. Ďurovič, S. “Desymmetrization of OD structures”. *Kristall und Technik,* 1047–1053 (14 1979).

144. Marks, L. D. “Fixed-Point Optimization of Atoms and Density in DFT”. *Journal of Chemical Theory and Computation* **9,** 2786–2800 (2013).

145. Marks, L. D. “Predictive Mixing for Density Functional Theory (and Other Fixed-Point Problems)”. *Journal of Chemical Theory and Computation* **17,** 5715–5732 (2021).

146. Monkhorst, H. J. & Pack, J. D. “Special points for Brillouin-zone integrations”. *Physical Review B* **13,** 5188–5192 (1976).

147. Deutsch, M., Müller, D., *et al.* “Systematic search algorithm for potential thermochemical energy storage systems”. *Applied Energy* **183,** 113–120 (2016).

148. Cot-Gores, J., Castell, A. & Cabeza, L. F. “Thermochemical energy storage and conversion: A-state-of-the-art review of the experimental research under practical conditions”. *Renewable and Sustainable Energy Reviews* **16,** 5207–5224 (2012).

149. Tsirelson, V. G., Avilov, A. S., *et al.* “X-ray and Electron Diffraction Study of MgO”. *Acta Crystallographica Section B Structural Science* **54,** 8–17 (1998).

150. Shen, C., Liu, R., *et al.* “Phase stability study of $La_1.2Ca_1.8Mn_2O_7$”. *Materials Research Bulletin* **36,** 1139–1148 (2001).

151. Vegard, L. “Die Konstitution der Mischkristalle und die Raumfüllung der Atome”. *Zeitschrift für Physik* **5,** 17–26 (1921).

152. Alfonso, D. R., Snyder, J. A., *et al.* “Opposite rumpling of the MgO and CaO (100) surfaces: A density-functional theory study”. *Physical Review B* **62,** 8318–8322 (2000).

153. Logsdail, A. J., Mora-Fonz, D., *et al.* “Structural, energetic and electronic properties of (100) surfaces for alkaline earth metal oxides as calculated with hybrid density functional theory”. *Surface Science* **642,** 58–65 (2015).

154. Hu, X. L., Carrasco, J., *et al.* "Trends in water monomer adsorption and dissociation on flat insulating surfaces". *Physical Chemistry Chemical Physics* **13,** 12447 (2011).

155. Müller, D., Knoll, C., *et al.* "Calcium Doping Facilitates Water Dissociation in Magnesium Oxide". *Advanced Sustainable Systems* **2,** 1700096 (2017).

156. Peelaers, H., Steiauf, D., *et al.* "$(In_xGa_{1-x})_2O_3$ alloys for transparent electronics". *Physical Review B* **92,** 085206 (2015).

157. Okhotnikov, K., Charpentier, T. & Cadars, S. "Supercell program: a combinatorial structure-generation approach for the local-level modeling of atomic substitutions and partial occupancies in crystals". *Journal of Cheminformatics* **8** (2016).

158. Doman, R. C., Barr, J. B., *et al.* "Phase Equilibria in the System CaO-MgO". *Journal of the American Ceramic Society* **46,** 313–316 (1963).

159. Bosoni, E., Beal, L., *et al. How to verify the precision of density-functional-theory implementations via reproducible and universal workflows.* 2023. arXiv: 2305.17274 [cond-mat.mtrl-sci].

160. Jain, A., Ong, S. P., *et al.* "Commentary: The Materials Project: A materials genome approach to accelerating materials innovation". *APL Materials* **1,** 011002 (2013).

161. Draxl, C. & Scheffler, M. "The NOMAD laboratory: from data sharing to artificial intelligence". *Journal of Physics: Materials* **2,** 036001 (2019).

162. Lejaeghere, K., Speybroeck, V. V., *et al.* "Error Estimates for Solid-State Density-Functional Theory Predictions: An Overview by Means of the Ground-State Elemental Crystals". *Critical Reviews in Solid State and Materials Sciences* **39,** 1–24 (2013).

163. Lejaeghere, K., Bihlmayer, G., *et al.* "Reproducibility in density functional theory calculations of solids". *Science* **351,** aad3000–aad3000 (2016).

164. Birch, F. "Finite Elastic Strain of Cubic Crystals". *Physical Review* **71,** 809–824 (1947).

165. *The FLEUR project.* https://www.flapw.de/.

166. Zagorac, D., Müller, H., *et al.* "Recent developments in the Inorganic Crystal Structure Database: theoretical crystal structure data and related features". *Journal of Applied Crystallography* **52,** 918–925 (2019).