Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Implementace kvantově-mechanických výpočtů elektronové struktury a vlastností neperiodických materiálových struktur

Nová ab-initio metoda založená na teorii funkcionálu hustoty, metodě konečných prvků a pseudopotenciálech

Mgr. Matyáš Novák

disertační práce
k získání akademického titulu doktor
v oboru Aplikovaná mechanika

Školitel: RNDr. Jiří Vackář, CSc.
Katedra: Katedra mechaniky

Plzeň 2020

University of West Bohemia in Pilsen
Faculty of Applied Sciences

Implementation of quantum-mechanical calculations of electronic structure and material properties of non-periodic structures

A new ab-initio method based on the density functional theory, the finite element method and pseudopotentials

Mgr. Matyáš Novák

dissertation thesis
to obtain the academic title “Doctor”
in field Applied Mechanics

Supervisor: RNDr. Jiří Vackář, CSc.
Department: Department of Mechanics

Pilsen 2020

Prohlášení

Předkládám tímto k posouzení a obhajobě disertační práci zpracovanou na závěr doktorského studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni. Prohlašuji, že jsem tuto disertační práci zpracoval samostatně (popř. části vzniklé v spoluautorství explicitně označil), a to s použitím odborné literatury a dostupných pramenů uvedených v seznamu, jenž je součástí této práce.

V Praze, 16. 10. 2020 Matyáš Novák

Poděkování

Rád bych na tomto místě poděkoval všem, kteří přispěli k tomu, že mohla být nová metoda na výpočet elektronových struktur vyvinuta, FENNEC naprogramován a disertace dokončena. Zejména pak svému školiteli, RNDr. Jiřímu Vackářovi CSc., a také a také Ing. Robertu Cimrmanovi, Ph.D. a RNDr. Ondřeji Šiprovi, CSc. — za veškeré jejich vedení, cenné rady, pomoc i trpělivost.

Děkuji také doc. Dr. RNDr. Miroslavu Holečkovi, který byl mým školitelem na počátku mého doktorského studia, prof. Miroslavu Tůmovi, CSc., který vedl moji diplomovou práci, v níž započala má práce na vývoji programu FENNEC, Mgr. Otakaru Frankovi Ph.D., neboť poslední kapitola disertace je plodem přípravy společného grantu, Vojtěchu Fišerovi, který ač ještě student bakalářského studia zvládl pomoci s novou implementací řešičů a všem ostatním spolupracovníkům našeho řešitelského týmu.

Především však děkuji své ženě Agátě, za krásných společných patnáct let života i za veškerou její pomoc, podporu, toleranci, pochopení a obětavost, bez kterých by bylo dokončení této disertační práce nemyslitelné.

Děkuji.

Title: **Implementation of quantum-mechanical calculations of electronic structure and material properties of non-periodic structures**

Author: Mgr. Matyáš Novák

Department: Department of Mechanics, Faculty of Applied Sciences, University of West Bohemia

Supervisor: RNDr. Jiří Vackář, CSc

Abstract:

A new ab-inito method and the related computer code has been developed within this work — in accordance with its aims — for calculating electronic states and structural properties of (primarily) nonperiodic, possibly electrically charged, material structures. This method has been implemented into the software package FENNEC.

The new method fills a gap among the current well-established methods and represents a counterpart of what the plane-wave method is within the class of the methods employing the Bloch's theorem (for materials with translational symmetry). The newly developed method is based on the density functional theory (DFT), the finite element method, or isogeometric analysis as an option, and environment-reflecting separable pseudopotentials, and it provides an excellent convergence control within a general, in principle arbitrarily precise basis.

During the development of the method, a new precise and computationally efficient formula for evaluating Hellmann-Feynman forces has been derived, implemented, and used for geometry optimizations. Its advantages and very good convergence properties have been demonstrated in sample calculations.

Among other originals results of this work, a purely algebraic description of separable pseudopotentials should be mentioned. This description allows constructing a projection basis to achieve any required accuracy.

Besides that, an adaptive version of the Anderson iteration scheme, which solves the problem of choosing the right mixing parameter (an approximation of the Jacobian) in the DFT loop, has been proposed, implemented, and analyzed.

Discretization of the Kohn-Sham equations using the finite element methods leads to a rank-k-update generalized eigenvalue problem. To solve it, an efficient method had to be developed: to find the appropriate eigenvalue solvers, integrate them into the FENNEC software package, and make them capable to solve the rank-k-update problem.

The convergence properties of that method in dependence on various technical parameters — i.e. the finite element shape, basis type, and the order and number of projectors for separable pseudopotentials — have been examined and analyzed to determine the best options for the new method.

The newly created method has been implemented using the Python finite element framework SfePy (and many other libraries) into the software package FENNEC. This software package is characterized by a modular plugin architecture which significantly simplifies its further development and makes its usage easy. FENNEC has been used, among other tasks, e.g. for geometry optimization of a deformed graphene fragment and calculations of forces in it as a preparation for the ab-initio construction of force-interaction molecular dynamics potentials.

Název práce: Implementace kvantově-mechanických výpočtů elektronové struktury a vlastností neperiodických materiálových struktur

AUTOR: Mgr. Matyáš Novák

KATEDRA: Fakulta mechaniky, Fakulta aplikovaných věd, Západočeská universita

ŠKOLITEL: RNDr. Jiří Vackář, CSc

ABSTRAKT:

V této práci — v souladu s jejím cílem — byla vyvinuta nová ab-inito metoda a odpovídající software pro výpočet elektronových stavů a strukturních vlastností (primárně) neperiodických, popřípadně elektricky nabitých, materiálových struktur. Tato metoda byla implementována v softwarovém balíku FENNEC.

Tato nová metoda zaplňuje mezeru mezi současnými běžně užívanými metodami a představuje protějšek toho, čím je metoda rovinných vln mezi metodami užívajícími Blochův teorém (sloužícími pro výpočet materiálů s translační symetrií). Nová metoda je založena na teorii funkcionálu hustoty, metodě konečných prvků, popř. volitelně na isogeometrické analýze, a prostředí-reflektujících pseudopotenciálů, a nabízí výtečnou kontrolu konvergence díky obecné, v principu libovolně přesné bazi.

Během vývoje metody byla vyvinuta, implementována a pro geometrické optimalizace použita nová formulace výpočtu Hellmann-Feynmanových sil. Její přednosti, především výtečná konvergence, byly demonstrovány na ukázkových výpočtech.

Mezi další originální výsledky této práce patří čistě algebraické vyjádření separabilní formy pseudopotenciálů. Tento popis umožňuje konstrukci projekční báze tak, že jde dosáhnout libovolné požadované přesnosti.

Mimo to byla navržena, implementována a analyzována adaptivní verze Andersonova iterativního algoritmu, která řeší problém výběru mixovacího parametru (počátečního odhadu jakobiánu) v DFT selfkonsistentním cyklu.

Diskretizace Kohn-Shamových rovnic za pomoci metody konečných prvků vede k rank-k-update problému vlastních čísel, pro jehož řešení bylo třeba vyvinout efektivní metodu: najít vhodné řešiče, integrovat je do softwarového balíku FENNEC a rozšířit možnosti těchto řešičů o řešení rank-k-update problému.

Byly zkoumány a analyzovány konvergenční vlastnosti této metody v závislosti na různých technických parametrech — např. tvaru konečných elementů, typu báze a jejím řádu, počtu projektorů separabilních pseudopotenciálů — a touto cestou byly získány optimální hodnoty přinášející nejlepší výpočetní výkon a přesnost.

Tato metoda byla implementována v jazyce Python s využitím konečněprvkového rámce SfePy do softwarového balíku FENNEC. Tento software se vyznačuje modulární architekturou založenou na pluginech, která podstatně usnadňuje jeho další vývoj i přispívá k jeho velmi snadnému používání. Jako demonstrace schopností tohoto software je v práci prezentovaná optimalizace geometrie fragmentu grafenové vrstvy a výpočet sil potřebných na odtržení atomu z této vrstvy. Tyto výpočty budou sloužit k ab-initio konstrukci interakčních potenciálů pro simulace grafenu pomocí molekulární dynamiky.

Content

Introduction

Suitable tools for predicting the properties of newly proposed materials (e.g. elastic properties, hardness, magnetism, various material constants) seem to be extremely useful for present-day material development. Ab-initio electronic structure calculations take a unique place among these tools, because they allow us to determine the properties of the proposed material without any need for experimental input data and so without any need to produce a sample of the material (possibly even without knowing how to produce the material). In addition, ab-initio electronic structure calculations allow for deep insight into the properties of the material and provide a tool for the appropriate understanding of causal links and relations. Although verification by experiment is still necessary, the possibility to preselect suitable structures and parameters for the final experiment by means of previous theoretical calculations saves both time and costs in developing new materials, thus being almost indispensable in contemporary material engineering.

Therefore a considerable effort has been invested in developing methods for electronic structure calculations. Many different methods have been developed, each using certain approximations and so being suitable only for certain classes of problems. Up to the present, certain types of problems that can still be treated by means of the currently established methods only in a limited way, if at all.

The aim of my work — introduced in this thesis — has been to develop a new method for electronic structure calculation that could fill the existing gap between the currently established ab-initio methods in the field of non-periodic materials, possibly with charge neutrality violated (e.g. complex clusters, or structures with various kinds of defects). Among methods employing a general basis with an excellent convergence control, several are suitable for periodic materials (based on the Bloch theorem), but so far, there has been no such one suitable for solving general aperiodic problems directly and efficiently, without artificial, computationally demanding constructions (such as supercells).

The new method for electronic structure calculations is based on solving the Kohn-Sham equations (formulated by the density functional theory, abbreviated as DFT) discretized using the finite element method or the isogeometric analysis. L-dependent pseudopotentials are utilized to avoid numerical singularities. To retain sparsity of the matrices resulting from the discretization of the solved equations, the pseudopotentials have to be transformed into their separable form. This results in a rank-k-update generalized eigenvalue problem to be solved. The construction of the separable potentials is described in an original purely algebraic way that sheds more light on the ideas behind the construction process and offers possible improvements in the constructions of pseudopotentials.

One of the essential tasks in solid-state physics is geometry optimization — finding the positions of the atoms that minimizes the energy of the system. To minimize the energy efficiently, it is necessary to calculate *Hellmann-Feynman forces*: derivatives of the energy with respect to the positions of atom nuclei. Computing the Hellmann-Feynman forces requires developing a new formulation of the forces, suitable for use with the finite element method in combination with non/local potentials.

The new method including the new formulation of the Hellmann-Feynman forces has been implemented in the software package FENNEC. It has been developed largely in Python, some parts are written in Fortran or Cython. The development included the use and incorporation of many libraries. Particularly, I have made use of the finite element framework Sfepy, the numerical libraries Numpy and Scipy, and the eigenvalue solvers Jadamilu and Blzpack.

To transform the ideas mentioned above to fully functional computer code, one needs to find suitable solutions for many subproblems arising in many scientific disciplines (e.g. mathematical analysis, linear algebra, and numerical mathematic over quantum physics, computer sciences . . .). The thesis covers the essential building blocks of the new method quoted above.

Due to the necessity to combine elements from very different expert field — from linear algebra through numerical mathematics and computer science up to quantum physics — I have strived the thesis in as many self-contained ways as possible, to allow a reader proficient in one field to understand the parts of the thesis that cover other ones. The thesis is also intended for potential FENNEC users or developers, as the first introduction to the method and the related package.

Therefore, the thesis contains chapters the significant part of which summarizes the well established and fact from related fields. The purpose of these summations is not to give full and exhaustive details of the given topics, but just to provide a brief overview of key facts, results, algorithms, etc. of the given fields that are used.

That arrangement, by the way, reflects the character of my work on designing and implementing the new method: it also required mastering topics from different fields to find an efficient approach (either well known, requiring a new formulation, or even unknown in literature) to solve a particular problem or to overcome emerging difficulties, and to integrate these diverse pieces into a single functional method and a single functional program package.

To elucidate the motivation for the development of a new method, the thesis starts – in the first chapter – with a short survey of the existing methods for electronic structure calculations and their brief classification.

Chapter two summarizes the essentials of the density functional theory and its implementation. It contains the convergence analysis of mixing algorithms, including the results of a new modification of the Anderson mixing scheme and its comparison with other well established mixing schemes.

Chapter three explains the finite element method and its modification – isogeometric analysis. The chapter also describes the discretization of the Poisson equation, which is required for obtaining the Hartree potential during DFT computation, as an example of the application of the finite element method. Original results in this chapter include algorithms for designing meshes for finite element calculations (which are implemented in FENNEC) and convergence analysis for various finite element and isogeometric analysis bases.

In the fourth chapter, the Kohn-Sham equations — the fundamental equations of the DFT theory — are discretized using the finite element method. Numerical problems arising during the discretization are solved using separable pseudopotentials which are formulated using B-orthogonality. The chapter also contains the analysis of the required number of projectors in separable potential form and the comparison of two ways of constructing the projectors.

Chapter five is devoted to the total energy of the system and to the Hellmann-Feynman forces acting on each atomic center among other atoms which are the crucial part of the implementation of structural relaxations. The convergence properties of the newly developed formula for the force (useful e.g. for geometry optimization) are analyzed and compared with other known ways to express the force.

Chapter six offers the description of the method implemented in the software package FENNEC. The essential building blocks and main design features of the code are introduced, and the advantages of its open and expandable plugin architecture are discussed. Attention is also paid to the linear algebra solvers included in the package, mainly to the eigenvalue solvers for rank-k-update eigenvalue problem (which results from discretization using finite element method and pseudopotentials), whose performance is crucial for the overall performance of the package.

The last chapter demonstrates the use of the FENNEC package on the problem of a graphene fragment deformation. The positions of atoms in a deformed graphene sample are optimized using the FENNEC software package. These computations serve as proof of concept and verification of the capabilities of the FENNEC package. The obtained forces are used for constructing force-interaction molecular dynamics potentials in further research of deformed graphene layers.

1. Overview of methods for electronic structure calculations

Contents

In this chapter, a short survey will be dedicated to the existing methods for computational material science, primarily for calculating the electronic structure, and the motivation for the whole present work (i.e. why the new method is actually needed) will be explained. The more extensive summary of existing methods can be found e.g. in [1].

The first section in this chapter explains what are *empirical* and *ab-initio* methods: While the empirical methods, based on modeling the experimental data without ambitions to understand the physics behind, are computationally efficient and suitable for large scale problems under limited conditions, ab-initio methods based on the first principles of contemporary physics are much more computationally demanding but much more general, with a substantially wider scope, providing a deep insight into the mechanisms affecting material properties.

Because the newly developed method belongs to the latter class, the second section is dedicated to the properties and categorization of ab-inito methods: An elementary criterion is a way of treating the quantum character of electrons (indistinguishability of particles and the Pauli exclusion principle: methods based on the density functional theory (DFT) [2], Hartree-Fock and post-Hartree-Fock methods, Monte-Carlo methods, . . .).

From another viewpoint, the methods can be classified into (1) *reciprocal-space* (or *k-space*) methods, which assume that the system is formally infinite and conveniently make use of the translational symmetry, and (2) *real-space* methods which assume that the system is finite, do not rely on any symmetry requirements and allow for handling of extended systems at the cost of imposing additional boundary conditions.

Other classification criteria include the type of the *discretization basis* (general basis like plane waves vs. special restricted bases derived from atomic orbitals), or the treatment of core (non-valence) electrons (pseudopotential vs. all-electron methods).

The third section summarizes the cornerstones of the newly developed method introduced in this thesis (DFT, FEM, and pseudopotentials) and basic motivations for their choice. Moreover, it explains why the method is developed at all: Even if there is a great deal of well-established methods, the development of real-space methods was lagging behind the development of the reciprocal-space ones; hence, a truly ab-initio method with a general and in principle complete basis, i.e. with complete convergence control, has been missing among the real-space approaches.

This thesis aspires to fill this gap: the resulting method implemented in the new code FENNEC is aimed especially on precise and robust calculations of electronic structure and properties of the materials lacking translational symmetry (e.g. finite or non-periodic objects such as clusters, nanostructures or materials with defects), with possibly broken charge neutrality, that can be handled using many existing methods only in a limited way, at the cost of computationally expensive artificial constructions. Thus, such a new method can substantially expand the range of problems and materials that can be efficiently investigated using the ab-initio approach.

1.1 Empirical and ab-initio methods

Computational material science is aimed to predict the properties of materials (elasticity, hardness, magnetism, etc.) by means of calculations. A series of methods have been developed so far for this purpose. These methods can be classified within the range between two edges: (1) *empirical* and (2) *ab-initio*, methods.

The *empirical methods* are methods based on modeling the experimental data describing the behavior of the material in some respect. The experimental results are used as “magic” input parameters. E.g. for molecular dynamics [3], the model is based on molecules treated as small solid marbles interacting with each other by means of some kind of “springs”. It relies on “user-provided” atomic interaction potentials — forces between atoms as functions of their relative positions. As long as the potentials correspond to the real forces in the material under study, one can obtain relatively precise results. These methods are generally computationally efficient and suitable for large scale problems, but they are limited to the class of problems for which the particular method and its input parameters have been prepared and tuned, which is correct under certain conditions only. Otherwise, unpredictable errors can arise. E.g. the forces can depend not only on the positions of immediately neighboring atoms but also on the positions and state of other atoms, on charge transfer in the interatomic bonds, on external electric and/or magnetic field, on other conditions. So a small variation of the problem under study may require substantially different input parameters and the obtained results may be inaccurate or completely wrong.

Due to the limitations of the empirical methods, there is a need for more general and reliable methods not depending on “magic input parameters”. The *ab-initio methods*, not using any empirical models and based on essential laws of contemporary theoretical physics — specifically on the first principles of quantum theory — aspires to fulfill that requirement. They aim to describe the true state of the material — particularly the state of atom nuclei and surroundings electrons — in terms of quantum mechanics (i.e. wavefunctions) and to get a deep insight into the mechanisms affecting the properties of materials. Nowadays they are the most general and reliable methods for computational material science.

Note that the previous paragraphs are not meant as a criticism of the empirical methods. Both the empirical and the ab-initio methods are applied to problems of different sizes and so they do not compete with each other: what can be done with the one, can't be achieved with the other and vice versa. Moreover, for some empirical methods (e.g. the interaction-potential-based molecular dynamics mentioned above) the more computationally demanding ab-initio methods can be used as a source for the input parameters instead of an experiment (see chapter 7). So, the empirical methods are a perfect complement of the ab-initio methods and vice versa.

Besides those two limit approaches mentioned above, the *semiempirical methods* represent some kind of a mixture of the previous ones. Although such methods still employ the ab-initio approach, they substitute some elements of the exact equations by empirically obtained approximations. Although the results of these methods are still dependent on the selection of input parameters, they are based on physical principles and their applicability is usually wider than in the case of empirical methods.

In fact, even the methods denoted as truly ab-initio and not using any experimentally obtained input data rely on some approximations and need some input technical parameters. E.g., in practice, using some basis in terms of linear algebra can embrace just a finite and limited number of basis functions and so the basis cannot be complete. With improving computational capabilities and theoretical understanding of materials, the methods developed to more ab-initio shape. Therefore we regard most of older ab-initio-based methods as semiempirical nowadays. It concerns both the methods based on DFT and on the Hartree-Fock formalisms (e.g. implemented in the Spartan software package[1], the Pariser–Parr–Pople method [4], or methods based on empirical pseudopotentials).

1.2 Ab-initio methods — more about

As mentioned above, ab-initio methods are the most general and reliable methods for computational material science at present time. Contrary to the empirical methods, they do not replace the principles of physic by any empirical models. The goal of these methods is to determine the true state of matter — state of atom nuclei and surroundings electrons — within the quantum theory formalism.

Nearly all the methods of this kind relies on the Born-Oppenheimer approximation, which assumes that electrons and atom nuclei can be treated separately. The assumption is based on the fact that masses of particles in atom nuclei are much greater than masses of electrons, so the atom nuclei can be considered as stationary during the calculation of electronic states. From the Born-Oppenheimer approximation it follows that the main task for ab-initio methods in computational material science is to calculate the electronic structure of material — i.e., in terms of DFT, to determine the distribution of the electronic charge in the material; or, generally, to determine the multi-electron wavefunction describing the electronic states which can be obtained as a solution to the Dirac equation [5]. Because

[1]`http://www.wavefun.com/index.html`

it is computationally very demanding and the relativistic effects for the valence electrons are negligible due to their low kinetic energy (compared to the energies of core states), in real problems it is sufficient to solve the Schrödinger equation (i.e. the non-relativistic approximation) for the valence states instead of the Dirac one. The core states (i.e. the non-valence electrons) can be treated separately by means of the Dirac formalism in spherical approximation (i.e. using the assumption of spherical potential, which is computationally much less expensive and nearly correct since the core states are minimally affected by interatomic bonds). Resultantly, the ab-initio methods represent various approaches to solve these tasks by means of some computationally efficient approach, maybe using additional approximations.

One could object that such approximations (like negligence of relativistic effects and spherical approximation for the core states) are not better then approximations in empirical methods. But there is a substantial difference: Whereas both are approximations, in the ab-initio approach the nature and rate of the errors are known and the formalism is still grounded in the real physics. While in the empirical approach one must "trust" that the model and its input parameters have been chosen suitably for the material under study, the ab-initio method guarantees correct results under certain conditions: e.g. on condition that the kinetic energy of electrons remains low, the non-relativistic approximation is reasonable and the error is negligible. In the same way, the other simplifications (e.g. the errors of the discretization, etc.) can and should be evaluated (e.g. [6]); therefore the total error of an ab-inito method can be ascertained.

Even when using the approximations mentioned above, the ab-initio methods are much more complicated and computationally demanding compared to empirical methods. Even though the computational limitations are being constantly pushed forward (see e.g. [7]), a substantially smaller number of atoms (up to hundreds) can be treated. However, these methods allow us to calculate (in principle arbitrary) material properties as derivatives of the total energy (evaluated from the electronic structure, see the chapter 5) with respect to appropriate parameters. E.g. elastic constants are obtained as derivatives with respect to various deformations.

1.2.1 Classification of ab-inito methods

Many well or less established methods for ab-inito calculations of electronic structure have been developed. We summarize the main features of the methods in the following paragraphs.

Treatment of the quantum effects

The important aspect of a method is how it deals with the quantum character of electrons: indistinguishability of particles and the Pauli exclusion principle (see section 2.2 on page 24).

The most used approaches are (1) the *Hartree-Fock* formalism that approximates electron wavefunctions with *Slatter determinants* [1]; (2) its descendants the *Post-Hartree-Fock* methods; (3) the *Quantum Monte Carlo* methods that use probability integration algorithms for efficient evaluation of multi-electron equations [8]; (4)

the *density functional theory* [9] (commonly abbreviated as *DFT*) that we use throuhgout this thesis.

Periodicity: real and reciprocal space

The methods can be classified into *reciprocal-space* (or *k-space*) methods, and *real-space* methods. The former ones are designed for formally infinite periodic materials (like crystals), assume three-dimensional translational symmetry, and make use of the *Bloch's theorem* [10]. The latter ones, contrary to the previous class, assume that the system is finite, do not rely on any symmetry requirements, and allow for handling of extended systems at the cost of imposing additional boundary conditions.

The Bloch's theorem states that each electron wavefunction in periodic material has the following form:

$$\psi(\vec{x}) = e^{i\vec{k}\cdot\vec{x}}u(\vec{x}) \tag{1.1}$$

where $\vec{x}$ is a position in the real space, ψ is the wave function (denoted as Bloch wave), u is a periodic function with the same periodicity as the material, and the wave vector $\vec{k}$ is the crystal momentum vector.

The shape of wavefunctions according to Eq. 1.1 suggest naturally to apply the Fourier transformation to the problem, which transforms its (original) *real-space* formulation into the so called *reciprocal space*, alternatively denoted as *k-space* (by the exponent in Eq. 1.1) [10]. Accordingly the methods employing the Bloch's theorem are called *reciprocal(-space) methods* or *k-space methods*. Other methods, employing neither the Bloch's theorem nor the Fourier transform, are commonly denoted as *real-space methods*.

The consequence of the Bloch's theorem (1.1) is that the wavefunctions can be calculated just in the unit cell (the smallest cell of symmetry) of the reciprocal space called *first Brillouin zone*. Therefore, the periodicity of a material significantly simplifies the problem.

Discretization bases

Because the Schrödinger equation cannot be solved analytically, (perhaps) all methods use discretization to restrict the space of solutions to some subspace, spanned by a basis. Both in real and in reciprocal space we can choose many different bases for expressing the quantities we deal with.

As a reasonable choice, it seems to use a basis derived from the wavefunctions of isolated atoms, or a basis, that approximates the orbitals. Such a choice leads to fast convergence of the method with a low number of basis functions, but at the cost of significant restriction: the wavefunctions in complex materials are often very different from the wavefunctions of isolated atoms, which negatively affect the accuracy of the solution. If the basis is built up specifically to describe just the atomic orbitals, its capability to describe a general wavefunction affected by interatomic bonds is usually poor. Moreover, such bases are often non-orthogonal — which regards particularly the functions centered at different atomic sites — and it may be very difficult to extend to describe the true wavefunctions more precisely. These bases are also generally not spatially uniform, which brings difficulties in forces calculations, see section 5.2.1 on page 143.

There are methods from both the real-space and the reciprocal-space class which employ atomic orbitals in one way or another. As examples, it can be mentioned e.g. Fireball code using pseudoatomic orbitals [11], or Gaussian [12] that uses Gaussian bases that resemble orbitals, or — in the case of the reciprocal methods — e.g. many versions of LAPW [13] which augments plane waves in the interstitial region with a basis based on atomic orbitals in spheres around atoms. As other examples, we can mention various variants of LCAO (linear combination of atomic orbitals) that exist in both reciprocal- and real-space versions, or the muffin-tin approach used in LMTO method [14].

An alternative choice is to use a *general* basis: a basis, which is not related to a particular problem and which is a subset of a complete basis. We can obtain such a basis in a reciprocal space if we express $u(\vec{x})$ from Eq. 1.1 in a Fourier basis restricted by the periodicity condition of Bloch's theorem

$$u(\vec{x}) = \sum_{\vec{G}} e^{i\vec{G}\cdot\vec{x}}\,, \tag{1.2}$$

where $\vec{G}$ is the three-dimensional integer multiple of (symmetry) translational vector in the reciprocal space. According to Bloch's theorem, we can limit $\vec{k}$ in Eq. 1.1 to vectors from the first Brillouin zone. The resulting basis

$$e^{i(\vec{k}+\vec{G})\cdot\vec{x}} \tag{1.3}$$

is called *plane waves*. It is one of the most established basis for electronic structure calculations (e.g. [15]), and it is implemented in many software packages, e.g. in the ABINIT software package [16] or Castep [17]. For a real-space approach, *finite elements* [18, 19] bases plays a similar role in many fields of real-space calculations, however, no implementation for electronic structure calculations has been commonly accepted so far.

When using such a general basis, more basis functions are usually needed to achieve the same degree of precision, compared to bases specially fitted to a particular problem. On the other hand, this approach has considerable advantages. First, such a basis can be constructed with arbitrary precision and it can be easily extended if necessary; therefore it enables an excellent *convergence control.* Second, it enables truly general solution without forcing it into any "predefined shape", i.e. without anticipating any special wavefunctions — in contrast to the case of the basis composed from the wavefunctions of isolated atoms that usually speeds up the convergence of the method and substantially lowers the computational costs, but it can also spoil, distort or falsify the results. Consequently, the range of problems that can be solved using general-basis methods is wider.

The choice of a general basis instead of a special one can be seen as an analogy to the choice of an ab-initio method instead of an empirical one: the resulting method will be generally more computationally demanding, but also more robust, more reliable and precise and the basis will not have to be tuned up to a given problem.

Core electrons treatment

Some methods recalculate consistently both the *core electrons* and *valence electrons* in every iteration step (not necessarily treating both in exactly the same way): such methods are called *all-electron methods*. Other methods more-or-less make use the fact that the response of the core states to interatomic bonds is weak, lowering the computational demands; the most common way is using a *frozen core* approximation and/or (possibly) applying *pseudopotentials* (it should be noted that up-to-date pseudopotentials do not necessarily mean frozen-core). See section 4.2 for more details.

1.3 FENNEC cornerstones

This thesis introduces a new ab-initio method for electronic structure calculations, based on the density functional theory, finite element method (that provides a general real-space basis), and pseudopotentials (their advanced environment-reflecting version), and describes the implementation of that method — i.e. a software package FENNEC. Both — i.e. the method and the implementation — represent a novel, original work contributing to the current development of ab-initio methods, even if most of the ingredients of the method have been well known before. New, original components are (a) the algorithm for evaluating the Hellmann-Feynman forces, and (b) the adaptive mixing algorithm that speeds up the convergence of the DFT-loop. These components will be described later.

1.3.1 Density functional theory

Our approach is based on the density functional theory (DFT), which is probably the most common choice in computational material science. Since the Hartree-Fock methods often neglect the correlation energy term[2] in the Schrödinger equation (see section 2.3.3) and the Quantum Monte Carlo methods are too computationally demanding for large scale problems, it can be viewed as a sweet spot: it is accurate enough for common problem types and it is still able to calculate real-sized problems. However, it doesn't mean a claim to be the best method generally: there are many methods with many different properties that might aspire to justify such a claim. The users of these methods would point to many advantages of their respective approaches. Therefore no simple evaluation of methods is possible nor will be attempted in this thesis.

The choice of the DFT theory for FENNEC must be at least partially considered as a matter of taste, even though the taste is justified by its many advantageous properties:

[2]There are methods that follow the Hartree Fock approach and address the issue, they are denoted as post-Hartree Fock methods.

— DFT is theoretically precise in the sense that all errors due to used approximations can be estimated and bounded. It has no empirical assumption that could unpredictably spoil the results.[3]

— DFT enables modular implementation: some options (as e.g. spin or relativistic effects) may be easily switched on/off or they can be treated using more or less exact approximation. This property of DFT allows to further decrease computational expenses if a particular effect is not so significant for a given material and can be neglected/taken into account just approximately.

— Ratio of DFT computational demands and achieved precision is suitable for a large variety of problems.

— DFT is the well-established method and standard tool in solid-state physics with well known strong and weak points.

Of course, the DFT theory has also its weak points (e.g. poor treating of hydrogen bonds) and so it definitely can be considered neither as the only good method nor even as the best method. But the summarized properties and much more the up-to-now experience shows that it is at least one of very good choices for electronic structure calculations.

1.3.2 Other ingredients

For reducing computational demands, the new method uses pseudopotentials, which belong to commonly used approaches to overcome computational difficulties with core electrons. Their use not only cures the singularities of potentials of atomic nuclei and eliminates the need to treat relatively deep energies and high gradients of core electron wavefunctions, but moreover, it substantially reduces the required number of degrees of freedom in the DFT iteration loop.

Environment reflecting pseudopotentials [20] used by FENNEC improve the original ab-initio pseudopotential approach by the possibility of tuning the pseudopotentials for a given material structure, recalculating the core states selfconsistently with the current valence charge density. For further discussion of the topic, see chapter 4.

The finite element method is used since it provides a general (in principle complete) real-space basis and because it makes it possible to take advantage of robust, industrially proved, and continuously developing software. Generally, it is a very well established method with very good theoretical and software support. More about the reasons for this choice will be explained in chapter 3.

1.3.3 Why a new method at all?

For investigating non-periodic problems in material science, employing real-space ab-initio methods is more natural since the problems of this kind require dealing

[3]Some used approximations of exchange-correlation interaction can be considered as empirical, but their error is still bounded and can be estimated.

with a large number of atoms in an arrangement only little governed by symmetry (or not at all).

Even though also the non-periodic materials can be studied using reciprocal-space methods — adding an artificial translational symmetry (forming supercells), this approach substantially (manifold) increases the computational costs. It can be suitable in cases of materials with translational symmetry just along some axis/axes (concerns 1D materials like e.g. carbon nanotubes and 2D materials as e.g. graphene), however, even in these cases, the necessity to repeat the atomic structure under study in sufficiently distant incarnations to eliminate their interaction often makes the calculation very demanding or — in the case of broken charge neutrality — even computationally unattainable. Therefore, the scope of the applicability of this approach is limited.

Moreover, even some materials with basically periodic arrangement cannot be efficiently calculated using reciprocal-space methods e.g. in case of a defect in the material, or if an external force spoils the symmetry so that the material is non-periodically deformed.

As mentioned above, the approaches based on Bloch's theorem also encounter serious problems particularly if the charge neutrality of the sample under study is violated, since infinite repeating of such an object with electric charge results in an infinite charge in total.

The relative simplicity (in the sense of computational demands) of the reciprocal-space methods and, particularly, aiming at semiconductors with crystalline structures standing in the solid-state physics for a long period, are the reasons why the methods for periodic structures had been in a spotlight almost up to recent times and why much more effort has been made to them, compared to the real-space methods. Resultantly, there are many well-established methods for materials with translational symmetry that covers nearly all needs of solid-state physics, but the development of real-space methods suitable for non-periodic problems was lagging behind. A truly ab-initio real-space method with a general and in principle complete basis — something equivalent of what the plane-wave method is within the k-space class — has been missing among the real-space approaches, disregarding the new method described in this thesis. Recently, however, efforts to tackle this problem in various ways have intensified and some interesting attempts have been made, especially in recent years: [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55], most of them based on the finite element method, although also attempts based on isogeometric analysis and finite differences emerged as well. However, none of them have been accepted as a breakthrough in this field and so, the competition for the first successful general-purpose real-space method using a general basis is still open. It follows from the previous discussion that a new method for electronic structure calculation in non-periodic materials can substantially extend the capabilities of computational material physics as regards the scope of problems accessible within the ab-initio approach, and the growing interest focused on this field of material physics in recent years further confirms a proposition that there is a real need for a real-space code with a general basis.

2. Density functional theory

Contents

The main task of ab-initio calculations in computational material physics is to determine the electronic states in a material. The density functional theory — one of the most frequently used methods for this purpose — represents a tool for solving the multielectron Schrödinger equation by replacing it by one-electron Kohn-Sham equations [1]. Such an approach greatly simplifies the calculations by describing the interactions among electrons via potentials dependent on the electron charge density.

In this chapter, the fundamentals of the *density functional theory* (abbreviated as DFT), that were originally published by Hohenberg [56], Kohn, and Sham [9], will be presented. The purpose of the chapter is to introduce the key parts of the theory in a way that would facilitate their encoding by means of linear algebra, as discussed in a greater detail in subsequent chapters. This chapter does not aspire to present a full and detailed outline of the foundations of quantum mechanics or of the DFT. A more rigorous and complete expanation of the theoretical background can be found in the textbooks covering the topics such as, e.g., [10, 57, 58, 59, 60, 61, 62].

[1]As we will see, it is, in fact, only one equation, though, the equation is commonly called in plural since more its solutions are needed.

The first short section of this chapter gives a brief informal introduction into the formalism of quantum mechanics, just to present basic concepts of quantum mechanics for those, which are not proficient in this field of physics, and introduce a few terms, which will be needed in the later development.

In the next section, the stationary Schrödinger equation and basic properties of wave functions are discussed.

In the third section, the *density functional theory* that allows an efficient solution to the Schrödinger equation will be introduced.

The fourth section is dedicated to some important details of the implementation of the density functional theory and it summarizes the DFT algorithm for calculating electron wave functions.

The last section is dedicated to the mixing algorithm (used within the DFT algorithm), which is one of the crucial elements of the performance of the method. This section contains original results: (1) an analysis of convergence rate of various mixing algorithms employed in DFT calculations performed by means of the finite element method, (2) presentation of an adaptative scheme for the Anderson mixing, and (3) a comparison of the performance of the newly proposed algorithm with other approaches.

2.1 Selected elements of basics of quantum mechanics

The new method for electron structure calculation the presentation of which is the main topic of this thesis can be viewed as a mathematical model of a material, to compute its properties. The method is ab-initio, formulated within the density functional theory, that is a particular application of quantum mechanics.

For a reader unfamiliar with the topic, a very brief introduction into the quantum mechanics will be presented here, for the purpose of better understanding of this particular model and its implementation. Areader interested in a deeper understanding is referred to any of good textbooks, as e.g. [63, 62, 61].

2.1.1 Wavefunctions

In classical physics, a system can be described by its *state*: results of measurements of certain physical quantities performed on the system. E.g., for a system consisting of one particle, the respective set of measurements includes measurement of the position and of the velocity of the particle.

In quantum mechanics, the situation is different because certain physical quantities cannot be treated independently. A suitable description can be achieved if the state of a quantum system is represented by a vector in a specifically defined Hilbert space. So the state of a single particle is no longer determined by the set of coordinates and velocity components (as in classical mechanics), but by the state vector.

Often it is convenient to work in the so-called coordinate representation, where the state vectors are represented by quadratically-integrable functions, $L^2(\mathbb{R})$. The state vector is then commonly called a wave function. This terminology reflects some well-known properties of quantum systems such as interference phenomena. However, this is not our topic. For our purpose it is important that the probability that a quantum system (in our case, an electron) will be detected on a given position $\mathbf{x}$ is given by

$$P(\mathbf{x}) = \frac{|\psi(\mathbf{x})|^2}{\|\psi\|^2} \ , \tag{2.1}$$

The division by the norm of the function reflects the obvious requirement that the total probability of a system to be somewhere should be one. We can see, that even if the quantum theory is stated in complex numbers, only the magnitude of a result of a state vector determines the probability density.

2.1.2 Operators and potentials

In classical physics, one of the key terms is *potential*: a field that determines corresponding energies and forces in the system (e.g. electric field, gravitational field, etc.). In quantum mechanics, the term *potential* is understood more broadly. Whereas in classical physics potentials can be derived from forces, it is not possible in the quantum mechanics: there can be a potential, that causes physical effect even if no force is present, see the thought experiments in [64].[2]

In quantum mechanics, the potentials are represented by operators on the Hilbert space of state vectors [62]. Together with the term which describes the kinetic energy, the potential forms the Hamiltonian of the corresponding system. Eigenvalues of the Hamiltonian determine possible energies of the system (2.19).

In the following development, we will exploit some properties of operators. Therefore, we will define them formally here, for their later use.

Some definitions are formulated for a case of an infinite-dimensional Hilbert space (which is the playground for quantum mechanics) — in these cases, a similar definitions apply to a finite-dimensional vector space (which we later obtain by a discretization).

Locality

We have already mentioned that not all potentials can be expressed as a field. Among those, that cannot be described by a field, exists a special class of *nonlocal* operators:

Definition 2.1 (Local operator)**.** *An operator $V : \mathcal{L} \to \mathcal{L}$ on a Hilbert space $\mathcal{L}$ is* local*, if the value $V(\psi)$ in place $\vec{x}$ depends only on the value of ψ on an infinitesimal neighborhood of $\vec{x}$. A operator, which is not local, is* nonlocal*.*

This definition is fulfilled, if there exists a function v with a finite number of arguments that holds

$$\forall \psi \in \mathcal{L}, \forall \vec{x} \in \mathbb{C} : (V\psi)(\vec{x}) = v(\vec{x}, \psi(\vec{x}), \nabla\psi(\vec{x}), \nabla\nabla\psi(\vec{x}), \ldots) \,. \tag{2.2}$$

[2]This nowadays standard interpretation of the Aharonov-Bohm effect has been questioned by Vaidmain in [65].

Definition 2.2 (Pure local operator)**.** *Given a Hilbert space $\mathcal{L}$ over M, an operator $V : \mathcal{L} \to \mathcal{L}$ is* purely local*, if the value $V(\psi)$ in place $\vec{x}$ depends only on the value of $\psi(\vec{x})$.*

The definition[3] is equivalent to the condition that there exists a function $\upsilon : \mathbb{M} \times \mathbb{C} \to \mathbb{C}$, that holds

$$\forall \psi \in \mathcal{L}, \forall \vec{x} \in \mathbb{T} : (V\psi)(x) = \upsilon(\vec{x}, \psi(x))\,. \tag{2.3}$$

For a linear operator V, locality condition (2.3) can be rewritten such as there must exists a function $\upsilon : \mathbb{C}^n \to \mathbb{C}$, that holds

$$\forall \psi \in \mathcal{L}, \forall \vec{x} \in \mathbb{T} : (V\psi)(\vec{x}) = \upsilon(\vec{x})\psi(\vec{x})\,. \tag{2.4}$$

From this fact immediately follows, that just linear purely local operators can be expressed as a field: the application of the operator to a given function is just its multiplication by υ from (2.4)

$$V\psi = \upsilon(\vec{x})\psi\,. \tag{2.5}$$

While common external potentials (e.g. an electric field potential) are usually purely local, there exist local but not purely-local operators that cannot be expressed as a conservative field (e.g. the Laplace operator: the operator of kinetic energy), similarly as nonlocal potentials called pseudopotential (see chapter 4.2.2).

The locality is a substantial property of an operator: we will see in chapter 3.2.3, that matrices arising from the discretization of such operators has an undesirable property that prevents an efficient solution of discretized problems (using finite element method) containing such matrices.

Support

The definition of locality is sometimes confused with *small support*, which will be later needed for operators.

Definition 2.3 (Support)**.** *Given a function $\psi : \mathbb{V} \to \mathbb{C}$, support* $\operatorname{supp}(\psi)$ *of the function ψ is a set defined by*

$$\operatorname{supp}(\psi) = \{x \in \mathbb{V} : \psi(x) \neq 0\}\ . \tag{2.6}$$

Given an operator $V : \mathcal{L} \to \mathcal{L}$ on the Hilber space $\mathcal{L}$ of functions $\mathbb{V} \to \mathbb{C}$, support $\operatorname{supp}(V)$ *of the operator V is a set defined by*

$$\operatorname{supp}(V) = \{x \in \mathbb{V} : \exists \psi \in \mathcal{L} : V(\psi(x)) \neq 0\}\ . \tag{2.7}$$

Given definition of support, we can define *small support* property of a function or an operator

[3]In some literature is the term *local* reserved for *purely local* operators, whereas the *local* operators are called *semi-local.* We will use the above defined terminology, since the difference between local and pure-local operators will not be significant in our case.

Definition 2.4 (Small support)**.** *Let* $\| \, \|$ *be a meassure of a set on* $\mathbb{V}$*. Then function* $\psi : \mathbb{V} \to \mathbb{C}$ *has a* small support*, if*

$$\| \operatorname{supp}(\psi) \| \ll \| \mathbb{V} \| \,. \tag{2.8}$$

An operator V *on a space of such functions has a* small support*, if*

$$\| \operatorname{supp}(V) \| \ll \| \mathbb{V} \| \,. \tag{2.9}$$

While *support* talks about size of nonzero domain of a function or an operator, locality concerns only operators: if an operator acts non-locally, result of its application on a function depends not only on the value or the course (on the derivaives) of the function in given point, but on the whole function. Typical non-local operators are projections, as their evaluation typically involve integral. An extreme example of a function with a small support is the Dirac delta function.

A common simplification is e.g. to talk about a *local potential*, which should be understood as “a potential expressed by a local operator”.

2.2 The stationary Schrödinger equation

The state of electrons (their positions, momenta, etc.) in the material is determined by the electrons wave function ψ, which satifies the multielectron Dirac equation [5]. In many materials (and in the great majority of materials that are a subject of material physics and engineering) the speed of valence electrons is not close to the speed of light, therefore, relativistic effects for the electrons are often negligible and the Schrödinger equation can be used [66, 5].

We are interested in *stationary states*, which further simplifies the equation[57] to the following form of a eigenvalue problem:[4]

$$H_{\mathrm{S}}\psi(\vec{x}_1, \vec{x}_2 \dots \vec{x}_n) = e\psi(\vec{x}_1, \vec{x}_2 \dots \vec{x}_n) \qquad \psi : \mathbb{C}^{3n} \to \mathbb{C}, \quad e \in \mathbb{R}\,, \tag{2.10}$$

where n denotes the number of electrons. This form of the equation is called *stationary Schrödinger equation.* The operator H_{S} is the Hamiltonian, it will be discussed later. Each wave function $\psi_i(x)$ satifying the Schrödinger equation describes a stationary state of the system with energy e_i.

The state with the lowest energy is called the *ground state.* Other states are called *excited states.* The *degenerate* states are those with the same energy. Our task is to determine the ground state of a system.

Properties of electron wave functions

The electrons wave function must have the following properties:

- ▷ It is a common requirement (for spatially finite systems) that the wave function is normalized to unity,

$$\int |\psi|^2 = \|\psi\|^2 = 1 \quad \Longrightarrow \quad \|\psi\| = 1\,. \tag{2.11}$$

- ▷ According to the *Pauli exclusion principle* [67], the electron wave function must be antisymmetric,

$$\psi\left(\dots, \vec{x}_j, \dots, \vec{x}_k, \dots\right) = -\psi\left(\dots, \vec{x}_k, \dots, \vec{x}_j, \dots\right)\,. \tag{2.12}$$

The direct consequence is that

$$\psi\left(\dots, \vec{x}, \dots, \vec{x}, \dots\right) = 0 \tag{2.13}$$

and therefore no two electrons can occupy a same state.

[4]Without a loss of generality we will neglect spin of the electron in following discussion. The effect of spin will be taken into account in chapter 2.4.2.

Hamiltonian

The *Hamiltonian* H_S is a simple local operator (see Definition 2.1 on page 21).

To keep the matter as simple as possible, the Hartree atomic units will be used through this thesis (if it is not said otherwise): these units are designed so that equations frequently used have simple form. Important physical constants are simply one: the charge of an electron $e = 1$, the mass of the electron $m_e = 1$, the (reduced) Planck constant $\hbar = 1$). For more details see [58].

Using the Hartree atomic units, the Hamiltonian H_S can be written as sum of three operators:

$$H_\mathrm{S} = \frac{1}{2}\nabla^2 + U + V_\mathrm{ext}\,, \tag{2.14}$$

where $\frac{1}{2}\nabla^2$ is the operator of kinetic energy, U describe the electron-electron electrostatic interaction

$$U(\vec{x}) = \frac{1}{2}\sum_{i,j}^{n} \frac{1}{\|\vec{x}_i, \vec{x}_j\|} \tag{2.15}$$

and V_ext describe contributions of external potentials. If no external forces are present, V_ext is generated just by the electrostatic attractive force of atomic nuclei and it can be expressed as

$$V_\mathrm{ext}(\vec{x}) = V_\mathrm{ion}(\vec{x}) = \sum_{j=1}^{\text{number of atoms}} -\frac{Z_j}{\|\vec{x} - \vec{c_j}\|}\,, \tag{2.16}$$

where c_j denotes the position of atomic nuclei and Z_j its charge.[5] V_ext should also contain all other external forces, for example electromagnetic field.

Note that V_ion potential has a singularity in each atomic center. For the purpose of discretization (see the next chapter), it is also a substantial fact, that all operators in the Schrödinger equation are linear and local (see the definition of locality (2.1)).

Coupling of wave functions

The computational difficulty of the many-body Schrödinger equation follows from the fact that although the space of solution is a direct sum of “single electron spaces”:

$$\mathcal{L}^{3^n} = \overbrace{\mathcal{L}^3 \oplus \mathcal{L}^3 \oplus \mathcal{L}^3 \dots}^{n}\,, \tag{2.17}$$

the occurence of the electron-electron term U means that the solution cannot be *decoupled* to a product of one-electron wave functions

$$\psi(\vec{x}_1, \vec{x}_2 ... \vec{x}_n) \neq \prod_{j=1}^{n} \psi_j(\vec{x}_j)\,. \tag{2.18}$$

The consequence of the *coupling* is that the Schrödinger equation (2.10) is not solvable analytically, unless in a very special situation, and the cost of its numerical solution grows rapidly with the number of electrons in the system.

[5]By convention, electrons has positive and atomic nuclei has negative charge in electronic structure calculations.

The total energy of the Schrödinger equation

Since the solution of the Schrödinger equation (2.10) ψ is normalized to one, the energy of the system can be expressed as

$$\int_{\vec{x}^n} \psi^+(\vec{x}_1, \vec{x}_2 \dots \vec{x}_n) H_S \psi(\vec{x}_1, \vec{x}_2 \dots \vec{x}_n) = \mathcal{E}_\rho\,. \tag{2.19}$$

The energy $\mathcal{E}_\rho$ is called the *total energy* (of the system).

2.3 Fundamentals of the density functional theory

To overcome the complexity of the Schrödinger equation, more approaches have been developed (for their overview see e.g. [1]). The density functional theory (abbreviated as DFT) occupies a prominent place among them, since it is not an approximation, at least in principle, but it is equivalent to the original Schrödinger (2.10) equation.[6]

The DFT was derived by Hohenberg, Kohn, and Sham [56, 9]. Their approach is the preferred way how to introduce DFT (e.g. [57, 10]), I will follow it in the brief summary of the basics of the theory.

A more formal an approach to the DFT is outlined, e.g., in [68, 69], where the authors offer a more rigor way to develop the density functional theory, based on the Legendre transformation. The Legendre transformation map functionals of an external potential and the corresponding ground state density in a mathematically precise way, so the following theorems can be built from scratch on a stronger foundation and with a deeper insight into the problem.

2.3.1 The Hohenberg-Kohn theorems

For developing the density functional theory we state the well known variational principle [56]

Theorem 2.1 (Variational principle)**.** *The energy a ground state ψ is stationary on the space of all normalized wave functions*

$$\frac{\partial \mathcal{E}_\rho}{\partial \psi} = \frac{\partial \int \psi^+ H \psi}{\partial \psi} = 0 \tag{2.20}$$

and

$$\forall \psi' : \int \psi^+ H \psi \leq \int \psi'^+ H \psi'\,. \tag{2.21}$$

The inequality is sharp if the ground state is not degenerate.

[6]For practical calculations, the DFT introduces an unknown quantity — the exchange and correlation potential — which still has to be approximated. It is the ease and practicallity how these approximations can be introduced which make the DFT so popular.

Proof. The proof (e.g. [70]) can be obtained by expressing ψ in the basis of eigenfunctions of H where the energy of ψ' will be a normalized linear combination of eigenfunction energies. Since the ground state energy is the lowest eigenvalue of H, any "normalized sum" of non-degenerate states must have its energy greater equal than the energy of the ground state.

Moreover, since the function of energy has the minimum in the ground state, its derivative must be zero there. □

The key term of the DFT theory is the *charge density* ρ,

$$\rho(\vec{x}) : \mathbb{R}^3 \to \mathbb{R} = n \int_{\vec{x}_2} \dots \int_{\vec{x}_n} \psi(\vec{x}_1, \vec{x}_2, \dots \vec{x}_n)^+ \psi(\vec{x}_1, \vec{x}_2, \dots \vec{x}_n) \tag{2.22}$$

that express an amount of charge on the given place. Charge density of a ground state is abbreviated as *ground state density.*

Since the main properties of the density functional theory relly on *Hohenberg-Kohn theorems* we prove them here briefly.

Theorem 2.2 (The First Hohenberg-Kohn theorem)**.** *Given a ground state density ρ, the potential V_{ext} is determined uniquely by the density.*[7]

Proof. Assume that there exist two distinct potentials V_{ext} and V'_{ext} with the same ground density ρ. These potentials yield two distinct Hamiltonians H and H' and two distinct ground wave functions ψ and ψ'.

Since they yield the same density the Hamiltonian is linear and V_{ext} is purely local, we can write

$$\int \psi'^+ H \psi' = \int \psi'^+ H' \psi' + \int (V_{\text{ext}} - V'_{\text{ext}})\rho \tag{2.23}$$

$$\int \psi^+ H' \psi = \int \psi^+ H \psi + \int (V'_{\text{ext}} - V_{\text{ext}})\rho\,. \tag{2.24}$$

Moreover, consequence of the variational principle (the theorem 2.1) is that

$$\int \psi'^+ H \psi' > \int \psi'^+ H' \psi' \tag{2.25}$$

$$\int \psi^+ H' \psi > \int \psi^+ H \psi\,. \tag{2.26}$$

Summing 2.23 and 2.24 yields

$$\int \psi'^+ H \psi' + \int \psi^+ H' \psi = \int \psi'^+ H' \psi' + \int \psi^+ H \psi\,, \tag{2.27}$$

whereas summing 2.25 and 2.26 yields

$$\int \psi'^+ H \psi' + \int \psi^+ H' \psi > \int \psi'^+ H' \psi' + \int \psi^+ H \psi\,, \tag{2.28}$$

which is a contradiction. □

[7]We omit the discussion of a few special cases, as the possibility of degenerate states or of different potentials yielding the same ground-state wave functions, as these cases do not affect the following discussion and the application of the theorem.

A direct and a crucial consequence of the theorem is the fact, that a charge density determines also all wave functions and the total energy of the system. Therefore there is a bijection between a ground state density and a corresponding potential.

The second Hohenberg-Kohn theorem states variational principle for ground-state density and can be rephrased as (more exact formulation can be found in [70, 9])

Theorem 2.3 (The Second Hohenberg-Kohn theorem)**.** *The ground state density ρ minimizes the total energy over the space of all possible charge densities.*

Proof. Since by the first theorem the ground density yields unique potential and the potential yields unique wave function, the variational principle (the Theorem 2.1) completes the proof. □

The two of the most important theorems of the density functional theory states in plain words that the ground state density uniquely defines the state of the system. Therefore it is possible to express the state of the system and so each potential that appears in the system as a function of the ground state density.

2.3.2 Representing the system by noninteracting electrons

To derive the *Kohn-Sham* equations — the key equations of the DFT theory — let us consider total energy of a system without any electron-electron interaction. In such a system the Schrödinger equation can be decoupled to a one-electron equation (index zero denotes zero electron-electron interaction):

$$\left(\frac{1}{2}\nabla^2 + V_{\text{ext}}\right)\psi_i^0(\vec{x}) = e_i^0\psi_i^0(\vec{x})\,. \tag{2.29}$$

Since the Pauli exclusion principle (2.13) holds, electrons will occupy the first n solutions of the equation with the lowest energies. Thus, the charge density simplifies to

$$\rho(\vec{x}) = \sum_{i=1}^{n} \psi_i^0(\vec{x})^+\psi_i^0(\vec{x})\,. \tag{2.30}$$

The eigenvalues occuring in 2.29, i.e., the energies of single electron wave function ψ

$$e_i = \int \psi^+\frac{1}{2}\nabla^2\psi + \int \psi^+ V_{\text{ext}}\psi\,, \tag{2.31}$$

are called *Kohn-Sham energies.* They are not the same as the true energies of the "original" system of "interacting" electrons, even if they share many of their properties. This holds not only for this simple model case but for the later introduced Kohn-Sham equations 2.44, too.

The same holds for *total energy $\mathcal{E}_\rho$* of a system. If we sum the Kohn-Sham energies of the first n electrons with lowest energy, we obtain:

$$\mathcal{E}_\rho = \sum_i \left(\int \psi_i^+\frac{1}{2}\nabla^2\psi_i + \int \psi_i^+ V_{\text{ext}}\psi_i\right)\,. \tag{2.32}$$

However, due to missing electron-electron interaction, this “zero interaction energy” is only a crude approximation of the true total energy of the system of interacting electrons.

However, this issue can be resolved. The missing energy in (2.32) can be added by means of the two following term:

$\mathcal{E}_{\mathrm{H}}$ — **Hartree energy** energy of the electron-electron electrostatic force [58]

$$\mathcal{E}_{\mathrm{H}} = \frac{1}{2}\int_{\vec{y}}\int_{\vec{x}} \frac{\rho(\vec{x})\rho(\vec{y})}{\|\vec{x}-\vec{y}\|}\,. \tag{2.33}$$

$\mathcal{E}_{\mathrm{xc}}$ — **Exchange-correlation energy** that consists all the other missing contributions to the total energy (which will be discussed later).

Thus, corrected total energy expression will be

$$\mathcal{E}_{\rho} = \sum_{i=1}^{n}\Big(\int \psi_i^{+}\frac{1}{2}\nabla^2\psi_i + \int \psi_i^{+} V_{\mathrm{ext}}(\psi_i)\Big) + \mathcal{E}_{\mathrm{H}} + \mathcal{E}_{\mathrm{xc}}\,. \tag{2.34}$$

According to the first Hohenberg-Kohn theorem (2.2), the energies $\mathcal{E}_{\mathrm{H}}$ and $\mathcal{E}_{\mathrm{xc}}$ can be expressed as functionals of the ground state density

$$\mathcal{E}_{\mathrm{xc}} = \int_{\vec{x}} E_{\mathrm{xc}}(\rho)(\vec{x}) \qquad \mathcal{E}_{\mathrm{H}} = \int_{\vec{x}} E_{\mathrm{H}}(\rho)(\vec{x})\,. \tag{2.35}$$

Since the V_{ext} potential is always linear and purely local, its energy can be transformed as

$$\sum_{i=1}^{n}\int \psi_i^{+} V_{\mathrm{ext}}(\psi_i) = \int V_{\mathrm{ext}}\sum_{i=1}^{n}\psi_i^{+}\psi_i = \int V_{\mathrm{ext}}\rho \tag{2.36}$$

and the total energy of the whole system can be finally expressed as

$$\mathcal{E}_{\rho} = \sum_{i=1}^{n}\int \psi_i\frac{1}{2}\nabla^2\psi_i + \int V_{\mathrm{ext}}\rho + \int E_{\mathrm{H}}(\rho) + \int E_{\mathrm{xc}}(\rho)\,. \tag{2.37}$$

2.3.3 The Kohn-Sham equations

There is a simple observation that the integral of total charge density of a valid electron state of a system S must be equal to the number of electrons n in the system

$$\int \rho = n\,. \tag{2.38}$$

According to the variational principle (see theorem 2.1), the wave functions of the ground state are the ones with the lowest possible total energy. Therefore, they must satisfy the condition given by the application Lagrange multipliers l on equation 2.37 and condition (2.38):

$$\frac{\delta\Big(\Big(\int\sum_{i=1}^{n}\psi_i\frac{1}{2}\nabla^2\psi_i + \int V_{\mathrm{ext}}\rho + E_{\mathrm{H}}(\rho) + E_{\mathrm{xc}}(\rho)\Big) - l\left(\int\rho - n\right)\Big)}{\delta\rho} = 0\,. \tag{2.39}$$

By differentiating we obtain

$$\frac{\delta \sum_{i=1}^{n} \int \psi_i \frac{1}{2}\nabla^2 \psi_i}{\delta\rho} + V_{\mathrm{ext}} + \frac{\delta E_{\mathrm{xc}}(\rho)}{\delta\rho} + \frac{\delta E_{\mathrm{H}}(\rho)}{\delta\rho} = l\,. \tag{2.40}$$

The last two terms of (2.40) are usually called the *Hartree potential*

$$V_{\mathrm{H}}(\rho) = \frac{\delta E_{\mathrm{H}}(\rho)}{\delta\rho}\,. \tag{2.41}$$

and the *exchange-correlation potential* $V_{\mathrm{xc}}(\rho)$ (abbreviated as *XC-potential*)

$$V_{\mathrm{xc}}(\rho) = \frac{\delta E_{\mathrm{xc}}(\rho)}{\delta\rho}\,. \tag{2.42}$$

Now, let us do the same (2.39) variation for another system S' of non-interacting electrons with an external potential V'_{ext} equal to

$$V'_{\mathrm{ext}} = V_{\mathrm{ext}} + V_{\mathrm{H}} + V_{\mathrm{xc}}\,. \tag{2.43}$$

We obtain the same equation (2.40). It brings no help at the first sight since we cannot solve the equation. However, it follows from the equality of equations 2.40 for the two systems, that they share the same ground-state density and consequently wave functions. Hence, to obtain the correct ground state density for a system of interacting electron, it is sufficient to solve (2.29) for the system S' of non-interacting electrons in an external potential V'_{ext}.

The resulting system is called *Kohn-Sham equations*

$$H\psi_i = \left(\frac{1}{2}\nabla^2 + V_{\mathrm{ext}} + V_{\mathrm{H}} + V_{\mathrm{xc}}\right)\psi_i(\vec{x}) = e_i\psi_i(\vec{x})\,. \tag{2.44}$$

It is called in plural, since more its solutions are needed.

The ground state density of the original interacting electrons system can be expressed in terms of solutions of the Kohn-Sham equations as

$$\sum_{i=1}^{n} |\psi_i(x)|^2\,, \tag{2.45}$$

where i runs over the n occupied states with the lowest eigenvalues (see the next section for further discussion of occupied states). The ways of expressing the total energy within the DFT formalism will be further discussed in the chapter 5.

The potentials in Kohn-sham equations

We effectively transform the system of interacting electrons, which would have to be dealt with by means of a single multi-electron wave functio, into a system of non-interacting electrons, whose electron wave functions can be obtained independently as solutions of single-particle Schrödinger equation. Of course, there is a price we must pay for the simplification: we have had to include an exchange-correlation potential into the equation. Besides that, the electrostatic potential 2.15 transforms himself to the Hartree potential. Both the potentials are worth a brief discussion.

Hartree potential The Hartree potential is the potential of the electrostatic repulsion of the electrons in the system. It can be expressed as

$$V_{\mathrm{H}}(\rho) = \frac{\partial E_{\mathrm{H}}(\rho)}{\partial \rho} = \frac{1}{2} \frac{\partial \int_{\vec{y}} \int_{\vec{x}} \frac{\rho(\vec{x})\rho(\vec{y})}{\|\vec{x}-\vec{y}\|}}{\partial \rho} = \int_{\vec{y}} \frac{\rho(\vec{y})}{\|\vec{x}-\vec{y}\|} \,. \tag{2.46}$$

Equation (2.46) requires integration in each point where the potential is evaluated. It is computationally demanding, therefore it is desirable to use another formulation in practical computation. Since the Hartree potential is nothing else than classic electrostatic potential, it can be expressed as a solution of the *Poisson equation* [58]

$$-\frac{1}{4\pi}\nabla^2 V_{\mathrm{H}} = \rho \,. \tag{2.47}$$

This equation can be easily solved using an iterative solver.

Note that the expression (2.47) requires boundary conditions to be specified. However, since the number of the boundary points in the computation is much smaller than the number of the interior points, evaluating the boundary conditions directly from (2.46) does not significantly prolong the overall computational time.

For the later discussion, it will be substantial that the potential is linear and purely local (see Definition 2.1 on page 21).

The exchange-correlation potential The exchange-correlation potential is the "weirder" from the two. It describes two quantum interactions[8]: the exchange and the correlation.

The exchange arises from the antisymmetry of electron wave functions (see Eq. 2.13). The correlation "cures the error" coming from the fact that while the density approach computes with average positions of the electrons, the positions are correlated in reality: an immediate position of one particle affects the positions of the others [58].

Whereas exact exchange energy and exchange potential can be expressed analytically (e.g. [71]), there is still no analytical expression of the correlation one[9] — and so various approximations of the exchange-correlation energy and potentials are used in practical computations. The most used ones are the *local density approximation* (abbreviated LDA) [9] that approximates E_{xc} and V_{xc} as a function of the charge density, or the *generalized gradient approximation* (GGA) that uses both the charge density and its gradient [72, 73]. There exist relativistic exchange-correlation potentials that allow[10] sufficiently precise approximations of materials, where relativistic effects appear [74], and many more [75].

It will be important for the later development, that all the used approximations of V_{xc} are linear (in terms of their effect on a wave function) and local.

[8]In fact, the interactions are not real physical interactions mediated by a particle. They are "fictional forces" that must be introduced to enforce the laws of physics.

[9]Of course the correlation energy can be expressed as the difference between the energies of the exact Schrödinger equation and the Hartree-Fock approximation, that neglect the correlation energy, but such an expression is not usable for practical computation.

[10]In the combinations with a relativistic pseudopotential.

2.4 Practical implementation of the density functional theory

Equation (2.44) gives us the recipe for computing the wave functions, but there are still many subproblems, that need to be addressed in a practical implementation. The most crucial of them will be summarized in this section. The following approaches are common to many implementations of the density functional theory, so they have been only adapted to the finite element method case.

2.4.1 Self-consistency

The Kohn-Sham equations (2.44) have a substantially lower number of degrees of freedom than the original stationary Schrödinger equation (2.10), however, they have one principal drawback. The Hartree (2.46) and the XC (2.42) potential, that had to been included in the equations, are functions of charge density. However the charge density itself is obtained using the Kohn-Sham equations, thus we have to solve an instance of a "hen and chicken" problem.

This problem is commonly solved by *self-consistent cycle.* One starts with an initial guess of charge density (e.g. the sum of the densities of the isolated atoms or a uniform charge density). Next, the Hartree and XC potentials are computed from the *input charge density.* Then, using the potentials, wave functions are determined through Kohn-Sham equation(s) (2.44), from which the *output charge density* is computed using (2.30). Finally, the new input density is obtained by *so-called mixing algorithm.*

This sequence forms one so-called *DFT iteration.* The iterations are repeated in a cycle as long as the input and output densities differ. The convergence of the self-consistent cycle strongly depends (besides others) on the used mixing algorithm and its parameters. The next section (2.5) is dedicated to this topic.

2.4.2 Occupation of electron states

The equation 2.45 provides the expression for the ground state density. However, two additional aspects have been omitted in this expression and should be taken into account: (i) the existence of electron spin, which can split each state, and (ii) smearing the occupation numbers around the Fermi level, which is necessary to mitigate the instabilities during iterations.

Spin

Spin can be seen as a quantum equivalent of an intrinsic angular momentum of a particle. For an electron, its projection along a pre-defined direction attains values $\pm\frac{1}{2}\hbar$. Depending on this, one can speak about *spin-up* or *spin-down* electrons. [61] In many materials, we can assume, that the spins of wave functions are collinear. I.e. that there are no mixed states, one of the spinor components (spin-up or spin-down) is zero.

Spin density functional theory We have more options on how to deal with the spin. For our purpose the most natural and simple way is via the spin density functional theory. In this formalism, the energy of the system is function not only of the density but also of the spin polarization $\mu(\vec{x})$, i.e., the difference between the densities for spin-up and spin-down electrons:

$$\mu(\vec{x}) = \rho^{\uparrow}(\vec{x}) - \rho^{\downarrow}(\vec{x}) \ . \tag{2.48}$$

Starting with this, it can be shown that the wave functions for spin-up and spin-down electrons, $\psi^{\uparrow}$ and $\psi^{\downarrow}$, can be obtained by solving the Kohn-Sham equations but this time with different XC potential for the spin-up and for the spin-down electrons. The most common approximation for the spin-dependent XC potential is the local spin density approximation (LSDA) [76]. The solutions of spin-dependent Kohn-Sham equaitons are then merged into one set of wave functions and the first n wave functions according to their energies (eigenvalues) contribute to the charge and spin densities.

Spin-restricted computation As long as the material under study has no significant spin effects — primarily magnetism [10] — we can, from a computational perspective, assume, that there is no spin-dependent effect in the system, i.e., the XC potential does not depend on the spin density.

In this case, the Kohn-Sham equations look the same for spin-up and spin-down electrons and the respective wave functions are the same — as if there is no spin. However, the application of the Pauli exclusion principle has to be adjusted: it is neccesary to consider that for each solution of the Kohn-Sham equations, there exist two wave functions (one spin-up and one spin-down).

Smearing

Let us assume that we calculate an electron structure of a nitrogen atom using spin-restricted DFT. A nitrogen atom has five valence electrons: two 2s and three 2p, which occupy four valence orbitals (four spinless electron states that correspond to four solutions of the Kohn-Sham equation): one 2s and three 2p. Since each orbital can host two electrons (spin-up and spin-down), there are eight free spaces for valence electrons in these orbitals. The 2s orbital has lower energy and thus it will be always filled up (by two electrons). The 2p orbitals have the same energy (higher than the 2s one has), therefore, they have the same probability of a presence of an electron, and thus each of the orbitals should be occupied by one remaining electron.

However, if we calculate the energies e_i of the 2p orbitals using DFT, the energies will slightly differ due to numerical inaccuracies and inaccuracy in the input density. As a result — the 2p orbital with the lowest energy will be occupied by two electrons, while the others with higher energies will be empty. Moreover, the order of the orbitals (by energy) as well as their occupancy will be changing during the computation. Therefore, not only the obtained result will be wrong: the

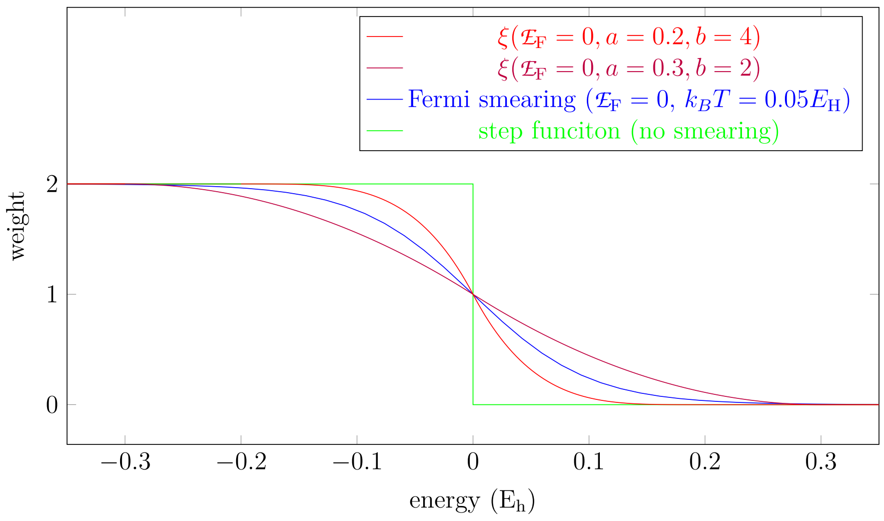


Figure 2.1: Examples of smearing functions for a spin-restricted computation. $\mathcal{E}_F$ is the Fermi energy that are the smearing functions constructed for. k_BT is a parameter for the Fermi smearing (the Boltzmann constant multiplied by the temperature).

changes in the occupancy of the states during the iterations typically[11] prevent the method from converging at all.

To cure this misbehavior, the *smearing* technique is introduced. Instead of a discrete division of the states to occupied and non-occupied, a non-growing [12] function $\xi : \mathbb{R} \to \mathbb{R}$ representing the occupancy of states is chosen so that (see Fig. 2.1)

$$\xi(-\infty) = f, \quad \xi(\infty) = 0, \quad \sum \xi(e_i) = n\,, \tag{2.49}$$

where $f = 2$ for spin-restricted computation and $f = 1$ otherwise. Occupancy of electron states, commonly named as *weights*, is given by

$$w_i = \xi(e_i) \tag{2.50}$$

There are many smearing techniques that differ by choosing the smearing function. For some examples (Fermi smearing, Gaussian smearing, etc.) see e.g. [78]. Note that the smearing can be viewed as assuming a nonzero temperature of the material which causes some electrons to be excited.[13] Some smearing functions (e.g. Fermi smearing) follow from this point of view.

[11]It is a basis dependent phenomena, in some bases that inherently retains the symmetry around an atomic site a computation without smearing is possible.

[12]In some smearing techniques, this restriction is lifted, see e.g. Methfessel-Paxton smearing in [77].

[13]It should be mentioned, that the DFT theory is proven only for a ground state, therefore this approach is disputable from the theoretical point of view. However, smearing is a standard and well-established approach in DFT calculations.

FENNEC uses the following approximation of Fermi smearing ξ (see Fig. 2.1) defined by

$$\begin{aligned}
\xi(x) &= f & x &< \mathcal{E}_\mathrm{F} - a \\
\xi(x) &= f((x - (\mathcal{E}_\mathrm{F} - a))/a)^b & \mathcal{E}_\mathrm{F} - a \leq x &< \mathcal{E}_\mathrm{F} \\
\xi(x) &= f(((\mathcal{E}_\mathrm{F} + a) - x)/a)^b & \mathcal{E}_\mathrm{F} \leq x &< \mathcal{E}_\mathrm{F} + a \\
\xi(x) &= 0 & \mathcal{E}_\mathrm{F} + a &\leq x \quad .
\end{aligned} \tag{2.51}$$

This approximation has the advantage, that outside the interval $(\mathcal{E}_\mathrm{F} - a, \mathcal{E}_\mathrm{F} + a)$ are the electron states either strictly occupied or unoccupied, which is convenient for computation.

The parameter $\mathcal{E}_\mathrm{F}$ is the *Fermi level*[14]: if the smearing is done by the step function (or by other words if the smearing is not done), the Fermi level would divide the occupied and the not occupied states: it is the "center" of the smearing function. Fermi level can be easily found by the bisect algorithm using the condition

$$\sum_n \xi(e_i) = n \ , \tag{2.52}$$

since the sum in (2.52) is monotone with regard to $\mathcal{E}_\mathrm{F}$.

The a, b are arbitrary parameters that define the width and hardness of the smearing. They can differ according to the solved problem, or they even can be varied during computation: a broader smearing width can be taken at first iterations of the DFT cycle to ensure the good convergence [15] and then narrowed to make the result closer to the ground state and so to lower the error of the computed total energy.

The charge density of a smeared wave functions Since the occupancy of the electron states is no longer discrete, we must properly modify the expression for the charge density:

$$\rho(\vec{x}) = \sum_i w_i |\psi_i(\vec{x})|^2 \tag{2.53}$$

2.4.3 The DFT algorithm

To summarize all the previous ideas, we can finally construct the algorithm for the density functional theory, see Algorithm 2.1 for the spin-restricted case.

However, even the simpler of the two equations in the algorithm, the Poisson equation (2.47), is not generally solvable analytically, so the work is not done yet: discretization, which is a necessary part of DFT implementation, will be the topic of the next chapters.

Now, let's focus on the problem of mixing.

[14]It is commonly denoted by E_F. Lowercase $\mathcal{E}_\mathrm{F}$ is used here to indicate its scalar nature since capital letters denote operators through this thesis.

[15]Such first cycles with wide smearing can be viewed as a way to obtain a good "guess" of the charge density for the later computing.

Algorithm 2.1: Spin-restricted density functional theory

```
1: $\rho_0$ = a guess of the charge density
2: **repeat** for $j = 1, \ldots$
3:     $\rho_j^{\text{in}} = \text{mix}([\rho_{j-1}^{\text{in}}, \rho_{j-1}^{\text{out}}], [\rho_{j-2}^{\text{in}}, \rho_{j-2}^{\text{out}}], \ldots)$
4:     solve $V_{\text{H}}$ from the Poisson equation: $-\frac{1}{4\pi}\nabla^2 V_{\text{H}} = \rho$
5:     $V_{\text{xc}} = \text{LDA}(\rho_j^{\text{in}})$
6:     solve the Kohn-Sham equations: $({}^1\!/\!{}_2\nabla^2 + V_{\text{ext}} + V_{\text{H}} + V_{\text{xc}})\psi_j = e_i\psi_i$
7:     $w_i = \text{smear}(e_i)$
8:     $\rho_j^{\text{out}} = \sum w_i(\psi_i \cdot \psi_i)$
9: **until** $\|\rho_j^{\text{out}} - \rho_j^{\text{in}}\| <$ threshold
```

2.5 Mixing algorithms

In this section, several mixing schemes will be compared and analyzed. The results will be used as a guide for suggesting a new Adaptive Anderson mixing scheme, which will be compared to several well-established schemes afterwards.

2.5.1 Definition of the problem

In terms of mathematics, the problem of mixing is a *fixed point* problem.

Definition 2.5 (Fixed point problem). *A fixed point of a function $F : \mathcal{H} \to \mathcal{H}$ is a point $\rho \in \mathcal{H}$ such that*

$$F(\rho) = \rho\,. \tag{2.54}$$

In our particular case, the function F is an operator on the Hilbert space $\mathcal{H}$.[16]. The operator takes a charge density ρ^{in} and calculates new charge density — *output density* ρ^{out} using $V_{\text{H}}(\rho^{\text{in}})$ and $V_{\text{xc}}(\rho^{\text{in}})$ potentials.

This fixed point problem cannot be solved analytically. Therefore, some kind of iterative process is required: this process is called the *self-consistent* cycle in the DFT terminology. The simplest approach is to take a reasonable guess of the initial charge density and then evaluate the function $F(\rho)$ repeatedly, taking the output of the previous iteration as the input for the next iteration until the process converges.

Definition 2.6 (Convergence). *A self-consistent cycle has been* converged*, if*

$$\left\|\rho^{in} - \rho^{out}\right\| < t\ , \tag{2.55}$$

where $t \in \mathbb{R}, t > 0$ is a given threshold.

[16]In practice, the operator is evaluated after a discretization (e.g. by the FEM, see chapter 3); so the problem could be expressed in terms of vectors, not functions. However, this distinction is insignificant for further development.

Sometimes the convergence is expressed in terms of a *relative convergence threshold*

$$\frac{\|\rho^{\text{in}} - \rho^{\text{out}}\|}{\|\rho^{\text{in}}\|} < t_{\text{REL}}\,. \tag{2.56}$$

A "reasonable guess" for the initial charge density can be given by a superposition of isolated atoms charge densities, by a charge density obtained from an alike configuration, from a faster and less accurate computation (e.g. with a lower number of basis elements), or even by a uniform charge density.

Unfortunately, the direct approach described above to solve the self-consistent problem leads generally to a poor convergence or (often) no convergence at all. Therefore, more sophisticated approaches have been found. One of them, in the simplest form, is the *linear mixing*: instead of the output density, a linear combination of the input and output densities is used as the input in the next iteration of the self-consistent cycle.

$$\rho^{\text{in}}_{i+1} = a_i \rho^{\text{out}}_i + (1 - a_i)\rho^{\text{in}}_i\,. \tag{2.57}$$

This approach could deaden the oscillations around the desired result. However, even the linear mixing might be unsatisfactory in many real problems. Therefore, more powerful mixing schemes are needed. To proceed formally, let us introduce the term *residual*:

Definition 2.7 (Residual)**.** *The i_{th}* residual *τ_i is defined as*

$$\tau_i = \rho^{out}_i - \rho^{in}_i\,. \tag{2.58}$$

Then (2.57) can be rewritten as:

$$\rho^{\text{in}}_{i+1} = \rho^{\text{in}}_i + a_i \tau_i\,. \tag{2.59}$$

By focusing on the residual, a new insight into the problem can be obtained: we try to find an input for which the residual is zero. Therefore we need to solve a multidimensional nonlinear *root finding problem*:

$$\text{Find } \rho \text{ such that } G(\rho) = 0, \text{ where } G(\rho) = F(\rho) - \rho\,. \tag{2.60}$$

Two significant issues are playing role in solving this problem. First, the evaluation of the function F (or G) is very costly, so we must keep the number of its evaluations as low as possible. And second, we have no possibility to evaluate the gradient of the function, which prevents us from using a wide range of gradient-based methods, from the classical Newton method to the well established BFGS methods [79].

The same problem, either in its fixed point or the root-finding form, appears in many fields and it has been studied by many authors. Their work resulted in several well-established algorithms for solving the problem, such as the Broyden's method and the Anderson or Pulay mixing schemes [80, 81, 82, 83]. Below, the algorithms will be described here just in a minimum extent. A rigorous mathematical survey of the methods can be found e.g. in [84].

Both Pulay and Anderson viewed the problem as a fixed point problem, which results in an iterative mixing scheme that takes into account a longer history of previous iterations. Anderson-like schemes calculates a new input density as a linear combination of previous input and output charge densities:

$$\rho_{i+1}^{\mathrm{in}} = \sum_{j=1}^{i} \left(b_{i,j}\rho_j^{\mathrm{in}} + a_i b_{i,j}\tau_j \right) \quad , \qquad \sum_{j=1}^{i} b_{i,j} = 1 \ . \tag{2.61}$$

The coefficients $b_{i,j}$ are selected according to the particular method (see below).

The important property of each particular method is the number of previous charge densities taken into account. This property is called the *history length*:

Definition 2.8 (History length (of a mixing method))**.** *If a method is allowed to have nonzero coefficients $b_{i,j}$ only for $j > i - n$, then the method has the* history length *n. The* available history length h_i in the i_{th} step[17] *is the number of $b_{i,j}$ allowed to be nonzero in that step.*

Thus, if a mixing scheme has a history length n, the new density is sought in the space of n previous densities (and residuals). See [83] for an analysis of the influence of the number of used previous densities on the convergence of the DFT cycle.

2.5.2 Overview of commonly used algorithms

In addition to the already presented linear mixing, the following basic algorithms are used nowadays.

Anderson and Pulay mixing Today, the *Pulay mixing* and the *Anderson iterating scheme* are essentially the same algorithm viewed from various sides: Pulay [80] has introduced the scheme, where the scalars $b_{i,j}$ minimize the L^2 norm of the residual over a subspace of n last input densities

$$\min \left\| \sum_{j=1}^{i} b_{i,j}\tau_j \right\|_2 \ . \tag{2.62}$$

However, only the space spanned by the input densities is taken into account in this scheme, which is effectively the same as setting all a_i to zero.

The constrained minimization is usually done by employing Lagrange multipliers, which leads to a low-dimensional system of linear equations [85]. This mixing was named after Pulay but he was not the first who invented the method: the same method in a bit different context had been published earlier by a less known (at least in the physics science community) scientist with Czechoslovak roots Márian Mešina [86].

In his work [82] (which is earlier than the work of Mešina), Anderson suggested mixing with nonzero a_i, but his approach used only the history length two.

[17] h_i can differ from n. E.g. for a mixing scheme with any history length n, the history length h_2 (in the second iteration) is always equal to one, since only the residual for the initial charge density is available.

Both approaches — Pulay's and Anderson's — evolved: their advantages (nonzero a_i and a longer history) have been combined to form a single algorithm called either the *Pulay mixing*, *DIIS* (Direct inversion of the iterative subspace), *Anderson mixing* or *Anderson iterative scheme.*

Broyden methods Broyden's work, contemporary with the Anderson's one, was aimed at a root-finding problem. Broyden introduced a modification of the well-known Newton method for solving nonlinear equations.

One iteration of the Newton method is given by

$$\rho_{i+1} = \rho_i - \frac{\tau_i}{J_G(\rho_i)} \tag{2.63}$$

where $J_G(\rho_i)$ is the Jacobian[18] of function G. Broyden's methods replace the Jacobian with its approximation: *Broyden's first method* constructs the Jacobian approximation and then inverts it, while *Broyden's second method* constructs directly the inversion using the Sherman-Morisson-(Woodbury) formula (see Theorem 6.1 on page 183) [87].

Generalized Broyden methods Broyden's work laid the foundations of the theoretical background of a class of methods called *generalized Broyden's* or *multi-secant methods*, which follow the Broyden's idea of approximating the Jacobian, but unlike the original Broyden's methods they use a history longer than two [81].

It can be shown [84] that not only the original Broyden's methods belong to this class of methods, but the Pulay/Anderson mixing belongs there too.

The Anderson mixing is nowadays the de facto standard in DFT calculations, and — as we can see in Fig. 2.2 — it performs substantially better than the Broyden's original methods. Therefore, we will focus especially on the Anderson method and its modifications.

Relatively recently, several papers were published that suggest a modification of the Pulay mixing scheme using "non-Broyden" approaches. These are represented by the periodical Pulay method and the guaranteed residual Pulay method.

Periodical Pulay The more recent one is the *Periodical Pulay method.* Its idea is simple: to alternate linear (2.57) and Anderson (2.61) mixing steps to achieve a better convergence [88]. However, it performs slightly worse than the Anderson method in our numerical examples (see Fig. 2.2).

Guaranteed residual Pulay method The second mentioned modification, the *Guaranteed residual Pulay method* [89] (abbreviated as *GR-Pulay*), shares the idea of alternating linear and Anderson mixing steps with the Periodical Pulay method. However, all linear steps are performed with $a_i = 1$ and all Anderson steps are performed with $a_i = 0$ in this scheme.

[18]The Jacobian of a function is the matrix of its partial derivatives: $J^f_{a,b}(\mathbf{x}) = \frac{\partial f_a}{\partial \mathbf{x}_b}(\mathbf{x})$ (we assume the finite dimensional case for the sake of simplicity).

Moreover, the result obtained by each linear steps are discarded from the history just after performing the next "Anderson step" which effectively replaces it. Thus, only the results of the Anderson steps and the last linear step are taken into account in each Anderson step.

Authors of the GR-Pulay method suggest that the whole "linear-Anderson multistep" should be considered as a single iteration. However, as it is the number of function evaluation that determines the performance of the scheme, we will consider each evaluation of a new charge density to be one iteration. Moreover, when performing our numerical tests (Fig. 2.2), we will consider a generalization of the concept, in which linear steps are done with varying values of a_i (Anderson steps are still done with $a_i = 0$), which sometimes brings a slightly better convergence.

Other mixing-related approaches

The efficiency of the mixing can be further enhanced using the following techniques.

Position dependent mixing So far we considered a_i to be the same for the whole charge density vector. In fact, the a_i can be different for each component of the vector: from the "Broyden's point of view" it follows (see (2.63)) that $-1/a_0$ should approximate the Jacobian of the function [84]. Therefore, for a heterogeneous material, setting the parameters independently in different parts of the domain can be a viable approach. A similar approach was published by Lin and Chao, who suggested using a spatially dependent preconditioner [90]. Heide and Ono demonstrated that a substantial improvement of convergence can be achieved by using different mixing schemes for different parts of the state vector [91].

Preconditioning The preconditioning techniques allow a better approximation of the Jacobian than the constant $-1/a_0$. Several preconditioners have been developed, e.g. Kerker preconditioner [92, 93], elliptic preconditioner [90] or Resta preconditioner [94]. However, all those preconditioners are material dependent and thus cannot be used as a general recipe for all problems.

Potential mixing The mixing of the charge density is not the only option for solving the self-consistent problem. Let's recall the First Hohenberg-Kohn theorem (on page 27) which states that the ground charge density is the function of the potential and vice versa. Therefore, taking the potential ($V_{\mathrm{Hxc}} = V_{\mathrm{H}} + V_{\mathrm{xc}}$) as a "state of the system" and iterating it instead of the charge density represents a viable choice.

The exploratory calculations performed by means of the electronic structure code SPRKKR [95] shows that in some cases this approach could lead to a better performance than mixing the charge density (see Fig. 2.2).

In our early tests performed by means of the FENNEC code, this approach did not provide any advantage in comparison with mixing the density. Therefore, the option to mix the potential instead of the density has not been maintained in the FENNEC codebase (when the code was updated for spin-polarized calculations).

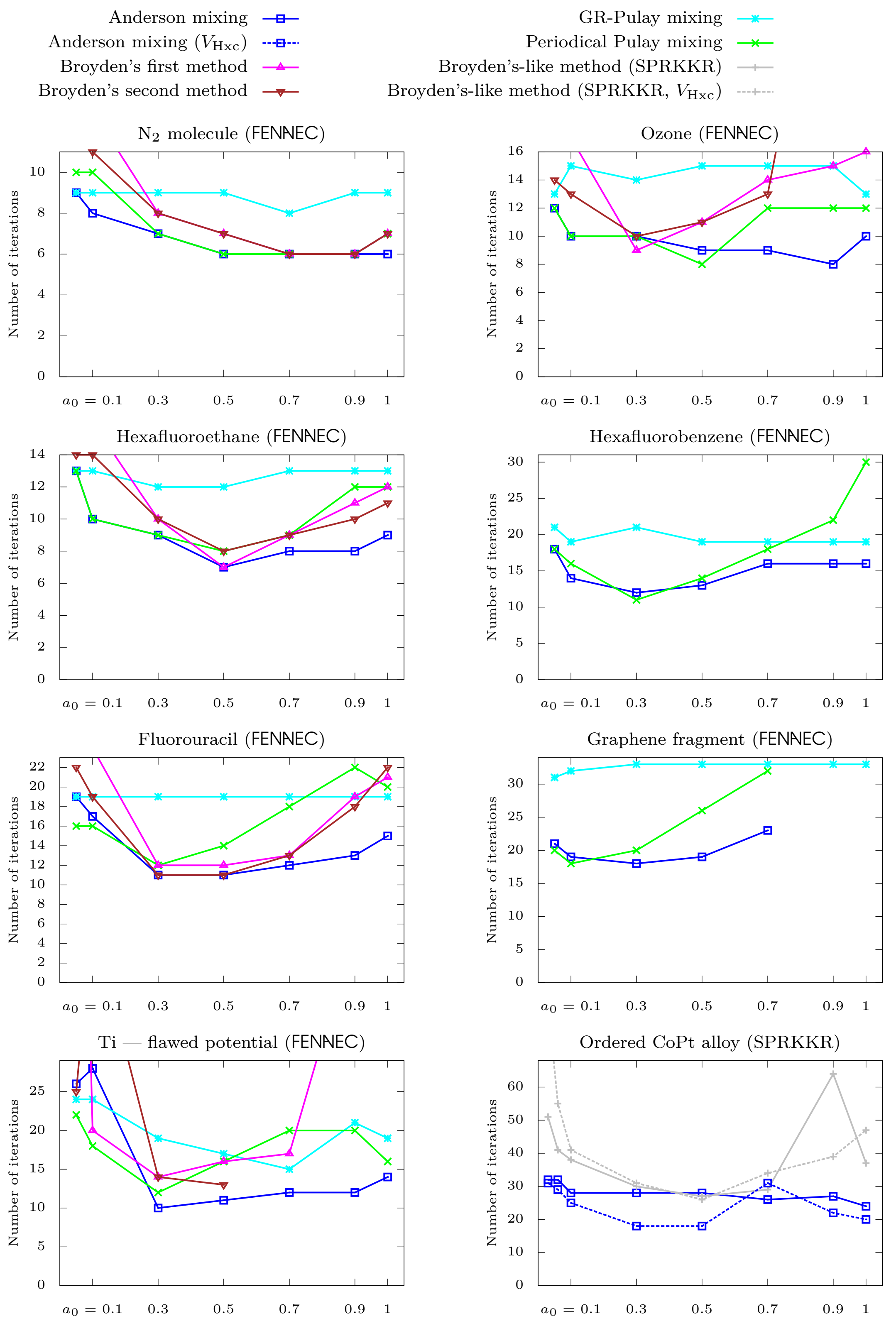


Figure 2.2: Dependence of the number of iterations, required to converge the DFT self-consistent cycle, on the a_0 coefficient, for various mixing algorithms. Two codes were employed (FENNEC and SPRKKR [95]). The convergence criteria were $t_{\mathrm{REL}} = 1 \times 10^{-9}$ for FENNEC and $t_{\mathrm{REL}} = 1 \times 10^{-8}$ for SPRKKR (see (2.6) and (2.56)). Algorithms denoted as V_{Hxc} mix the V_{Hxc} potential. Other algorithms mix the charge density. Missing value means that the self-consistent calculation has not been converged. The system for which the FENNEC calculations were performed is quoted in the title of each graph (using an adaptive hexahedral mesh and cubic Lagrange elements). The SPRKKR calculation was performed for the ordered magnetic CoPt ($L1_0$ structure).

This functionality has been reintroduced in response to the results of this study, however, the results have not been evaluated yet.

2.5.3 Analysis of the mixing algorithms and the parameter a_0

So far we have not discussed the *mixing parameters* a_i. All algorithms introduced up to now keep it constant during iterations $(a_i = a_0)$[19], and the value of a_0 has to be supplied by the user.

Therefore, a suitable value for a_0 should be determined. For this purpose we determined, for various systems, the numbers of iterations needed to converge the self-consistent calculations by the FENNEC code and, for comparison, also by the SPRKKR code for ab-initio electronic structure calculations [95]. In FENNEC, we use implementation from SciPy package [96]. Implementation of Broyden's method in SPRKKR is included in the package. The other mixing schemes is my own implementation. The performance of the Anderson mixing has been compared with its SciPy implementation: they yield the same results, which confirms the correctness of my implementation.

The results are summarized in Fig. 2.2. We can observe that the convergence depends on the value of a_0. Since a_0 is related to the approximation of the Jacobian, it is not surprising that the optimal values for a_0 are equal or at least close to each other for various mixing algorithms. The algorithms differ more in how they are able to handle a non-optimal a_0 than in the minimal achieved number of iterations or in the value of an optimal a_0. If the parameter a_0 differs from its optimal value, the self-consistent algorithm converges slowly or even does not converge at all.

The only exception is the GR-Pulay scheme, which seems to be much less sensitive to the value of a_0 than other schemes. However, its convergence is overall quite slow. This behavior of the GR-Pulay mixing is rather expectable since it "wastes" half of its iterations on finding the best next guess. This approach is capable of eliminating bad guesses but the extra iterations it requires mean that it is not competitive with other algorithms if they are supplied with a good value of a_0.

We can conclude that the crucial question is not which mixing scheme to pick but how to pick the right a_0 for a given problem. If we pick a proper value of a_0, the classical Anderson scheme performs best (at least for the kinds of problems we investigated).

Moreover, for more complicated examples, having a good estimate of the value of a_0 seems to be necessary for achieving a reasonably quick convergence or even to have the calculation converged at all (see Graphene fragment in Fig. 2.2).

Unfortunately, though some preconditioning techniques described above exist, there is no general prescription on how to choose the mixing parameter a_0 to obtain the best possible convergence. Thus, a_0 is often viewed as a kind of a "magic parameter". I present here a new original approach which is aimed at determining suitable mixing parameters by adapting the values of a_i during the iterations.

[19]The parameter a_0 is often called α.

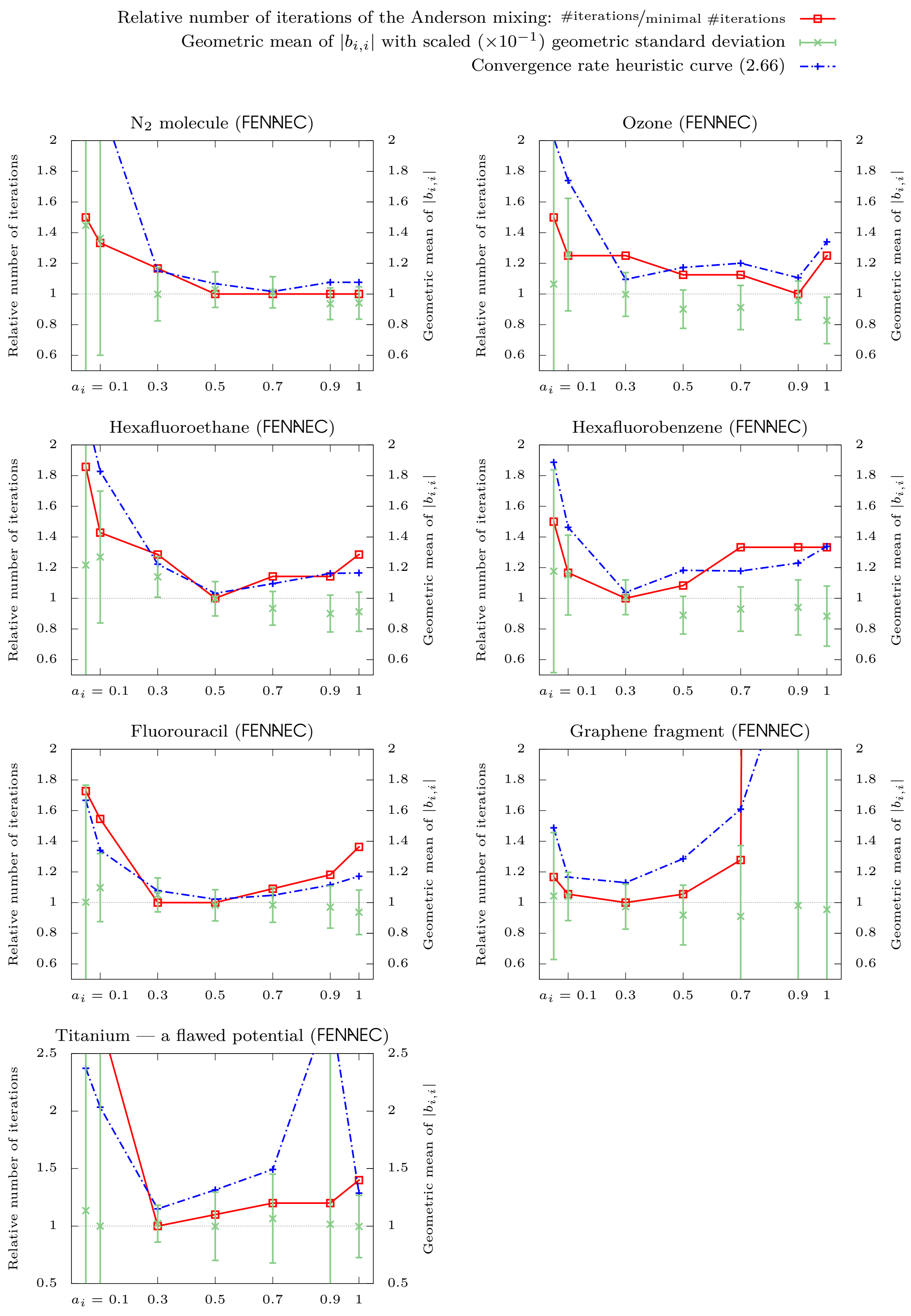


Figure 2.3: Influence of the coeficients $b_{i,i}$ on the number of iterations needed to achieve convergence for the Anderson mixing. $b_{i,i}$ are given in terms of their geometric mean with geometric standard deviation and in terms of the heuristic convergence parameter (2.66).

2.5.4 The Adaptive Anderson mixing

The coefficients $b_{i,i}$ in the equation (2.61) express the amount of the last input density and residual in the next input density. The observation crucial for our further development is that the convergence of the Anderson mixing scheme correlates with how these coefficients are close to one.

For further reasoning we introduce their geometric mean b across all the previous iterations:

$$b = \sqrt[n]{\prod_i \operatorname{abs}(b_{i,i})} \tag{2.64}$$

We use geometric mean, since $b_{i,i}$ is used for multiplication in (2.61) and thus we are interested in the mean with respect to the logarithmic scale. Geometric standard deviation is then defined as

$$\operatorname{GSD}\left(\{b_{i,i}\}\right) = \exp\left(\sqrt{\frac{\sum_i \ln\left({}^{b_{i,i}}/_{b}\right)}{n}}\right). \tag{2.65}$$

A large value of $\operatorname{GSD}\left(\{b_{i,i}\}\right)$ indicates large oscilations of the values of $b_{i,i}$ around the mean b.

If the coefficients $b_{i,i}$ differ significantly from one, as indicated either by the mean b or by the presence of large oscillations indicated by large $\operatorname{GSD}\left(\{b_{i,i}\}\right)$, the convergence of the self-consistent cycle is poor: compare the green geometric means of $|b_{i,i}|$ with the red curves in Fig. 2.3.

Using this observation, the following convergence criterion was constructed: the convergence is the better the lower is the heuristic convergence parameter c.

Definition 2.9 (Heuristic convergence parameter)**.** *For a given Anderson self-consistent cycle, the* heuristic convergence parameter *is a real number c defined as*

$$c = \max\left(b, \frac{1}{b}\right)\sqrt[4]{\operatorname{GSD}\left(\{b_{i,i}\}\right)}. \tag{2.66}$$

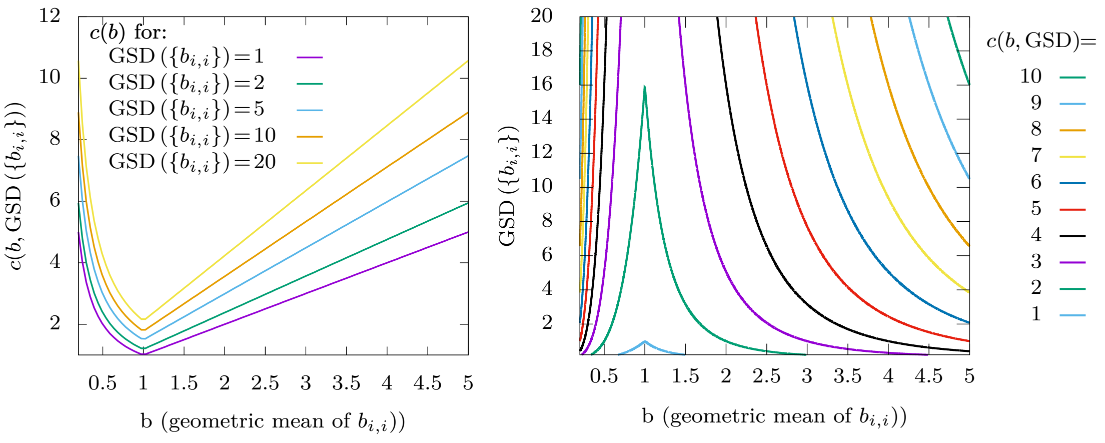


Figure 2.4: Visualisation of the convergence heuristic parameter $c(b, \operatorname{GSD}(\{b_{i,i}\}))$: the function values for given GSDs (left) and the contour lines of the function (right).

This heuristic convergence parameter (see its visualisation in Fig. 2.4) was constructed empirically. However, it seems that it can predict the convergence rather well: see the agreement of the red and blue curves in Fig. 2.3.

Our research group is going to explore this dependence in a mathematically more rigorous way in further work. However, an intuitive explanation can be already formulated. The suggested adaptation of a_i is just a heuristic: the following development does not claim that the described causal links are valid under all circumstances. It just describes the correlations that commonly occur, and this way it attempts to explain why the suggested adaptation scheme works usually well.

Let's recall (2.61): for $b_{i,i} \approx 1$, the new density ρ^{in}_{i+1} can be written as

$$\rho^{\text{in}}_{i+1} \approx \rho^{\text{in}}_i + a_i \tau_i + \sum_{j=1}^{i-1} \left(b_{i,j} \rho^{\text{in}}_j + a_i b_{i,j} \tau_j \right) \ , \quad \sum_{j=1}^{i-1} b_{i,j} \approx 0 \ . \tag{2.67}$$

Therefore, the latest iteration forms the major part of the new input density. Its other components (the sum in (2.67)) can be viewed as small corrections of the mix of the latest input and output density. Since the coefficients $b_{i,j}$ minimize the residual, the last step has lowered the residual more than the previous steps: the self-consistent cycle converges well. This is illustrated by the red lines in Fig. 2.5.

If $b_{i,i}$ is substantially lower than one, then the substantial part of the new input density is formed by the previous steps, not the last one. It can have two reasons: Either the latest residual has a larger magnitude then the previous ones — in this case the latest input density ρ^{in}_i is too far away from the desired solution (see the cyan case in Fig. 2.5). Or the lastest residual can be (at least partially) canceled out by a previous one (the blue case). In both cases, the latest guess

$$\rho^{\text{in}}_i = \rho^{\text{in}}_{i-1} + a_{i-1} b_{i-1,i-1} \tau^{\text{in}}_{i-1} + \dots \tag{2.68}$$

overstepped the solution[20]: the amount of residua added to ρ^{in}_i is too large. Decreasing a_i, we would obtain a better guess.

The opposite case, $b_{i,i} \gg 1$, can be explained using similar heuristic argumentation. If there existed τ_j of an opposite direction and a similar magnitude as τ_i, the situation described in the previous paragraph would occurred. Thus, we can assume that there is no such one. Moreover, since $\sum_j b_{i,j} = 1$, there must exists a negative $b_{i,k}$. Since the coefficients $b_{i,k}$ minimize (2.62), the τ_i and τ_k aim in a similar direction (they have to cancel themselves at least partially). Therefore, $b_{i,i} \gg 1$ indicates that we approaching the solution still from the

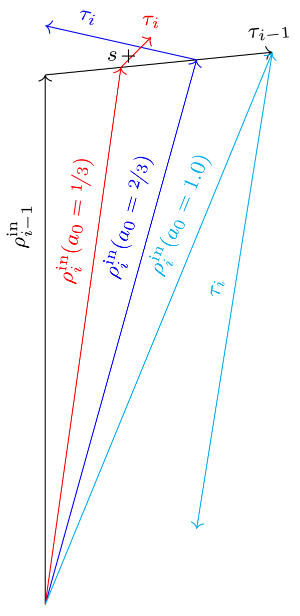

τ_i
τ_i
τ_{i-1}
s
ρ^{in}_{i-1}
$\rho^{\text{in}}_i(a_0 = 1/3)$
$\rho^{\text{in}}_i(a_0 = 2/3)$
$\rho^{\text{in}}_i(a_0 = 1.0)$
τ_i

Figure 2.5: An overestimation of a_0 leads to overstepping of the desired solution s and to oscillations.

[20]We assume that the residuals are pointing (at least roughly) towards the solution; fortunately, the opposite case is not common in DFT calculations.

"same side" and thus we should enlarge a_i to speed up
the convergence.

From these observations, a heuristic scheme for adapting the a_i coefficient during the Anderson iterations can be derived: if $b_{i,i}$ is greater than one, increase a_i, whereas if $b_{i,i}$ is smaller than one, decrease a_i. We call this algorithm *Adaptive Anderson scheme* onward.

The adaptation of a_i could (and for "reasonable" functions would) brings the values of $b_{i,i}$ close to one, since the adaptation is in fact approximating of the Jacobian of the function. We can demonstrate this link between Jacobian and the coefficient $b_{i,i}$, if we approximate the function by the following linear function G: let G is a function defined as

$$G(\rho) = d\tau_\rho = d(\rho_{\text{root}} - \rho) \qquad d \in \mathbb{R}\,, \tag{2.69}$$

for which we are looking for its root ρ_{root}. Application of the Anderson mixing (2.61) with a given fixed $a_i = a_0$ and with $\rho_0^{\text{in}} = 0$ yields

$$\rho_0^{\text{in}} = 0 \qquad\qquad \tau_0 = G(\rho_0^{\text{in}}) - \rho_0^{\text{in}} = d\rho_{\text{root}} \tag{2.70}$$
$$\rho_1^{\text{in}} = a_0 d\rho_{\text{root}} \qquad\qquad \tau_1 = G(\rho_1^{\text{in}}) - \rho_1^{\text{in}} = d(1 - a_0 d)\rho_{\text{root}}\,. \tag{2.71}$$

To obtain ρ_2^{in}, we have to minimize (2.62), thus it have to be satisfied

$$b_{1,0} + b_{1,1} = 1 \qquad\qquad b_{1,0} = -(1 - a_0 d) b_{1,1}\,, \tag{2.72}$$

which yields

$$b_{1,1} = \frac{1}{a_0 d}\,. \tag{2.73}$$

Let's recall that $-a_i{}^{-1}$ should be an approximation of the Jacobian of G. Since we apply the Jacobian on residuals, its best approximation is the derivative of G in their direction.

$$a_i \approx -\frac{1}{J_G} \approx \frac{1}{d}\,. \tag{2.74}$$

If we substitute (2.74) into (2.73), we obtain

$$b_{1,1} = 1\,. \tag{2.75}$$

Thus, the effort to keep the $b_{i,i} = 1$ through adaptation of a_i is in fact approximating of the Jacobian of the function by the (implicitly) obtained (directional) derivative of the function. Moreover, the relation of a_0 and $b_{i,i}$ in (2.73) indicate how to suggest the adaptation scheme for a_i:

$$a_i = a_{i-1} b_{i,i}\,. \tag{2.76}$$

However, for a real use, (2.76) should utilize some form of a damping to prevent oscillations of a_i. The details of my initial approach for a suitable adaptation function will be presented later.

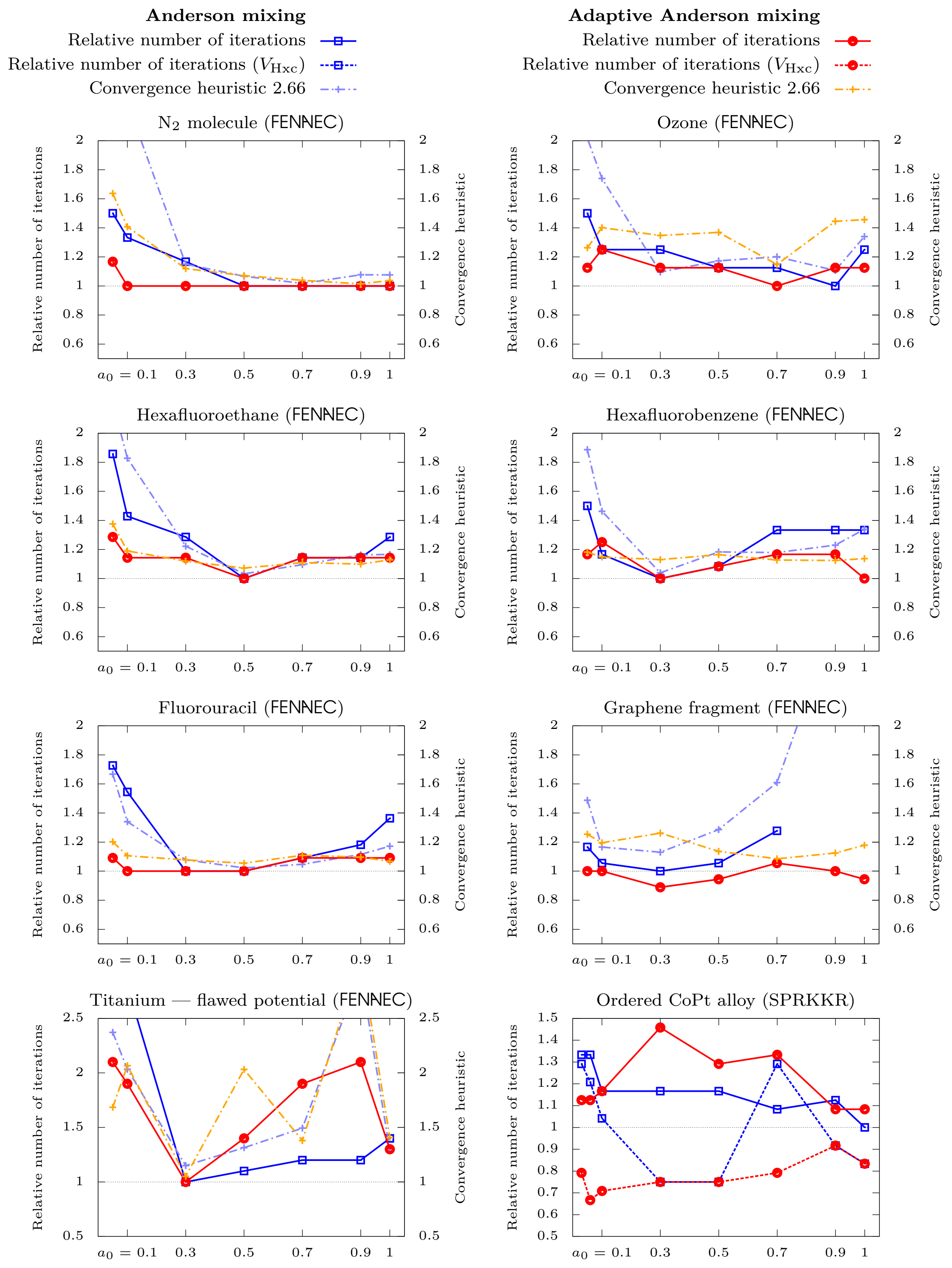


Figure 2.6: Comparing the efficiency of the Anderson mixing and of the Adaptive Anderson mixing: each plot depicts the relative (with regard to the achieved minimal number of iterations for Anderson mixing) number of iterations, required to converge the DFT self-consistent cycle for a specific system, in dependence on the a_0 coefficient.
The codes employed are FENNEC and SPRKKR [95]. The relative convergence criteria were 1×10^{-9} for FENNEC and 1×10^{-8} for SPRKKR. The algorithms denoted V_{Hxc} mix the potential, other algorithms mix the charge density. Missing values denote that the DFT cycle has not been converged. Dash-dotted lines present covnvergence heuristic parameters.

Analysis and discussion of the algorithm

In Fig. 2.6, we compare the results obtained using the Adaptive Anderson mixing with the results obtained using the original Anderson mixing with a fixed $a_i \equiv a_0$. We can note that the Adaptive Anderson mixing performs, in most cases, at least as well as the original Anderson algorithm for an optimal a_0 (or, in a few cases, only slightly worse). Additionally, the Adaptive Anderson mixing suffers much less from a bad initial guess of a_0, since the final adapted coefficients a_i approaches the optimal a_0 (see Fig 2.7).

There are two exceptions to this trend. The first exception is the case of ozone, where the convergence is the same for a wide range of a_0 and the optimal $a_0 = 0.9$ obtained in our calculation is evidently a “random result”. The second exception is titanium that will be discussed below.

An interesting situation has been encountered when dealing with the graphene fragment. In this case, the values of adapted coeficients a_i varies during the self-consistent cycle (see Fig. 2.7), which indicates that the Jacobian and thus the optimal value of a_i (not insignificantly) changes.

As a result, final a_i differ from the optimal fixed $a_i \equiv a_0$ coefficient and the adapted version of the algorithm performs better than the original version with any fixed a_0. Thus, this case demonstrates that the adaptation is not only a tool for determining the right a_0 but it can improve the performance of the self-consistency cycle generally.

This observation can be further confirmed by analyzing the $b_{i,i}$ coefficients, employing the heuristic convergence parameter c — see Fig. 2.6: we can see that the largest oscillations of $b_{i,i}$ were substantially reduced.

Limitations of the approach The only exception from the previous observations is the case of titanium atom. This computation used a pseudopotential (see Chapter 4.2) intentionally tuned to a wrong energy. This is the reason why this simple example (a single atom) required more iterations to converge than e.g. hexafluorethane.

Having a pseudopotential tuned to a given energy means that it acts properly only on wave functions with energies not far from that specific energy. It is not necessary to have this energy tuned very well, because the energy range, where the pseudopotential acts properly, is relatively wide and even outside the range the errors are usually small. However, if the reference energy is too far from the correct energy of the true wave function, problems may arise. The wave functions naturally change their energy during the self-consistent cycle: therefore, the effect of such an improper choice of energy range on the wave function changes too, which spoils the convergence of the cycle. The cause lies in the fact that the iterative step made during the Anderson iterative process cannot guess the change of the pseudopotential and thus the currently obtained derivative in Jacobian of the function is not a good estimation of the value of Jacobian in the next iterations.

This effect is especially strong for a large initial a_0, which results in poor estimates of the densities in first iterations, where the energies of wave functions differ substantially from the final values (see Fig. 2.8).

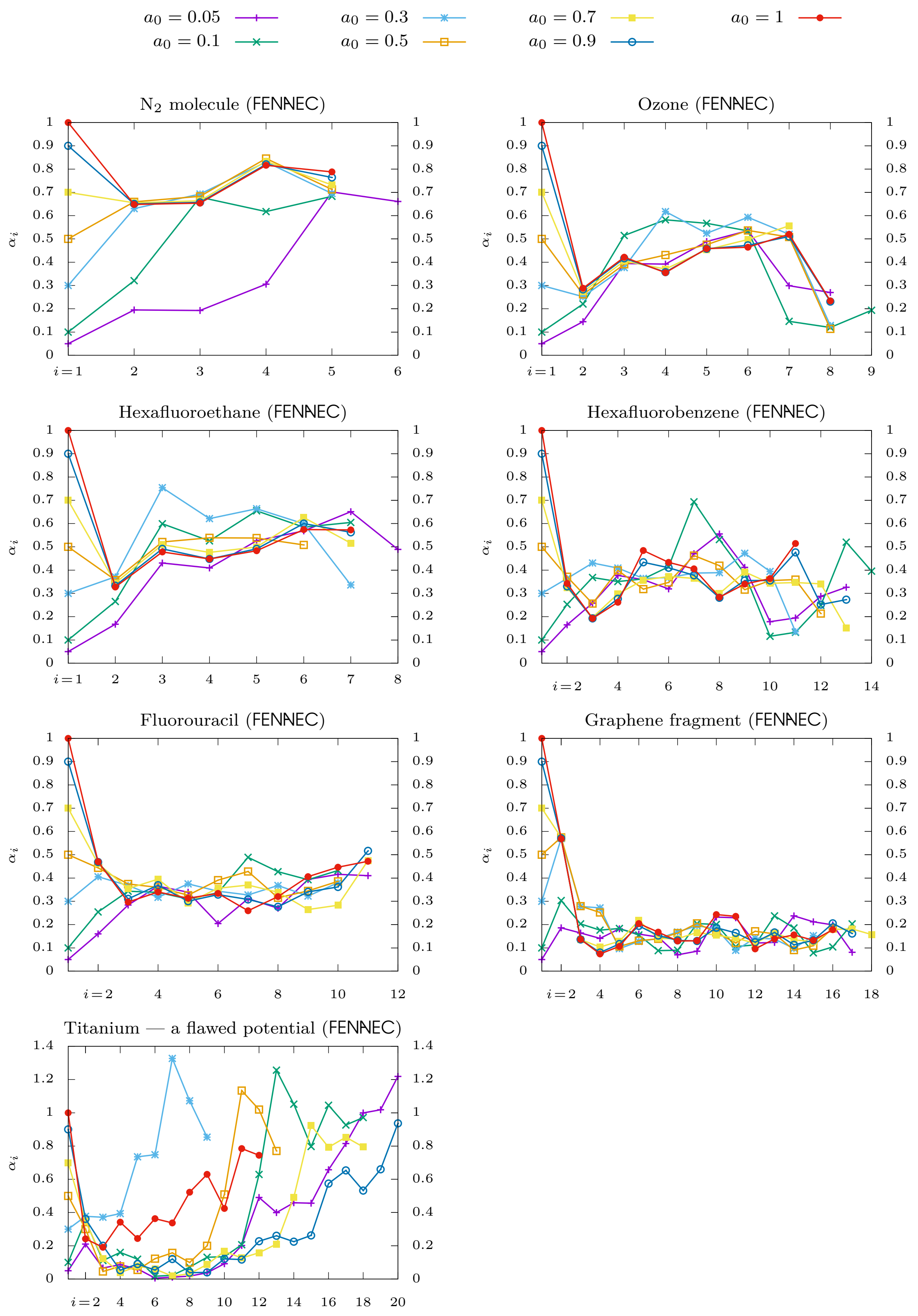


Figure 2.7: Adaptive Anderson Mixing - adaptation of the coefficient a_i during iterations for various systems. The iteration numbers are on the x-axis. Each plotted line denotes the evolution of coefficient a_i during iterations for a given starting a_0.

This example has been added to illustrate the limitations of the proposed adaptive mixing. All fluctuations of the DFT self-consistent cycle are reflected in $b_{i,i}$, not only the ones caused by a wrong estimate of a_i. And if such a secondary source of error is present (be it a wrongly tuned pseudopotential, e.g. a solver instability, or just a too "wild" optimized function), the adaptive mechanism for a_i not only does not help, but it can also spoil the convergence at all. Obviously, the adaptive scheme can just cure a bad guess of a_0 or it can help if there is a need for a smooth change of Jacobian approximation, but it cannot resolve other kinds of problems that affect the mixing performance. Therefore, the adaptive mixing should be used with a proper carefulness.

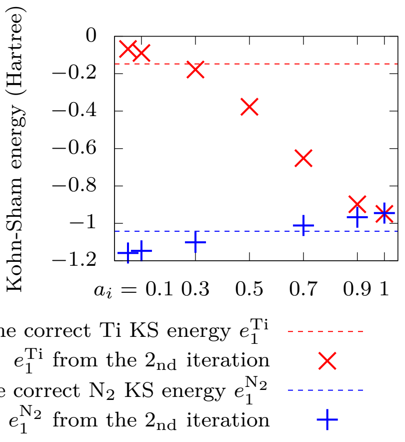


Figure 2.8: Comparing the effects of having a "good" and a "bad" pseudopotential: dependence of the Kohn-Sham energy of the lowest-energy wave function during the second self-consistent iteration on the parameter a_0 for a nitrogen dimer and for a titanium atom with the system pseudopotential intentionally constructed in a flawed way (for a wrong reference energy, see Section 4.2.1 on page 114 for details). Correct final energies are denoted by dashed lines.

On the other hand, even in such case, we can use the analysis of $b_{i,i}$, and the convergence heuristic (2.66), as indicators for the best choice of a_0. In addition, the oscillations of $b_{i,i}$ can be also used to detect instabilities in the DFT self-consistent cycle. Moreover, we can still observe that even in this case the convergence heuristic (2.66) correlates with the speed of convergence. Thus, maybe a better adaptation function could handle even such a case of DFT cycle with secondary source of "stochastic" errors. This topic will be subject to our further research.

Details of the adaptation function

The adaptation algorithm has been determined purely empirically and we — in no way — claim that it is the best possible choice: it can be considered to be the first attempt in our further research. We observed above that the algorithm converges most efficiently if $b_{i,i} \doteq 1$. Further numerical tests showed that the optimal $b_{i,i}$ should be even slightly greater than one, the more the longer the available history is in the given step (to slightly underestimate the Jacobian for the sake of stability). Thus, the optimal $b_{i,i}$ is set to

$$g_i = 1 + dh_i\,, \tag{2.77}$$

where d is the algorithm parameter with the value 0.02 and h_i is the available history length in the i_{th} step (recall Definition 2.8). Then the adaptation of a_i is determined via the following expression:

$$a_i = f(b_{i,i}/g_i)a_{i-1}\,. \tag{2.78}$$

The function f should be designed so that it allows both to reach the correct a_i quickly and to prevents large fluctuations of a_i. In our approach, f is designed as follows:

$$\bar{f}(x) = x^{(1/p)}\,, \tag{2.79}$$

$$f(x) = 0.1 \qquad \text{for} \qquad \bar{f}(x) \leq 0.1\,, \tag{2.80}$$

$$= \bar{f}(x) \qquad \text{for} \qquad 0.1 < \bar{f}(x) \leq 2\,, \tag{2.81}$$

$$= 2 + log(\bar{f}(x)/2) \qquad \text{for} \qquad 2 < \bar{f}(x)\,. \tag{2.82}$$

The coefficient p is equal to either one, two or three, depending on whether two, one or none of the following criteria are met: (i) the adaptive change of a_i is in the same direction as in the previous step, and (ii) the adaptive change is in the same direction during all steps.

Algorithm 2.2: Adaptive Anderson mixing

```
1: $\rho_1^{\text{in}}$ = a guess of the charge density
2: **for** $i = 1, 2, \ldots$ **do**
3:     $\rho_i^{\text{out}} = F(\rho_i^{\text{in}})$
4:     $\tau_i = \rho_i^{\text{out}} - \rho_i^{\text{in}}$                              ▷ New residual
5:     **if** $\|\tau_i\| <=$ threshold **then return**
6:     If some residuals are collinear to the newer ones, discard them from history
7:     Solve coefficients $b_{i,j}$ from (2.62)                         ▷ [85]
8:     $a_i = a_{i-1} * f(\text{len}(\{b_i\}), b_{i/i})$                ▷ Adaptation of $a_i$
9:     $\rho_{i+1}^{\text{in}} = \sum_{j=1}^{i} b_{i,j}\rho_i^{\text{in}} + a_i b_{i,j}\tau_i$      ▷ Set the new input density
```

2.5.5 Conclusion

The Adaptive Anderson mixing — as it is summarized in Algorithm 2.2 — differs from the standard Anderson mixing implementation just in the adaptive a_i coefficient. Therefore, it can be easily implemented in the existing softwares. It seems that, especially for complex systems, using the adaptive a_i can lead to a better performance than relying on constant $a_i \equiv a_0$, even in the case of optimal choice of a_0 (see Graphene fragment results in Fig. 2.2). Besides that, the adaptive scheme exhibits much smaller dependence on the initial guess of a_0. This mixing scheme has been implemented into two DFT codes, FENNEC and SPRKKR.

There are still open questions regarding the proposed mixing. The adaptation function has been designed empirically to try the idea of the adaptive mixing. Further effort in developing the scheme could probably bring even better results, especially the robustness (see the titanium case in Fig. 2.2). Also the principles of the mixing themselves — especially the criteria assessing the suitability of a_i and $b_{i,i}$ coefficients — are worth a more rigorous investigation. Such an effort could shed more light on the whole topic and allow designing a better adaptation scheme for a_i. Our further research will be aimed at these topics.

Moreover, several mixing-related techniques (some of those described above) are worth to be implemented in FENNEC, since they could lead to a performance improvement. The first of them is the mixing of the potential instead of the density, which can be easily implemented. The second technique is the use of regularization in the Adaptive Anderson algorithm [97, 98]. This is a more sophisticated approach to deal with nearly collinear residuals, instead of discarding them (see lines 5 of the algorithm).

Another interesting technique is Broyden's first method with a longer history, which has not been considered in the comparison in this Section (it has not been implemented neither in FENNEC nor in SciPy yet). According to Fang's numerical experiments [84], it could perform a bit better than the Anderson method for some problems. And finally, the position-dependent mixing seems to be a promising idea especially combined with the adaptation of a_i. The coefficients $b_{i,j}$ can be determined for a particular domain in the same way as "global $b_{i,j}$" and the a_i values can be adapted for different subdomains by a corresponding position-specific $b_{j,j}$.

3. Finite element method and isogeometric analysis

Contents

The Kohn-Sham equations (2.44) cannot be solved analytically in practice. Thus, this infinite-dimensional problem must be approximated by some finite-dimensional one — this process is called a *discretization* — and solved numerically.

The discretization replaces the original complex problem with several continuous unknowns with an approximate problem involving a potentially huge number of discrete unknowns, suitable for a computer.

From many possible ways of discretization, the *finite element method* (abbreviated as *FEM*) is used in the software package FENNEC, especially for the following features:

generality The FEM offers a general basis and related excellent convergence control[1]. Moreover, the basis assumes no prior knowledge of the problem solution.

reliability The FEM is used in many fields of engineering and science. It has both a strong theoretical background and many successful practical implementations.

scalability By choosing a proper mesh and a level of its adaptive refinement, the accuracy of the method and computational demands can be managed. The assembling/solution of the resulting matrix problems can also be parallelized.

[1]This means, that the basis can be built such that the error coming from the discretization becomes arbitrarily low.

availability Reliable FEM software packages and solvers are available (even as Open Source), that can be used for implementing the new method.

The *isogeometric analysis*[99] (abreviated as *IGA*) is incorporated into the new method as a alternative to FEM in the case that continuous derivatives of the solution (of the discretized problem) are needed [100, 101].

Solving a differential equation using FEM or IGA requires three steps. First, the equation must be transformed to its weak form. Second, the weak form of the equation must be discretized: the solution is sought in a finite-dimensional (finite element) space as a linear combination of basis functions. The discretization results (in our case) in linear algebra problems: the Poisson equation (5.47) leads to a system of linear equations and the Kohn-Sham equations (2.44) make up a generalized eigenvalue problem — so the third step is clear: the problems have to be solved.

This chapter describes the finite element method and isogeometric analysis concerning its use in the package FENNEC and the analysis of the best parameters for employing FEM/IGA for electronic structure calculations using DFT.

The first section describes topics common to both FEM and IGA: the weak or integral forms of a problem and the concept of discretization. Discretization is demonstrated on the Poisson equation (2.47), which is solved in iterations of the DFT self-consistent cycle.

The second section briefly reviews the FEM from the theoretical point of view, with a practical implementation on mind — details not relevant to the FENNEC implementation are omitted for the sake of conciseness. Detailed information on the theory of the finite element method and its applications can been found in many textbooks, e.g. [102, 103, 104, 19, 105, 18].

The third section is dedicated to the basics of isogeometric analysis, especially to the differences between FEM and IGA. The source of the summary is primarily the work of my collaborator and FEM/IGA specialist Robert Cimrman [100, 106], supplemented by other sources (e.g. [107, 108, 99]).

In the last section, some parameters of the finite element method (e.g. the basis type and order, shape of elements) are described and their influence on the convergence of electronic structure calculations is analyzed. The results in this section are original, they have been published in [106] and extended for the purpose of the thesis.

3.1 Weak problem and discretization

As have been said in the previous section, to solve an equation using FEM, the equation must be transformed to its weak form and discretized, this will be covered in this section. First, we state definitions that allow us to formulate the problem and its transformation into a weak form. Then we will demonstrate the process on the Poisson equation (5.47), whose solution is a part of the DFT self-consistent cycle. Finally, we discretize the problem to be solved either by FEM or IGA.

3.1.1 Mathematical formulation of the problem

To build a formal definition of FEM, we first have to define the problem and its components.

Definition 3.1 (Volume (solution domain)). *The* volume *or the* solution domain *of a partial differential equation (abbreviated as PDE) in* $\mathbb{R}^n$ *is a subset of space* $\mathbb{R}^n$*, where we seek the solution of the PDE.*

Note that the domain is not necessarily nor usually a vector subspace, it is just a "part of space" on where we need to describe the solution.

Definition 3.2 (Solution domain boundary (surface)). *Boundary* $\partial\Omega$ *of a volume* Ω *is the subset of* Ω*, whose members are not members of any closed subset of* Ω*.*

The solutions of PDE lie in the space of differentiable functions in Ω. However, since FEM solves the problem in its weak form, a space of functions with *weak (partial) derivatives* is sufficient.

The weak derivative is a generalization of the concept of derivatives for a larger class (even not necessarily continuous) of functions, based on a generalization of the integration by parts to more dimensions:

$$\int (fg)' = \int fg' + \int f'g \ . \tag{3.1}$$

If the function f is zero on the boundary of an interval (a, b), the integrated equation (3.1) simplifies to

$$\int_a^b fg' = -\int_a^b f'g \ , \tag{3.2}$$

which leads to the following idea of the *weak derivative*:

Definition 3.3 (Weak partial derivative). *Let* f, g *be functions on a closed space* $\mathcal{L}$ *of the integrable functions in* Ω *with the boundary* $\partial\Omega$*. Let* $\mathcal{C}_0^1$ *be the subspace of* $\mathcal{L}$ *of all differentiable functions, that are zero on* $\partial\Omega$*. If it holds for an* $\xi \in \mathcal{L}$ *and* $i \in \{1 \dots n\}$*, that*

$$\forall \phi \in \mathcal{C}_0^1 : \int_\Omega f \frac{\partial \phi}{\partial \vec{x}_i} = -\int_\Omega g\phi \ , \tag{3.3}$$

then g *is the* weak partial derivative *of the function* f *by* $\vec{x}_i$*.*

If a function f is differentiable, its (partial) derivative and weak (partial) derivative are the same function. [19]

Using weak derivatives, we can define the *Sobolev space* as the space, where the solution is sought:

Definition 3.4 (Sobolev space). Sobolev space $\mathcal{W}^{1,2}(\Omega)$ *is a space of functions with (Lebesgue-) square integrable values and weak derivatives in* Ω*.*

This space is a Hilbert space of functions whose values and weak partial derivatives are square-integrable in the Lebesgue sense, i.e., belong to $L^2(\Omega)$, [109].

The indexes ^{1,2} denotes that the functions in the Sobolev space have weak derivatives of the first order and that the space uses the L^2 norm. Since we will use only such Sobolev spaces, these indexes will be omitted in the following development.

Dirichlet problem

All equations solved by FEM in FENNEC are of the Dirichlet type, therefore, we will focus only on the *Dirichlet problems* in the following text. For discussion of Neumann problems with so-called *natural boundary condition* (where the derivatives of the solution are enforced on a boundary part) or mixed problems (both condition types employed) and further types of problems, please see any above-mentioned FEM textbook or the following brief tutorial [110].

The following (more or less informal) definitions will allow us to define the weak form of the PDEs in the next section.

Definition 3.5 (Dirichlet boundary condition)**.** *Given a Sobolev space $\mathcal{W}(\Omega)$, a* Dirichlet boundary condition *of a PDE for an unknown function $\psi \in \mathcal{W}(\Omega)$ is a condition (restricting the solutions of the PDE) of the form*

$$\forall \vec{x} \in \partial\Omega : \psi(\vec{x}) = f(\vec{x}), \tag{3.4}$$

determined by a function $f \in \mathcal{W}(\Omega)$.

Definition 3.6 (Dirichlet problem)**.** *The* Dirichlet problem *is a (partial) differential equation with a solution in a Sobolev space $\mathcal{W}(\Omega)$, equipped with the Dirichlet boundary conditions.*

A Dirichlet problem can be transformed (see below) to a homogeneous Dirichlet problem.

Definition 3.7 (Homogeneous Dirichlet problem)**.** *The* Homogeneous Dirichlet problem *is a Dirichlet problem with zero boundary conditions.*

The solutions of such a problem lie in the *zero boundary space*:

Definition 3.8 (Zero boundary space)**.** *The* zero boundary space $\mathcal{W}_0(\Omega)$ *is the maximal subspace of $\mathcal{W}(\Omega)$, whose members satisfy the zero boundary condition on the boundary $\partial\Omega$ of Ω:*

$$\mathcal{W}_0(\Omega) = \{\psi \in \mathcal{W}(\Omega) : \forall \vec{x} \in \partial\Omega : \psi(\vec{x}) = 0\}\,. \tag{3.5}$$

It is easy to prove:

Theorem 3.1 (Zero boundary space)**.** *The zero boundary space $\mathcal{W}_0(\Omega)$ is a vector space.*

Proof. $\mathcal{W}(\Omega)$ itself is a vector space and any linear combination of the functions zero on the boundary is still zero on the boundary; thus $\mathcal{W}_0(\Omega)$ is closed both to multiplication and addition. □

The non-homogeneous Dirichlet problem will be solved with the help of the transformation to the homogeneous Dirichlet problem, for which we will need an *affine subspace.*

Definition 3.9 (Affine subspace)**.** *Given a vector space*[2] $\mathcal{W}(\Omega)$*, its vector subspace* $\mathcal{W}_0(\Omega)$ *and a* translation vector $f \in \mathcal{W}(\Omega) \setminus \mathcal{W}_0(\Omega)$*. Then* affine subspace $\mathcal{W}_0^f(\Omega) = \mathcal{W}_0(\Omega) + f$ *is a space of vectors*

$$\mathcal{W}_0^f(\Omega) = \{\psi : \psi = \psi_0 + f,\ \psi_0 \in \mathcal{W}_0(\Omega)\} \qquad (3.6)$$

A visualisation of an afine subspace can be seen in Fig. 3.1.

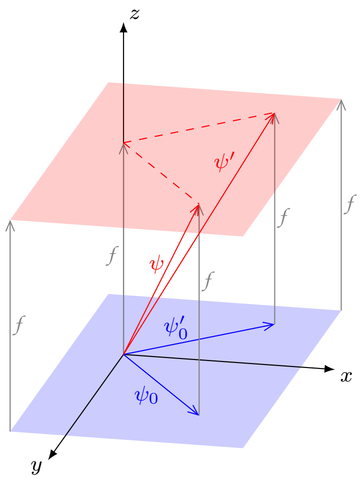


Figure 3.1: The red *affine subspace* is translation of the blue vector subspace.

Weak problem formulation

We will formulate the *weak form* of the homogeneous Dirichlet problem, that is required for solving our equations using FEM. For the sake of clarity, we will define the weak form for a particular problem type. Generalizations for other problem types (e.g. an eigenvalue problem) are obvious.

Definition 3.10 (Weak solution)**.** *Given a homogeneous Dirichlet problem (in its strong form) of the following type*

$$V(\psi) = \sigma \qquad \psi \in \mathcal{W}_0(\Omega), \quad \sigma \in \mathcal{W}(\Omega), \quad V : \mathcal{W}(\Omega) \to \mathcal{W}(\Omega)\,, \qquad (3.7)$$

the weak solution *of the problem is a function* $\psi \in \mathcal{W}(\Omega)$ *that for each* test function $\phi \in \mathcal{W}_0(\Omega)$ *satisfies the condition*

$$(V(\psi)\ \cdot\ \phi) = (\sigma\ \cdot\ \phi)\,, \qquad (3.8)$$

or in integral notation

$$\int_\Omega V(\psi)\phi^+ = \int_\Omega \sigma\phi\phi^+\,. \qquad (3.9)$$

The solution of the original problem is always the solution of the weak one (see e.g. [19]).[3]

[2]We will use for convenience notation suitable for functional spaces, however, this definition holds for all vector spaces.

[3]The opposite implication does not hold: there are many solutions that differ only on a zero measure set and so they are in the "integral" sense equal to each other, but such solutions do not have to solve the original "strong" problem. However, since weak solutions are in practice always obtained using some reasonable basis, the existence of such "weird" solutions does not matter. Moreover, as Strakoš points (see the excellent book [6]), not the strong form of the problem, but the weak one — viewed as a minimization of an energy functional — can be considered in many cases as the primary form of the problem.

Theorem 3.2 (Solutions of Dirichlet problems)**.** *Given the weak form of a Dirichlet problem with the boundary conditions f on its boundary $\partial\Omega$, its solution lies in an affine subspace $\mathcal{W}_0(\Omega) + \psi_f$, where ψ_f is an arbitrary function from $\mathcal{W}(\Omega)$ that satisfies the boundary condition.*

Proof. Let ψ be a solution of the problem. Then by definition

$$\forall \vec{x} \in \partial\Omega : \psi(\vec{x}) = \psi_f(\vec{x})\,. \tag{3.10}$$

Therefore

$$\psi - \psi_f \in \mathcal{W}_0(\Omega) \quad \Longrightarrow \quad \psi \in \mathcal{W}_0(\Omega) + \psi_f\,. \tag{3.11}$$

$\square$

The theorem allows us to take an arbitrary translation vector V_{H}^f that satisfies the boundary condition and then seek the solution in the following form:

$$V_{\mathrm{H}} = V_{\mathrm{H}}^0 + V_{\mathrm{H}}^f \qquad V_{\mathrm{H}}^0 \in \mathcal{W}_0(\Omega)\,. \tag{3.12}$$

This theorem is applied in the next subsection.

3.1.2 Weak form of the Poisson equation

We can now apply the above transformation to the Poisson equation (Eq. 2.47 on page 31), that needs to be solved during the DFT iterations. For the sake of self-containment of this chapter, we first restate the equation, that has to be solved for an unknown V_{H}:

$$-\frac{1}{4\pi}\nabla^2 V_{\mathrm{H}} = \rho\,. \tag{3.13}$$

To transform the equation (3.13) to its weak form (see Eq. 3.8), let us multiply the equation by a complex conjugate of the test function ϕ and integrate over the whole volume:

$$\int_\Omega -\frac{1}{4\pi}\nabla^2 V_{\mathrm{H}}\phi^+ = \int_\Omega \rho\phi^+. \tag{3.14}$$

The Laplace operator contains the second derivatives, which forces requirements on the continuity of derivatives of the utilized FE basis.[4] Since the FE bases with continuous derivatives are not constructed easily (see Section 3.2.3), the better approach is to use the *first Green identity*

$$-(\nabla^2\psi)\phi = (\nabla\psi)\cdot(\nabla\phi) - \nabla\cdot((\nabla\psi)\phi)\,, \tag{3.15}$$

to remove the second derivatives:

$$\int_\Omega -\frac{1}{4\pi}\left(\nabla V_{\mathrm{H}}\nabla\phi^+ - \nabla\cdot(\nabla V_{\mathrm{H}}\phi^+)\right) = \int_\Omega \rho\phi^+ \tag{3.16}$$

[4]It is not completely true, as the commonly used FEM bases have discontinuous derivatives only on a set of zero measure, however, still, the discontinuity brings problems that are better to avoid.

and then to apply *Gauss's theorem*[5], that transforms the integral of a divergence to the flux through the boundary

$$\int_\Omega \nabla \cdot \Psi = \int_{\partial\Omega} \Psi \cdot \vec{n} \,. \tag{3.17}$$

This common FEM approach results in

$$\int_\Omega -\frac{1}{4\pi}\nabla V_\mathrm{H} \nabla \phi^+ - \int_{\partial\Omega} \vec{n} \cdot (\nabla V_\mathrm{H} \phi^+) = \int_\Omega \rho\phi^+ \,. \tag{3.18}$$

Since the test function is zero on the boundary by definition ($\phi \in \mathcal{W}_0(\Omega)$), the surface integral vanishes and we obtain the weak form of the Poisson equation:

$$\int_\Omega -\frac{1}{4\pi}\nabla V_\mathrm{H} \nabla \phi^+ = \int_\Omega \rho\phi^+ \,. \tag{3.19}$$

To treat the non-homogeneous Dirichlet boundary conditions, we substitute (3.12) into (3.19):

$$\int_\Omega -\frac{1}{4\pi}\nabla V_\mathrm{H}^0 \nabla \phi^+ = \int_\Omega \rho\phi^+ - \int_\Omega -\frac{1}{4\pi}\nabla V_\mathrm{H}^f \nabla \phi^+ \,. \tag{3.20}$$

While the known function V_H^f could be taken arbitrarily, the most usual choice for FEM will be shown in the next section. The solution V_H of the original problem (3.19) is obtained as $V_\mathrm{H} = V_\mathrm{H}^0 + V_\mathrm{H}^f$. Notice that the right hand side of (3.20) varies according to the selected translation vector V_H^f.

Thus, we obtain the weak formulation of the Dirichlet problem. The next step is the problem discretization using a FE basis.

3.1.3 Discretization

A discretization of a problem in its weak form consists of approximating an infinite dimensional space $\mathcal{W}(\Omega)$ by a finite dimensional space $\mathcal{W}_h(\Omega)$. The letter h stands for the *discretization parameter*, see e.g. [19] for the theory of convergence of the finite element method.

This approximation is done using a projection $P_{\mathcal{W}_h^\Omega} : \mathcal{W}(\Omega) \to \mathcal{W}_h(\Omega)$, a few details of the choice of the projection will be given in the section 3.2.3. The projection $P_{\mathcal{W}_h^\Omega}$ can be spatially restricted to the corresponding boundary space: $P_{\mathcal{W}_{h0}^\Omega} : \mathcal{W}_0(\Omega) \to \mathcal{W}_{h0}(\Omega)$.

The construction of the space $\mathcal{W}_h(\Omega)$ will be covered in the next sections, now we will assume that we have a suitable space, with its basis $\{\varphi_i : i \in \mathbb{N}, i \leq n\}$ of dimension n. We will seek the solution in its subspace, which is equivalent to $\mathcal{W}_0(\Omega)$:

Definition 3.11 (Zero boundary finite element space)**.** *Let $\mathcal{W}_h(\Omega)$ be a finite element space with n_0 basis functions that are zero on the boundary (inner functions), with the basis ordered in such a way that the n_0 inner functions are in front of the other (boundary) basis functions. Then the* zero boundary finite element space *$\mathcal{W}_{h0}(\Omega)$ is the finite element space spanned by the n_0 inner basis functions.*

$$\mathcal{W}_{h0}(\Omega) = \operatorname{span}\{\varphi_i : i \in \{1, \dots, n_0\}\} = \operatorname{span}\{\varphi_i : \varphi_i \in \mathcal{W}_0(\Omega)\} \,. \tag{3.21}$$

[5]Called also the *Divergence theorem.*

If we have a boundary condition determined by a function f, we can choose $V^f_{\mathrm{H},h}$ from $\mathcal{W}_h(\Omega)$ and approximate the solution and the test function in $\mathcal{W}_{h0}(\Omega)$:

$$V^0_{\mathrm{H}} \doteq \sum_i v_i \varphi_i \;, \qquad i \in \{1, \ldots, n_0\} \;, \tag{3.22}$$

$$\phi \doteq \sum_j t_i \varphi_j \;, \qquad j \in \{1, \ldots, n_0\} \;, \tag{3.23}$$

$$V^f_{\mathrm{H}} \doteq V^f_{\mathrm{H},h} = \sum_k b_k \varphi_k, \text{ such that } P_{\mathcal{W}^{\Omega}_{h0}}(f) = V^f_{\mathrm{H},h} \;, \quad k \in \{n_0+1, \ldots, n\} \;, \tag{3.24}$$

$$V_{\mathrm{H}} = V^0_{\mathrm{H}} + V^f_{\mathrm{H}} \;. \tag{3.25}$$

The translation vector V^f_H has been constructed in the simplest way: as a function satisfying the boundary conditions and zero in the “interior” of the discretization domain.

Substituting the approximations (3.22)–(3.25) into the weak Poisson equation (3.20) yields

$$-\frac{1}{4\pi}\int_\Omega \sum_i v_i \nabla\varphi_i \sum_j t_j \nabla\varphi_j^+ = \int_\Omega \rho \sum_j t_j \varphi_j^+ + \frac{1}{4\pi}\int_\Omega \sum_k b_k \nabla\varphi_k \sum_j t_j \nabla\varphi_j^+ \;, \tag{3.26}$$

which can be expressed in the scalar product notation:

$$-\frac{1}{4\pi}\left(\sum_i \mathbf{v},\mathbf{i}\nabla\varphi_i \,\cdot\, \sum_j \mathbf{t},\mathbf{j}\nabla\varphi_j\right) = \left(\rho \,\cdot\, \sum_j t_j\varphi_j\right) + \left(\sum_k b_k\nabla\varphi_k \,\cdot\, \sum_j t_j \nabla\varphi_j\right) . \tag{3.27}$$

From the linearity of the gradient operator it is sufficient to satisfy (3.26) only for the basis of $\mathcal{W}_{h0}(\Omega)$, which results in n_0 equations

$$\forall j \in \{1, \ldots, n_0\} : \frac{-1}{4\pi}\left(\sum_i v_i \nabla\varphi_i \,\cdot\, \nabla\varphi_j\right) = (\rho \,\cdot\, \varphi_j) + \left(\sum_k b_k \nabla\varphi_k \,\cdot\, \nabla\varphi_j\right) , \tag{3.28}$$

with v_i as the unknown coefficients. Therefore, a system of linear equations is obtained:

$$A\mathbf{v} = \mathbf{b} \,. \tag{3.29}$$

This system can be easily solved, as compared to other parts of the DFT iterations, see section 6.3.3 for details. To obtain the solution of the original Poisson equation (3.13), the boundary term (3.24) must be added back to the obtained solution of the linear system (3.29).

As we will see in the next chapter, the practical solvability of the system and an acceptable discretization error are not automatic properties of the FE method. However, the use of pseudopotentials (see chapter 4.2) ensures, that the resulting charge density (that determines the right-hand side of the Poisson equation 3.13) does not have too steep gradients and does not fluctuate — therefore it is rather well describable by common finite element bases. Moreover, there is no problematic term in the equation and therefore no additional work for solving the equation is required. This topic is covered by appropriate parts of the next chapter, whose analysis of the discretization of the Kohn-Sham equations can be applied to the case of the Poisson equation without loss of generality as well.

The previous statement has one exception: the resulting matrix A from the discretized Poisson equation (3.29) can be ill-conditioned, especially if a refined mesh with hanging nodes is used (see definition 3.19 on page 63), which can result in poor convergence of numerical solvers applied to the problem. However, this issue can be rather easily cured by employing a preconditioner. According to our experiences, a general-purpose preconditioner is sufficient in practice (see chapter 6.3.3).

3.2 Finite element method

So far the description has been common both for IGA and FEM. These methods differ in how the discretization space $\mathcal{W}_h(\Omega)$ is constructed. In this section the construction of a FEM basis will be introduced, the construction of IGA bases is described in the next section 3.3.

Please note that although the following definitions are exact, more general definitions exist that allow even greater freedom in constructing finite element spaces (e.g. the possibility of “curved” elements [103], different coordinate systems, etc.). Therefore the following definitions can be considered as a formal but simplified approach that suits our needs to have some background for solving the Kohn-Sham equations using FEM. To gain a better insight into the problem, the following references are recommended [102, 103, 104, 19, 18].

The idea of FEM is as follows: first, the described object/solution domain is divided into small parts, “cells” called *elements*: that can be treated (nearly) independently. This process is called *meshing.* Then local basis functions — *shape functions* – are defined on the elements, using an isomorphism with a fictional ideal *reference element.* This isomorphism allows treating the differences among elements only by transformation coefficients. And finally, a global discretization basis is constructed using the shape functions.

3.2.1 Meshing and meshes in FENNEC

There are two major approaches for a geometry description of a computational domain. The Lagrange description, in which the mesh is connected with the object(s) of interest themselves, e.g. a sample of a deformed material or a wing of an airplane [111]. On the contrary, in the Euler description the mesh is connected to the (stationary) space and properties of the computed objects are expressed as functions of the space (and time). While the Lagrange description might be used more often in disciplines such as engineering, the Euler description meets the needs of the quantum mechanics.

For describing a geometry, we will start with several formal definitions, that aim to define the *mesh*: the description of the volume (space) for computations and of the division of the volume to elements. Generally, a mesh is not required to cover the whole volume (e.g. it can only approximate curved boundaries of the volume).

Definition 3.12 (Discretized volume)**.** *The discretized volume Ω_h is a subset of $\mathbb{R}^n$, that approximates the original volume Ω of a problem.*

Elements of a mesh are constructed as convex hulls.

Definition 3.13 (Convex hull)**.** *Let Ω_h be a volume and $\mathbf{T} = \{\vec{v}_i \in \Omega_h\}$ is a set of vectors from the volume. The convex hull of $\mathbf{T}$ is the subspace $\mathrm{conv}(\mathbf{T}) \subset \Omega_h$ that holds*

1. $\mathbf{T} \subset \mathrm{conv}(\mathbf{T})$
2. $\forall \vec{x}, \vec{y} \in \mathrm{conv}(\mathbf{T}), \forall i \in \langle 0, 1\rangle : \vec{x} + i(\vec{y} - \vec{x}) \in \mathrm{conv}(\mathbf{T})$.
3. $\forall \mathbf{T}' \subset \Omega_h$ *that satisfy 1. and 2.* $: \mathbf{T} \subset \mathbf{T}'$.

Definition 3.14 (Vertex)**.** *The vectors $\vec{v}_i$ that determine a convex hull* $\mathrm{conv}\,(\vec{v}_i)$ *are called* vertices *of the convex hull.*

Definition 3.15 (Regular polyhedron)**.** *A set* $\mathbb{T} = \mathrm{conv}(\mathbf{T})$ *with* vertices $\vec{v}_i \in \mathbf{T}$ *is the* regular polyhedron *if*

1. *it is not degenerate:* $\dim \mathbb{T} = \dim \Omega_h$.
2. *it has no unnecessary vertex:* $\forall \vec{v}_i \in \mathbf{T} : \mathrm{conv}(\mathbf{T}) \neq \mathrm{conv}(\mathbf{T} - \vec{v}_i)$.

Definition 3.16 (Mesh)**.** *Given $\Omega_h \subset \mathbb{R}^n$, a set of regular polyhedrons $\{\mathbb{T}_k\}$,*

1. *that covers Ω_h: $\bigcup_{i=1}^{k} \mathbb{T}_k = \Omega_h$, and*
2. *whose elements are nonoverlaping: $\forall i, j \in \{1 \dots k\} : \mathbb{T}_i \bigcap \mathbb{T}_j$ has meassure of zero ,*

is called the mesh *of Ω_h. The polyhedrons $\mathbb{T}_k$ are called* elements *(of the mesh).*

Other useful terms for describing a mesh are *edge* and *face*. They are nearly self-explanatory, for the sake of completeness a definition suitable for the 3D case is given below.

Definition 3.17 (Face)**.** *The* Face *of a mesh element is a convex hull of a subset of vertices of the element, for which it holds that:*

- ▷ *It has dimension two.*
- ▷ *Its intersection with the interior (maximal open subset) of the element is zero.*
- ▷ *Adding any vertex of the element to the set violates the conditions above.*

Definition 3.18 (Edge)**.** *The* edge *of an element is a convex hull of two vertices of the element, whose intersection with the interior of any face is zero.*

In some literature, meshes are not allowed to have *hanging nodes* — nodes in the middle of a face or an edge of another element:

Definition 3.19 (Hanging node)**.** *Let* $\{\mathbb{T}_n\}$ *be a mesh and* $\vec{v}_{i,j}$ *be the* j_{th} *vertex of the* i_{th} *mesh element. Then* $\vec{v}_{i,j}$ *is called a* hanging node*, if*

$$\exists \mathbb{T}_k \neq \mathbb{T}_i : \vec{v}_{i,j} \in \mathbb{T}_k \quad \& \quad \forall l : \vec{v}_{k,l} \neq \vec{v}_{i,j} \, . \tag{3.30}$$

However, the hanging nodes are often allowed in practically used *non-uniformly refined meshes.* Therefore we introduce also the term *conforming mesh.*

Definition 3.20 (Conforming mesh)**.** *The* conforming mesh *is a mesh without hanging nodes.*

Since a mesh is necessarily spatially finite, whereas electron wave functions are not, there is an approximation hidden in the process of the selection of the volume, and in the subsequent meshing of the volume.[6]

However, wave functions are negligible in regions far from atomic cores and therefore the approximation does not spoil the results, though it is needed to have this fact on mind and to design the mesh sufficiently large in order to cover the proper distance from atomic cores: how far the mesh should reach is dependent on the material and the desired precision.

Practical aspects of meshing

So far we talked about the theoretical properties of a mesh. However, from the practical point of view, the question: “how to build a mesh?” is much less important than the question: “how to build a suitable mesh” for a particular task. From a certain point of view, meshing is the hardest part of the application of finite elements. Though the domains needed to be described for electronic structure calculations are not as complicated, as the ones resulting from the Lagrange approach in other FEM applications, the computational demands of the method require that proper attention should be given to the mesh construction.

A good mesh should...

1. ...have as few elements as possible for computational efficiency,
2. ...have sufficiently small elements to describe the problem with a desired precision,
3. ...have shapes of elements that yield matrices with good numerical properties in the finite precision arithmetic,
4. ...have shapes of elements that yield low error bounds on the discretization error.

The first and the second property are pretty clear: as we will see, the more elements the mesh has, the more basis functions the resulting FEM basis will have and so the larger dimension the resulting matrix will have. However, we need a sufficient number of basis functions to approximate the solution well.

[6]Unless a material periodic in all directions is studied, but in such a case it is better to use existing methods for periodic systems.

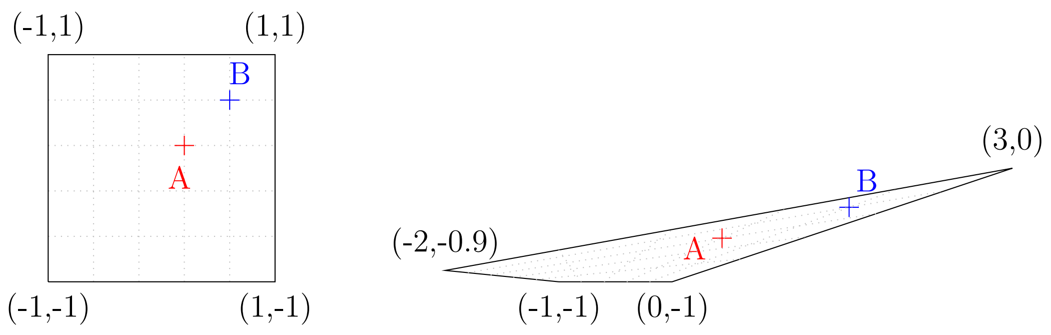


Figure 3.2: Example of a “suitable” and a “ill-shaped” mesh element. The points A and B have the same relative (see chapter 3.2.2) coordinates.

During the meshing, local properties of the problem should be taken into account: in regions, where the solution has high gradients and changes rapidly, more basis functions are needed than in regions where the solution does not fluctuate. Therefore there is no “one correct element size”: the most suitable element size differs according to both the wanted precision and to the behavior of the solved problem in a given place. Hence, in practice, the meshes are created with variable element size according to the expected behavior of the solution — or mesh *adaptivity* (called *h-adaptivity* to distinguish it from *p-adaptivity* see section 3.4.1 on page 95) is employed when a coarse mesh is taken at first, which is gradually refined during computation according to the intermediate results.

The third property determines the conditioning of the resulting matrices. The conditioning affects numerical errors arising during computation with the matrices and real performance of used solvers. Both the shape and the size of the elements affect the spectrum of the matrices (see e.g. [112]). To fulfill the first two properties, both small and big elements are usually required and therefore the conditioning of the resulting matrices is spoiled.

The fourth property of a suitable mesh is very similar to the third, but while the third spoke about numerical errors (errors that came from the finite precision arithmetic), this one takes care of the discretization error (the difference between the analytical and the discretized solution). Although the sizes and shapes of the finite elements affect both errors, neither of them is a function of the other [113].

A rough informal insight into the “badness” of an element can be gained from Fig. 3.2. Although the second element in the picture is three times smaller in area than the first one, the same perturbation of reference coordinates (see chapter 3.2.2) yields a greater error on this element than on the first one.

Though this observation does not cover the whole complexity of the problem[7], in our application the highly irregular shapes of finite elements (sometimes called *ill-shaped elements*) should be avoided, as this brings an artificial anisotropy to the problem. Especially it can be shown, that small or large angles at vertices usually imply high error bounds, same as hexahedra with a large ratio of the longest and shortest edge do. A brief introduction to the error analysis can be found in [115].

[7]It holds for isotropic material properties — for highly anisotropic environments, a skewed element is be preferable, cf. [112, 114].

The mentioned properties of a suitable mesh are somewhat contradictory. The first two conditions could be satisfied by a mesh with varying element sizes. However, it is not possible for hexahedral meshes either without ill-shaped elements (that do not meet the third condition) or without using the *hanging nodes*, that requires special care during FEM matrices assembling (e.g. [116]). Moreover, even if hanging nodes (see Def. 3.19) are employed, which in many cases solves the problem with ill-shaped elements, combining large and small elements leads naturally to ill-conditioned matrices since scalar products of elements shape functions are of different magnitudes. In such cases the conditioning of resulting matrices should be cured by preconditioning:

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b} \tag{3.31}$$

that transforms the spectrum of a matrix so that the condition number of the matrix (the ratio of the largest and the smallest magnitude eigenvalue) is limited. Therefore, the matrix P — the *preconditioner* – should be an easily invertible approximation of the matrix A. A practical implementation of a suitable preconditioner (both sufficiently exact and cheap to evaluate) is not a simple task and it is far beyond the scope of this thesis. For further discussion of the problem see e.g. [117, 118, 119].

Meshes used in FENNEC

Two basic mesh types are available in the FENNEC package: the uniform mesh — whose elements have all the same shape and size — and the non-uniform mesh. The uniform meshes are suitable for convergence analysis and other validation purposes, since they offer the best element shapes and excellent conditioning, and bring no prior knowledge (or anisotropy) into the problem. However, they are unsuitable for practical computations, since the resulting discretized problems are too large to be efficiently solved in the cases of larger atomic structures.

Therefore, non-uniform meshes with elements of different sizes have to be utilized. The element sizes should reflect the behavior of the wave functions: wave functions change rapidly in the proximity of the atomic cores, while they are smoothly decreasing in more distant regions. Therefore, nonuniform meshes should (and they are in FENNEC) created in such a way, that there are small elements near the atomic centers, while in distant regions, where only smooth tails of wave functions are present, the elements are created as large as possible. Such a mesh can be rather easily created using tetrahedra, but it is rather problematic for hexahedral meshes without hanging nodes.

The following routines for creating meshes in FENNEC are available.

Uniform cube mesh A uniform cube mesh is created by an even division of a rectangular cuboid, thus its elements are rectangular cuboids (most often cubes) of the same size and shape. This type of mesh does not reflect the behavior of the problem and thus using this type of mesh results in unnecessary large matrices. However, such meshes can be easily created with various element sizes to cover the same volume and the meshing does not bring any prior knowledge into the computation. Therefore, these meshes are ideal for convergence analysis.

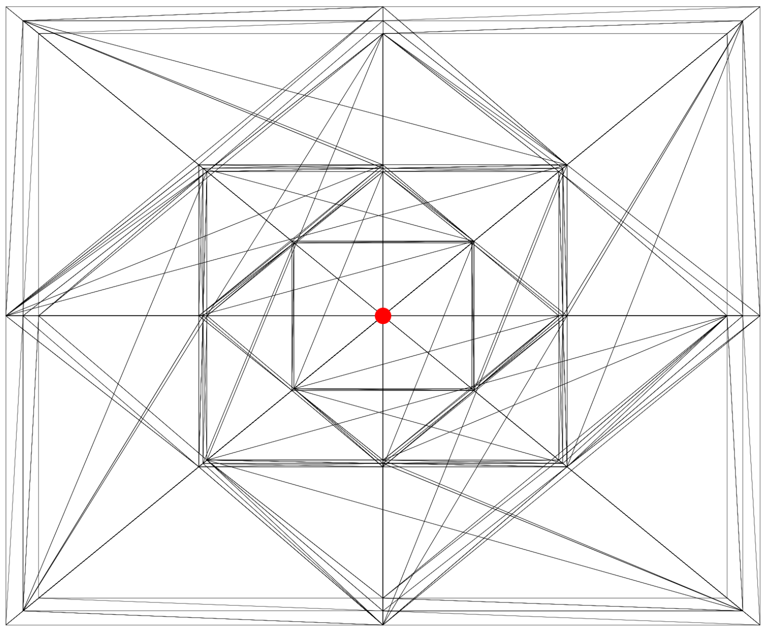


Figure 3.3: An adaptive tetrahedral mesh for one atom, cube of side 20 a.u, $\rho_{\text{fin}} = 3, r = 1$.

Adaptive tetrahedral mesh The advantage of tetrahedra is the possibility to mesh general domain shapes and to be arbitrarily locally refined using the same element type (i.e. the tetrahedra) even without adding vertices to the edges or faces that would lead to hanging nodes. This property of tetrahedra is a great advantage over hexahedra, but to keep good element shapes, refining the edges is required [120], which leads to forced refinements if the hanging nodes are to be avoided, see below.

The idea of how to generate the refined mesh is straightforward: recursively refine all the elements that meet the criteria to be refined, as long as such elements exist. In our case, the finest mesh is required near atomic centers, where the wave functions have the most steepest gradients. Thus the criterion is given by a simple boolean function $f(d(\mathcal{T}), \rho(\mathcal{T}))$, that determines whether the element $\mathcal{T}$ should be refined or not. The function takes two arguments: $d(\mathcal{T})$ — the distance of the tetrahedron from the nearest atomic centre, and $\rho(\mathcal{T})$ — the volume of the tetrahedron.

$$f(d(\mathcal{T}), \rho(\mathcal{T})) = \rho(\mathcal{T}) > \max\left(1, \left(\frac{d(\mathcal{T})}{r}\right)^2\right) \rho_{\text{target}}(s)\,, \tag{3.32}$$

where the amount of refining is parameterized by a minimal distance r and the target finest volume $\rho_{\text{target}}(s)$ (given as the size of a regular tetrahedron with side s). This criterion function is rather simple but it generates sufficiently suitable meshes. For better results in more complex cases, the radius of circumcircle of the tetrahedron could be included in the criterion.

The simplest algorithm to refine a tetrahedron is to create a new vertex in the midpoint of the tetrahedron and divide the tetrahedron into four smaller tetrahedrons. However, such an approach generates ill-shaped tetrahedra, thus a more sophisticated refinement strategy [120] is used: an element is refined so that six vertices are added, each in the middle of each edge of the tetrahedron, and the tetrahedron is divided into eight smaller tetrahedrons similar to the original one (with the same shape).

A drawback of this way of refinement is that the algorithm creates hanging nodes if adjacent tetrahedra have a different level of refinement. Therefore, after performing one level of refinement the elements with hanging nodes on their edges are divided into smaller elements to make the nodes non-hanging. For this purpose, the Delaunay triangulation (see e.g. [121]) is used to ensure a good quality of resulting finite elements.

The algorithm to create the mesh starts with the Delaunay triangulation of the vertices of the volume bounding box and the atomic centers. If there is more than one atomic center, a cube around each atomic center is created and its vertices are added to the starting vertices to ensure a better layout of vertices around the centers.

Algorithm 3.1: Adaptive tetrahedral mesh

```
 1: procedure TETRAHEDRAL ADAPTIVE MESH(atomic centres, distance $d$, max. level of refinement)
 2:     create a cube with atomic centres inside, so that all centres are at least $d$ distant from its sides
 3:     create a set of vertices from vertices of the cube and all atomic centres
 4:     if $count(centres) > 1$ then                          ▷ to avoid long sharp triangles
 5:         for each centre do
 6:             add to the set the vertices of a cube of side $d/2$ with its midpoint located in the atomic center
 7:         average the above added vertices, that are closer to themselves than $d/4$
 8:     create tetrahedron from the set of vertices using the Delaunay triangulation
 9:     repeat
10:         for each (in the previous iteration refined) element do
11:             if Eq.(3.32) does not hold for the element then
12:                 refine the element by adding a vertex on each its edge
13:                 mark all its neighborhoods
14:         for each marked non-refined element do
15:             divide the element to subelements using the Delaunay triangulation
16:     until no element have been refined or max. level of refinement is reached
```

We call the mesh adaptively refined, which may cause some confusion. Adaptivity is usually understood as a dynamical process performed during a computation reflecting its intermediate results. We do not follow this approach, since changing the mesh during computation could (and would) spoil the convergence of the DFT self-consistent cycle. Therefore, in our case the adaptivity may be called *preadaptivity*: the mesh is adapted prior to the computation according to the positions of the atoms.

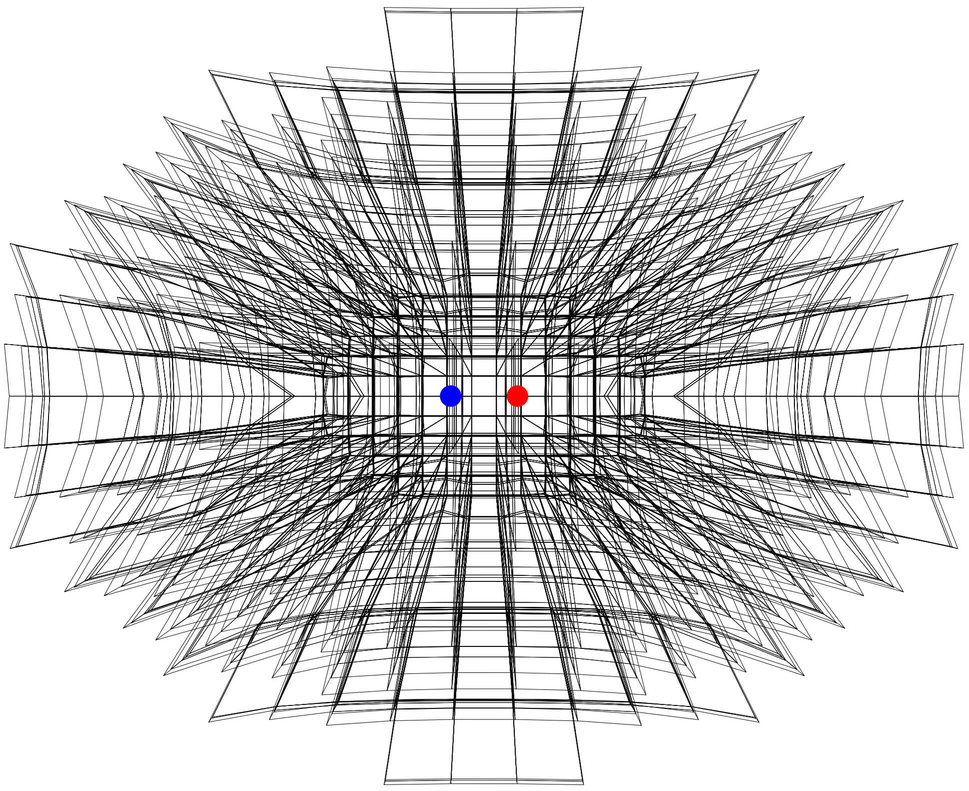


Figure 3.4: An elastically streched hexahedral mesh for nitric oxide. Parameters: max distance=16 a.u., element size=0.8 a.u., min/max radius=2.5 a.u./4 a.u., min/max stiffness=1/3000, stretch factor=3

Elastically stretched hexahedral mesh To reflect the nature of our problem, an algorithm for generating hexahedral mesh with non-uniform elements has been created for computing with earlier versions of SfePy, which did not support the hanging nodes. The mesh is created by stretching elastically a uniform cube mesh. The rigidity of the mesh nodes is set so that it prevents the mesh from being stretched near the atomic centres, while the more distant elements can be stretched to a greater volume. The elements far from atomic centers elements are discarded thereafter (see Fig. 3.4).

Without utilizing the hanging nodes (or non-hexahedral elements), the transition between small and large elements of a hexahedral mesh cannot be done without compromises. It is the principal reason, why a mesh created by this algorithm is far from perfect. But even though the presented algorithm creates prolonged ill-shaped elements in corners of the mesh, we will see (see chapter 5.4.2), that the resulting meshes still offer reasonable numerical properties.

Algorithm 3.2: Adaptive hexahedral mesh

```
1: procedure ADAPTIVE HEXAHEDRAL MESH(atomic centres, distance $d$, minimal element side=$e$, strech factor $r$, minimal and maximal stiffnes $s_{\min}, s_{\max}$ , minimal and maximal radius $r_{\min}, r_{\max}$)
2:     create an uniform cube mesh with the element edge size $e$, so that all atomic centres are inside and at least $d$ distant from the cube sides
3:     remove from the mesh all the vertices in distance greater that $d$ from all the atomic centres
4:     set the maximal stiffness to all the vertices closer than $r_{\min}$ to any atomic center
5:     create a convex hull from these points
6:     for each vertex $\vec{p_i}$ on the mesh boundary do
7:         project $\vec{p_i}$ onto the convex hull [122] as $p_i'$
8:         shift $\vec{p_i}$ by vector $r * (\vec{p_i} - \vec{p_i'})$ and set it fixed
9:     set the minimal stiffness to all vertices more distant than $r_{\max}$ from the convex hull
10:    let $f$ = linearly growing function $(r_{\min}, r_{\max}) \rightarrow (s_{\min}, s_{\max})$
11:    for each remaining vertex $\vec{p_i}$ do
12:        find its minimal distance $m$ to any atomic center
13:        set its stiffness to $f(m)$
14:    treating it as an elasticity problem, find the equilibrium positions of all non-fixed points using FEM
15:    remove from the mesh all the vertices in a distance greater that $d$ from all centers
```

Adaptive hexahedral mesh A hexahedral mesh can be rather easily refined if one can use hanging nodes. During FENNEC development, level-one hanging nodes support has been implemented to SfePy: one hanging node on each edge and one hanging node inside each face have been allowed. An up-down strategy has been chosen for generating such meshes: the algorithm starts with big cubes and it iteratively refines them to smaller ones. This approach is simpler than the down-to-up approach (merging small cubes), yet it can provide suitable meshes if the parameters are appropriately selected. If one starts with a uniform cubic mesh, each element can be refined a maximum of $n + 1$ times, where n is the minimal level of refinements of all its neighbors. See Fig. 3.5 for an example.

The idea of the mesh generating algorithm is very similar to the adaptive tetrahedral mesh algorithm: An element of the mesh is refined, if two criteria are met: first, the element must not be adjacent to an element non-refined in the previous iteration (to have only level-1 hanging nodes), and second, a criterion function must be satisfied. The currently used criterion function is either an always-true function or that the element in the i_{th} step of the refinement is closer than d_i to any atomic center, where d_i are a chosen set of distances: they should be selected accordingly to the nature of the problem.

Algorithm 3.3: Adaptive hexahedral mesh

```
1: procedure Adaptive hexahedral mesh(atomic centres, base cube size $e$, distance $d$, criterion function, maximal level of refinement)
2:     create a uniform cube mesh with the element edge size $e$, such that all the atomic centers are inside and at least $d$ distant from the cube sides
3:     mark the whole mesh as a refinable area
4:     repeat
5:         remove all the boundary elements from the refinable area
6:         refine all the elements in the refinable area that meet the criterion function
7:     until no element refined or the maximal level of refinement reached
8:     remove all the elements more distant than $d$ to any of the atomic centres
```

Which mesh to use?

The analysis including various meshes has been performed and the results can be found in the last section of this chapter (3.4.2). We can observe from the results that even a uniform hexahedral mesh performs better than a refined tetrahedral mesh (in term of the problem size for the given accuracy) since it allows us to use a better (tri-quadratic, tri-cubic . . .) finite element basis. The advantage of refined meshes over the uniform ones can be seen there, too. Therefore, for practical computations with FENNEC, the best choice is to pick the refined hexahedral mesh.

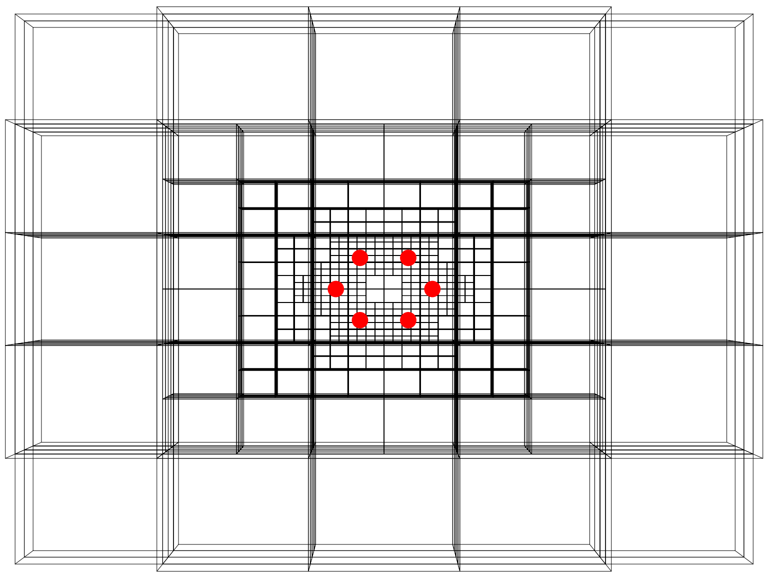


Figure 3.5: An adaptive hexahedral mesh for a single carbon hexahedron. Max. distance=16 a.u., base cube size=8 a.u., four levels of refinement, the criterion function is to have the minimal distance from the atomic centres less than 10 a.u., 8 a.u, 4 a.u. and 2 a.u. respectively.

3.2.2 Finite element and its local properties

In the previous subsection, we divided the volume domain into elements. To allow independent/parallel evaluation of the integral forms over the elements, we need to define a local discretization basis (whose members are called *shape functions*) on each element of the mesh: to construct the *finite elements*.

Finite element

The *finite elements* can be defined in several ways, we will follow the Braess definition [19].

Definition 3.21 (Finite element)**.** *The* finite element $\mathcal{T}$ *is a triplet* $(\mathbb{T}, \mathcal{M}, \Sigma)$ *of a regular polyhedron* $\mathbb{T}$*, a finite vector space of functions* $\mathcal{M}$ *over* $\mathbb{T}$ *(that defines a model reduction) and a linear vector functional* Σ *(that determines the discretization coefficients of the space* $\mathcal{M}$*).*

The finite element has to fulfill the following properties:

1. $\mathcal{M} = \operatorname{span}\{\phi_i(\vec{x}) : \mathbb{T} \to \mathbb{R}\}$.
2. *Let* $k = \dim \mathcal{M}$. *Then* $\Sigma = \{P_i : \mathcal{M} \to \mathbb{C}, i \in \{1 \ldots k\}, P_i \textit{ is linear}\}$.
3. *The mapping* Σ *is unisolvent:* $\forall \mathbf{u} \in \mathbb{C}^k \; !\exists \psi \in \mathcal{M} : \Sigma(\psi) = \mathbf{u}$.

The linear map Σ project any function from the space $\mathcal{M}$ to the to the vector space $\mathbb{C}^k$. The map implicitly defines the basis of $\mathcal{M}$ by the bijection between $\mathcal{M}$ and $\mathbb{C}^k$. Members of the basis are called *shape functions*.

Definition 3.22 (Shape functions)**.** *The* shape functions *of a finite elements are the functions* Φ*, that form a basis of* $\mathcal{M}$ *and have the* δ-property*:*

$$\Phi = \{\phi_i : P_i(\phi_j) = \delta_{i,j}\} . \tag{3.33}$$

For any $\psi \in \mathcal{M}$, the coefficients $\mathbf{u}$, $\Sigma(\psi) = \mathbf{u}$, are called *degrees of freedom* (DoFs), and we have

$$\psi(\vec{x}) = \sum_i \phi_i(\vec{x}) u_i .$$

The concepts of the above definitions are illustrated below using the Lagrange nodal basis and the induced projections, see 3.35).

The definition 3.21 of the finite element could be relaxed to allow $\mathbb{T}$ to be any closed subset of $\mathbb{R}^n$ with nonempty interior and a Lipschitz-continuous boundary [103], such definition is required for the isogeometric analysis, which allows the geometry to be described by NURBS curves (see Section 3.3).

Shape functions and Lagrange basis

The *shape functions* form the basis of a discretization space on a finite element. There are many types of shape functions, that can be used for FEM computations. Here we will describe a construction of one family of shape functions, the Lagrange basis. Bézier polynomials, B-splines, and NURBS bases, defined in the next chapter, are other examples. A basic classification of shape function types is in Section 3.4.1.

Lagrange basis Probably the most common shape functions — and the ones that FENNEC primarily uses — are the Lagrange basis shape functions. This basis is often called the *nodal basis* since the DoFs are directly the values of the approximated functions in given nodes: points in the elements.

A Lagrange basis is constructed using the *Lagrange polynomials*:

Definition 3.23 (Lagrange polynomials). *Given a set of n nodes $x_i \in \mathbb{R}$ (one-dimensional), the* Lagrange polynomials *ℓ_i are n functions defined as*

$$\ell_i^{\{x_1,\dots,x_n\}}(x) = \prod_{j\neq i} \frac{x - x_j}{x_i - x_j}\,. \tag{3.34}$$

Theorem 3.3 (Lagrange projection). *There exists a projection to the space spanned by Lagrange polynomials ℓ_i with n distinct nodes, that is exact ($p = \sum P_i(p)\ell_i$) for each polynomial p up to the order $n-1$.*

Proof. There is a one-to-one correspondence between the nodes and the Lagrange polynomials: each ℓ_i is equal to one in its own node x_i and zero in all the other nodes. Hence, the projections

$$P_i(p) = p(x_i) \tag{3.35}$$

are exact for ℓ_i. ℓ_i are n lineary independent polynomials of order $n-1$, therefore, they form a basis of the polynomial space of order $n-1$, which completes the proof. □

The projections P_i can be used as projections in definition 3.21 for constructing the finite element.

It is a common requirement that the shape functions should be able to represent a constant function exactly. This property is linked to the convergence of the FEM [105] and is automatically fulfilled, if the basis functions form a partition of unity:

Definition 3.24 (Partition of unity). *The set of functions $\beta_i : \mathbb{T} \subset \mathbb{R}^n \to \mathbb{C}$ forms the* partition of unity *on $\mathbb{T}$, if*

$$\forall \vec{x} \in \mathbb{T} : \sum_i \beta_i(\vec{x}) = 1 \tag{3.36}$$

The Lagrange polynomials have this desirable property:

Theorem 3.4 (Partition unity of Lagrange polynomials). *Given the set of n nodes $x_i \in \mathbb{R}$, the corresponding Lagrange polynomials form a partition of unity.*

Proof. ℓ_i are polynomials of degree $n-1$. Their sum $\chi = \sum \ell_i$ is a polynomial of the same degree. According to (3.34) we have for each node x_i

$$\chi(x_i) = 1 \qquad \Longrightarrow \qquad \chi(x_i) - 1 = 0\,. \tag{3.37}$$

A polynomial of the degree $n-1$ that is zero in n points have to be the zero polynomial (this fact can be easily proven by induction). Therefore χ is a constant polynomial equal to one. □

For a multi-dimensional finite element, which is a Cartesian product of one-dimensional subspaces (e.g. a cube), the Lagrange basis can be defined using the product of 1D Lagrange polynomials, determined by a *grid of nodes*.

Definition 3.25 (Grid). *Let X_j be n sets of distinct real numbers. Then the* grid *G_{X_j} is the Cartesian product of the sets. The* dimension *of the grid is n, the* size *of the grid in its i_{th} dimension is $|X_j|$.*

We will use a multiindex (that will serve as a discrete equivalent to Cartesian coordinates) for indexing a grid — for selecting a node of the grid:

Definition 3.26 (Multiindex). *Let G be a grid made from sets X_j. Then the* multiindex $\mathbf{i}$ *into the grid is a member of the set*

$$\{1, 2, \ldots dim(X_1)\} \times \{1, 2, \ldots dim(X_2)\} \times \ldots\ldots \times \{1, 2, \ldots dim(X_n)\}\,, \quad (3.38)$$

Denoting a j_{th} index of a node selected by a multiindex $\mathbf{i}$ as $\mathbf{i}_j$ we can finally define the multi-dimensional *Lagrange basis*.

Definition 3.27 (Lagrange shape functions). *Let G_{X_j} be a grid determined by the sets of numbers X_j and indexed by a multiindex* $\mathbf{i}$, *and let $\ell_i^{(X_j)}$ be the i_{th} Lagrange polynomial for the set of nodes X_j. Then the* Lagrange shape functions *(or the* Lagrange basis*) associated with the grid are the functions defined by*

$$\phi_{\mathbf{i}} = \prod_j \ell_{\mathbf{i}_j}^{(X_j)}\,. \quad (3.39)$$

This definition assigns to each node in a grid a basis function constructed as a product of those Lagrange polynomials, that are nonzero in the given node. Therefore, these functions still holds the property, that each basis function is zero in all nodes except the associated one. In the case of the Lagrange shape functions, the linear mappings P_i (see definition of a finite element (3.21)) are

$$P_i(\psi) = \psi(x_i)\,. \quad (3.40)$$

The Lagrange basis forms a partition of unity, which can be easily proven by induction (by dimension) using Theorem 3.4.

The usual choice of nodes is their even distribution in the element, see the rectangular element in Fig. 3.7 right. A visualization of such 2D Lagrange elements can be seen in Fig. 3.6.

Nodal bases The idea of the Lagrange basis can be generalized even for elements, on which a grid cannot be reasonable defined, e.g. for triangles. There are two ways how to accomplish this. The first is to collapse a hexahedron (or a quadrilateral) to a tetrahedron (or a triangle), unifying corresponding nodes [105].

The second way is a generalization of Lagrange polynomials:

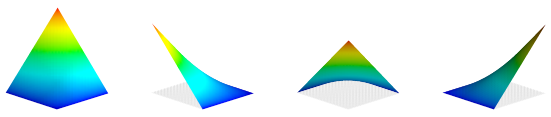
Definition 3.28 (Nodal interpolation polynomials). *Let $\{\vec{u}_i : i \in \{1 \ldots n\}\}$ be a set of n nodes of the dimension d. Let $\mathfrak{p}_i$ be n sets of polynomials in d coordinates. Then, the* nodal interpolation polynomial *has as its basis a set of functions $\{\phi_i : i \in \{1 \ldots n\}\}$ given as linear combinations of $\mathfrak{p}_i$ that satisfies*

$$\forall i, j \in \{1 \ldots n\} : \phi_i(\vec{u}_j) = \delta_{i,j}\,. \quad (3.41)$$

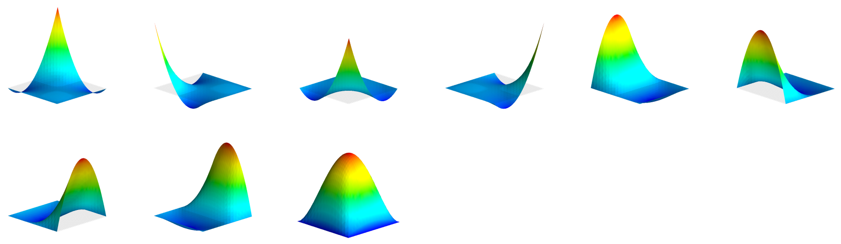
Figure 3.6: Lagrange FEM bases on a reference element for various element shapes and polynomial orders in 2D.

Quadrilateral

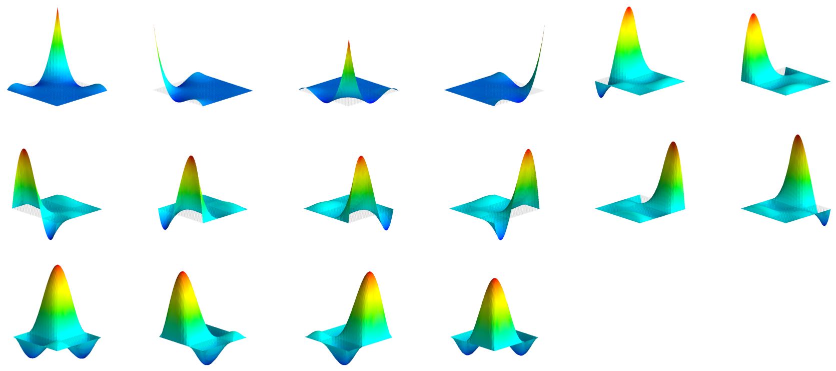
▷ Bilinear



▷ Biquadratic

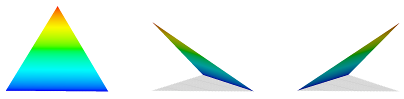


▷ Bicubic

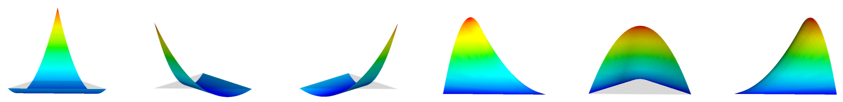


Triangles

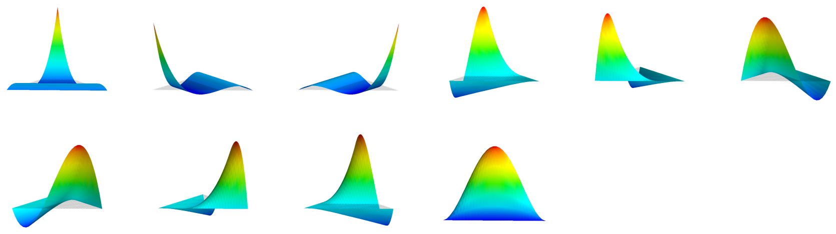
▷ Linear



▷ Quadratic



▷ Cubic



The requirements on the choice of the set of polynomials can be found in [123]. E.g. for a quadratic basis on a triangle, a suitable set of nodes are the vertices and edges midpoints and the suitable set of polynomials p_i is $\{1, x, y, x^2, xy, y^2\}$, see Fig. 3.7.

The use of these multidimensional interpolation polynomials instead of a product of one-dimensional ones is the reason, why while the Lagrange bases on tensor-product elements are linear, quadratic, etc. in each dimension — and therefore (in 2D) bi-linear, bi-quadratics, etc. — the Lagrange bases on simplex elements are only linear, quadratic, etc. See Fig. 3.7 for a visualization of the difference.

It should be noted, that this approach can be used even on hexahedra, the notable example being the *serendipity elements*. These elements are constructed using nodal interpolation polynomials determined by surface nodes of a grid of nodes [123]. This concept can be further expanded to allow more types of degrees of freedom than just function values in nodes. Adding derivatives along the edges of the element as degrees of freedom allows us to define shape functions with continuous derivatives [124].

However, since FENNEC uses mainly hexahedral elements in practice (see chapter 3.4.2) and the details of the Lagrange bases construction on simplex elements or even shape functions with continuous derivatives would not bring more light on the matter now, we refer an interested reader to one of the textbooks covering the topic, e.g.: [124, 123, 105].

Reference mapping

The strength of the finite element method lies in the fact, that although the elements in the mesh can have both different shapes and sizes, they all can be treated (as long as they have the same “topology”, e.g. they all are hexahedrons) in the same way.

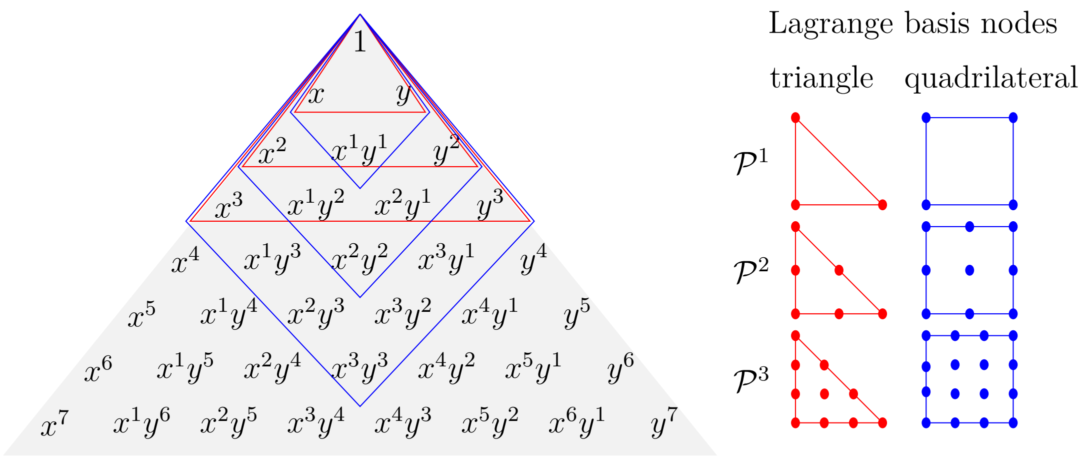

1
x y
x^2 x^1y^1 y^2
x^3 x^1y^2 x^2y^1 y^3
x^4 x^1y^3 x^2y^2 x^3y^1 y^4
x^5 x^1y^4 x^2y^3 x^3y^2 x^4y^1 y^5
x^6 x^1y^5 x^2y^4 x^3y^3 x^4y^2 x^5y^1 y^6
x^7 x^1y^6 x^2y^5 x^3y^4 x^4y^3 x^5y^2 x^6y^1 y^7
Lagrange basis nodes
triangle
quadrilateral
$\mathcal{P}^1$
$\mathcal{P}^2$
$\mathcal{P}^3$

Figure 3.7: Polynomials contained in Lagrange finite element bases on triangles and quadrilaterals. The bases are bi-linear, bi-quadratic, ... on quadrilaterals (blue), while they are only linear, quadratic, ... on triangles (red). [105]

We can perform all the necessary constructions (e.g. of shape functions) and computations on an *reference (finite) element* $\mathcal{T}_{\text{REF}}$ and then transfer the results using the *reference mapping* to the actual elements. The reference element is a fictional element with an "ideal" geometry, e.g. a square with coordinates $\{(-1,-1),(1,-1),(1,1),(-1,1)\}$ for 2D quadrilateral elements, a triangle with coordinates $(0,0),(1,0),(0,1)$ for triangles, or either a cube or a regular polyhedron for 3D cases.

Definition 3.29 (Reference mapping)**.** *Given a finite element $\mathcal{T}_k$, the* reference mapping *of the element is a continuous bijective function F_k from coordinates $\vec{x}^{REF}$ on the corresponding reference element $\mathcal{T}_{REF}$ to the coordinates $\vec{x}$ on the finite element $\mathcal{T}_k$.*

Note the word "corresponding" — if there are more types of elements: e.g. the mesh contains both hexahedra and tetrahedra, or a computed expression contains both volume and surface integrals, each type of elements has to have its own reference element and reference mapping.

Barycentric coordinates For FEM calculations, it is useful to know not only the reference mapping (e.g. to obtain real coordinates of quadrature points, see below), but its inversion, too (e.g. for evaluating a function approximated in the finite element basis in given real coordinates).

Both the mapping and its inversion can be determined by an isomorphism of such a coordinate system (the same on both the reference and the actual element), that is independent on the actual shape of the element: we need a coordinate system, whose coordinates will be retained (in a "topological" sense) under spatial deformations of the element.

Definition 3.30 (Barycentric coordinates)**.** *Given set of n vectors $\vec{v}_i$,* barycentric coordinates[8] *of a vector $\vec{x}$ is such n-tuple b_i that satisfies*

$$\vec{x} = \sum b_i \vec{v}_i \,, \qquad \sum b_i = 1 \,. \tag{3.42}$$

The barycentric coordinates can be viewed as functions of the (reference) Cartesian coordinates on the reference element:

$$b_i = \mathscr{b}_i(\vec{x}^{\text{REF}}) \,. \tag{3.43}$$

The functions $\mathscr{b}_i$ form the *partition of unity* (see definition 3.24). Thus, the coordinates can be viewed as blending the vertices of the reference element (another example of such a blending are the Bézier curves in Section 3.3.1). We can blend the vertices $\vec{v}_i^k$ of an actual finite element in the same way, which defines us the reference mapping:

$$\vec{x} = F_k(\vec{x}^{\text{REF}}) = \sum \mathscr{b}_i(\vec{x}^{\text{REF}}) \vec{v}_i^k \,. \tag{3.44}$$

[8]The definitions of barycentric coordinates differ in literature, e.g. sometimes the predicate "absolute" or "normalized" denotes the requirement $\sum a_i = 1$, or the term is limited to triangle-like elements, where the barycentric coordinates have been invented. Our generalization (definition 3.30) corresponds to "generalized barycentric coordinates".

The barycentric coordinates are especially suitable for simplex elements (triangles, tetrahedra, ...), where they can be easily determined. If we split the triangle by adding a point $\vec{x}$ as a new vertex (see Fig. 3.8), a barycentric coordinate of the point associated with a given vertex can be computed as a normalized volume of the triangle/tetrahedron opposite to the vertex: [125]

$$\mathscr{b}_i(\vec{x}) = \frac{|B_i|}{\sum_j(|B_j|)} \,. \tag{3.45}$$

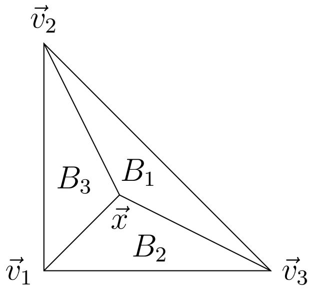


Figure 3.8: Barycentric coordinates on triangles, see (3.45).

This equation can be easily utilized to find the barycentric coordinates of a point on the reference element. To obtain the inversion of the reference mapping, two steps have to be done. First, the right element of the mesh has to be found. For this purpose, the KD-tree [126] is employed in FENNEC for efficient searching. And second, (3.45) can be utilized to find the reference coordinates.

Coordinates in general elements The problem is a bit more complex on a general element since barycentric coordinates are not unique for any element with the number of vertices greater than $\dim(\mathbb{T})+1$ [127]. Therefore, we need to construct a partition of unity, associated with the vertices of the element (the functions $\mathscr{b}_i$), that define the coordinates uniquely.

The functions $\mathscr{b}_i$ form, in fact, a linear Lagrange basis on the triangle associated with its vertices. The natural generalization[9] in the case of a general element is again to employ a (bi-, tri-, etc.) linear Lagrange basis associated with its vertices:

$$\vec{x} = \sum_{\mathbf{i}} \phi_{\mathbf{i}}(\vec{x}^{\mathrm{REF}})\vec{v}_{\mathbf{i}} \tag{3.46}$$

However, the Lagrange basis can be directly constructed only on an element with a regular geometry, or only in 2D [128]. Hence, we have no simple option (as 3.45) to determine the coordinates on a general distorted element. Using a nonlinear (Newton) solver is a common way (e.g. [129]) to obtain $\vec{x}^{\mathrm{REF}}$ for a given $\vec{x}$.

Numerical quadrature

The next important technique in FEM tied to the concept of the reference element is the numerical quadrature: approximating a value of a definite integral over an element. Moreover, the most common way for the numerical integration is again based on the Lagrange polynomials.

Usually, the integrals in a weak formulation of a problem cannot be evaluated analytically, or even an analytical form of the integrated function is not available at all. Thus, a numerical approximation of the integral is often the only feasible way to compute the integrals. A method to accomplish this is usually called either *quadrature*, *numerical integration* or *integration scheme.*

[9]But it is not the only possibility, see [128] for the other possibilities.

Since we have already constructed a mesh to divide the integration domain, it is natural to evaluate integrals as a sum of integrals over individual elements of the mesh using the reference element mapping. The most common class of quadratures is based on a projection of the integrated function f into a space spanned by a basis β_i,

$$f \approx c_i \beta_i \,, \tag{3.47}$$

which yields the following approximation of the integral of a function f:

$$\int_{\mathcal{T}_{\mathrm{REF}}} f \approx \int_{\mathcal{T}_{\mathrm{REF}}} \sum_j c_i \beta_i = \sum_j c_i \int_{\mathcal{T}_{\mathrm{REF}}} \beta_i \,. \tag{3.48}$$

The integrals in 3.48 can be precomputed as *weights*:

Definition 3.31 (Weights (of a quadrature)). *Given an integration scheme as in 3.48, its* weights *are given by*

$$w_i = \int_{\mathcal{T}_{REF}} \beta_i \,. \tag{3.49}$$

It is convenient to use the Lagrange basis for the interpolation (we change the notation of index i to the multiiindex $\mathbf{i}$ to be consistent with Definition 3.27):

Definition 3.32 (Lagrange interpolation polynomial). *Given a grid of nodes* $G = \{\vec{x}^{qREF}_{\mathbf{i}_n}\}$ *on an element* $\mathcal{T}_{REF}$ *and a Lagrange basis* $\varphi_{\mathbf{i}}$ *associated with the grid, the* Lagrange interpolation polynomial *of a function* $f(\vec{x}^{REF})$ *is the function*

$$f(\vec{x}^{REF}) \approx \sum_{\mathbf{i}} f\left(\vec{x}^{qREF}_{\mathbf{i}}\right) \varphi_{\mathbf{i}}(\vec{x}^{REF}) \,. \tag{3.50}$$

Thus we have:

$$\int_{\mathcal{T}_{\mathrm{REF}}} f\left(\vec{x}^{\mathrm{REF}}\right) \approx \sum_{\mathbf{i}} f\left(\vec{x}^{\mathrm{qREF}}_{\mathbf{i}}\right) w_{\mathbf{i}} \,. \tag{3.51}$$

Both the grid and the weights can be easily expressed only on a reference element with a regular geometry. Hence, it is convenient to express all integrals on the reference element and then to transfer them using the reference mapping (3.44) to the actual one [130, 131]. Thus, an integral over a general element can be approximated as:

$$\int_{\mathcal{T}_k} f(\vec{x}) = \int_{\mathcal{T}_{\mathrm{REF}}} f\Big(F_k(\vec{x}^{\mathrm{REF}})\Big) \left|J_{F_k}(\vec{x}^{\mathrm{REF}})\right| \approx \sum w_{\mathbf{i}} f\Big(F_k(\vec{x}^{\mathrm{qREF}}_{\mathbf{i}})\Big) \left|J_{F_k}\Big(\vec{x}^{\mathrm{qREF}}_{\mathbf{i}}\Big)\right| \,. \tag{3.52}$$

Definition 3.33 (Quadrature points). *The quadrature nodes* $\vec{x}^{qREF}_{\mathbf{i}}$ *on the reference element are called the* quadrature points.

The quadrature points can be mapped to a spatial element $\mathcal{T}_k$ using the reference mapping:

$$\vec{x}^{q}_{k,i} = F_k\left(\vec{x}^{\mathrm{qREF}}_{\mathbf{i}}\right) \,. \tag{3.53}$$

where i is a linearization of the multiindex $\mathbf{i}$. If we denote

$$w_{k,i} = w_{\mathbf{i}} \left|J_{F_k}(\vec{x}^{\mathrm{qREF}}_{\mathbf{i}})\right| \,, \tag{3.54}$$

then the integration over the whole domain can be expressed just as a simple dot product of the values of an integrated function in quadrature points and the vector of weights:

$$\int_\Omega f(\vec{x}) \approx \sum_{k,i} w_{k,i} f(\vec{x}_{k,i})\,. \tag{3.55}$$

Gauss quadrature An unanswered question remains, how to choose the nodes that determine the used Lagrange basis. From Theorem 3.3 it follows, that each quadrature that uses the Lagrange polynomials is exact for all polynomials up to the order $n-1$. Using specially selected nodes, the accuracy can be raised. In this sense, the best possible quadrature is the Gauss quadrature.

Definition 3.34 (Gauss quadrature)**.** *The* Gauss quadrature *of the order* $2n-1$ *on an interval I is an integration scheme, that uses the Lagrange interpolation polynomials with n nodes in roots of the n_{th}* monic orthogonal polynomial*[6] on the interval.*

From the orthogonality of the underlying polynomials it follows that the Gauss quadrature with n quadrature points is exact for all polynomials up to the degree $2n-1$; and it is simple to show that no exact quadrature (with n quadrature points) of the order $2n$ can exist. See [6] for a deeper insight into the matter and rigorous proofs of the introduced theorems.[10] The generalization of the (Gauss) quadrature to more dimensions (using multidimensional orthogonal Lagrange basis polynomials) is possible.

From the FEM point of view, the important facts about the Gauss quadrature are:

- ▷ The integration of a function in FEM is done by a scalar product of the *weights* and the function values in *quadrature points.*
- ▷ The used Gauss quadrature is suitable for polynomials, however, it could fail for non-polynomials (e.g. rational) functions. It should be used carefully for such functions and the convergence of the quadrature and the whole method should be verified.
- ▷ In 1D, to have an exact integration scheme for polynomials up to degree $2n-1$, a quadrature with n Gauss points is needed. In 3D in a hexahedral mesh, such quadrature has n^3 quadrature points in each element of the mesh. However, this claim only holds for the integration on the reference element.

  For integration over a general element, the Jacobian J_{F_k} (see 3.52) must be taken into account. In the case of a regular cubic mesh, the Jacobian does not increase the required order. Nevertheless, the Jacobian of a general distorted element is not constant and thus the order of the integration should be increased accordingly. For a general distortion, the reference mapping might not even be polynomial and thus the Gauss integration scheme is not exact in such cases [132].

[10]The cited book is exceptional in the fact, that it does not focus on the one topic, but it reveals deep linkages between various topics from very different fields of mathematics, which provides a much better insight into the matter.

▷ In the case of regular elements, for integrating the charge density (a sum of square roots of wave functions), a quadrature of the exactness $2p$ is required, where p is the maximal order of polynomials in the used basis.

For common problem types and regular elements, the required exactness of quadrature, which does not bring additional error into the finite element method, is $2(p - m)$, where m is the order of the solved differential equation [133].

To conclude, the resulting required exactness of the quadrature is at least $2p$; thus a quadrature with least $p + 1$ points is required (in the case of Gauss integration scheme). However, this will change with the introduction of pseudopotentials, that are rational functions, see page 129 in Section 4.3.2.

3.2.3 Finite element space

Having all the necessary ingredients to build up the finite element space at the local level, we can "glue" the finite elements together.

Finite element space

Equations discretized using FEM (see e.g. 3.26) have usually a form of a sum of scalar products of basis functions, interlaced by an operator. It will be substantial in the later development, whether two basis functions have a nonzero scalar product. The necessary condition for it is, that the multiplied functions *overlap*.

Definition 3.35 (Overlapping of functions)**.** *Two functions* φ_1, φ_2 overlap*, if and only if the intersection of their supports (see definition 2.3 on page 22) is a set of measure greater than zero.*

After this small detour, we can return to building the *finite element space*. We just put the finite elements together:

Definition 3.36 (Finite element space)**.** *The* finite element space *is a set of finite elements with a global* finite element basis*.*

This simple definition does not solve the key problem: how to define the global basis of the finite element space. A trivial finite element basis could be just a union of all shape functions (extended to the whole domain by the zero function) — i.e. that each shape function would have its own "private" degree of freedom. However, such a basis has a major drawback for our application: the elements would not *communicate* due to no overlap between shape functions of adjacent elements, and the functions approximated using such a basis could and would be discontinuous on the element boundaries. In electronic structure calculations, the physical fields need to be continuous.

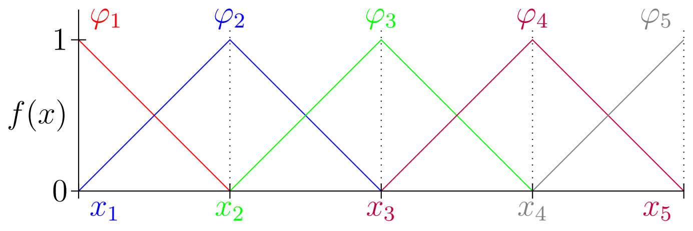


Figure 3.9: Example of a one dimensional linear finite element basis φ_i in a one-dimensional space. A mesh polyhedron $\mathbb{T}_k$ is determined by its two vertices x_k and x_{k+1}. Finite elements are separated by dotted lines. The k_{th} finite element is defined (see Definition 3.21) as

$$\mathbb{T}_k = \operatorname{conv}\left(\{x_k, x_{k+1}\}\right)$$
$$\mathcal{M}_k = \operatorname{span}\{\phi_{k,0}(x) = (x_{k+1} - x)/(x_{k+1} - x_k), \quad \phi_{k,1}(x) = (x - x_k)/(x_{k+1} - x_)\}$$
$$P_j^k(f) = f(x_{k+j}) \qquad j \in \{0,1\}\ ,$$

where a function f is approximated on the element k by

$$f \doteq \sum_j P_j^k(f)\phi_{k,j} = \sum_j f(x_{k+j})\phi_{k,j}\ .$$

If we assign the zero function to $\phi_{0,1}$ and $\phi_{5,0}$ (to treat the borders), the finite element basis is given by functions

$$\varphi_i = \phi_{i,0} \cup \phi_{i-1,1}\ .$$

Continuity

To make the basis continuous and to ensure convergence of the method [134], the shape functions from more (typically adjacent) elements need to be glued together:

Definition 3.37 (Finite element basis)**.** *A global finite element basis is a basis of $\mathcal{W}_h(\Omega)$, such that restrictions of the basis functions to each element $\mathcal{T}_k$ are a linear combination of shape functions of $\mathcal{T}_k$.*

A basis function φ_i corresponding to a global DoF i covers the whole domain, but is nonzero only in the elements that share the DoF i (the small support property, see below).

Note that if we disregard hanging nodes, the restriction of a basis function on an element is directly a shape function — the linear combination of shape functions in the above definition is required to cover the treatment of hanging nodes in practical calculations.

Definition 3.38 ($\mathcal{C}^k$ class (FEM basis continuity))**.** *A finite element basis is in the class $\mathcal{C}^0$, if all the functions of the finite element basis are continuous.*
A finite element basis is in the class $\mathcal{C}^n$, $n \in N$, if all its basis functions derivatives up to n_{th} exist and are continuous.

To ensure a finite element basis to be in $\mathcal{C}^n$, we must construct a basis in such a way that each shape function that is nonzero on a boundary of an element is continuously extended to adjacent elements.

If we denote the i_{th} shape function of the element $\mathcal{T}_k$ as $\phi_{k,i}$, the restriction of the finite element basis function φ_j to $\mathcal{T}_k$ can be expressed as

$$\varphi_j|_k = a_{k,i}^{(j)} \phi_{k,i} \,, \tag{3.56}$$

where the linear combination coefficients $a_{k,i}^{(j)}$ need to ensure the basis continuity.

It can be rather easily done for $\mathcal{C}^0$ continuity on a conforming mesh (without the hanging nodes), where the restrictions of basis functions on an element are directly the shape functions. For example, in the case of the Lagrange basis (see Definition 3.27), whose shape functions coefficients are determined by values in the nodes, it holds that

$$a_{i,j}^{(k)} \in 0, 1 \qquad a_{i,j}^{(k)} = 1 \quad \Longrightarrow \quad \forall l \neq i : a_{l,j}^{(k)} = 0 \,. \tag{3.57}$$

Then the $\mathcal{C}^0$ continuity can be easily ensured by the proper (global) numbering of vertices in the mesh connectivity (i.e. the array of vertex indices that belong to each element). The numbering also allows the *assembling* of the matrices by adding the contributions of each element to the corresponding global matrix items.

Such an approach is not possible in general in the case of a non-conforming mesh: in such a case, a common technique is still to evaluate the quantities on the reference element using shape functions and then use (3.56) to reconstruct the results in the global finite element basis (see e.g. [135]).

The construction of a finite element space with continuous derivatives is much harder to achieve. Common approaches require use of polynomials of the degree $2n + 1$ for the continuity $\mathcal{C}^n$ [103]. The complexity and computational demands of such approaches are the reason, why FENNEC follows the SfePy idea to employ isogeometric analysis for this purpose. This technique will be described in Section 3.3.

So far we assumed the so-called *conforming finite elements.*[19]

Definition 3.39 (Conforming finite elements)**.** *The finite element basis is conforming, if*

$$\mathcal{W}_h(\Omega) \subset \mathcal{W}(\Omega) \,, \tag{3.58}$$

i.e. if the discrete space spanned by the basis is a subset of the function space of solutions.

This property is automatically fulfilled, if the finite element basis is $\mathcal{C}^0$. [136]. It should be mentioned for the sake of completeness that nonconforming finite elements exist (e.g. [137]), where the continuity of basis functions is typically required only on a limited subset of element boundaries (e.g. in given points). Nonconforming elements are typically used in special applications to overcome some difficulties (e.g. a too large number of degrees of freedom required for a conforming method [138]).

Since the nonconforming finite elements have often undesirable properties such as a non-guaranteed convergence[134], common implementations of the FEM are conforming. Therefore, we do not describe the nonconforming variants of FEM and the word “conforming” will be omitted in the following text. However, Ciarlet shows [136] that to achieve the conforming approximation, the scalar products on the original and the discretized spaces need to be the same. This is in fact violated in the case of inexact numerical integration. From this point of view our method could be classified as nonconforming, since rational functions (see Section 4.2) will be integrated.

Small support and sparsity

In practice, the shape functions and finite element basis are usually constructed, so that

- ▷ each shape function is either nonzero in one vertex and zero on all faces that do not contain the vertex, or it is zero in all vertices and nonzero on at most one face (recall the Lagrange bases in Fig. 3.6);
- ▷ finite element basis functions are combined only from the adjacent elements shape functions, see Fig. 3.9 for an example.

Therefore the FE basis functions have a very desirable property: *a small support* (recall definition 2.4 on page 23). The small support of the basis functions is one of the most important properties of the finite element basis since the matrices arising from FEM discretization are then *sparse.*

Definition 3.40 (Sparse matrix). *A matrix M is sparse if the number of nonzero elements in the matrix is substantially lower than the number of all elements of the matrix.*

To ensure the sparsity of the resulting matrices, it is sufficient to meet the requirements of the following theorem:

Theorem 3.5 (Sparsity of local operators). *Let the elements of a matrix M be determined by the following scalar products of finite element basis functions φ_k with operators A and B:*

$$M_{ij} = \int_{\mathcal{W}_h(\Omega)} A\varphi_i B\varphi_j^+ = (A\varphi_i \cdot B\varphi_j). \tag{3.59}$$

Then if φ_k have a small support and A and B are local operators, M is sparse.

Proof. The proof is simple: if basis functions φ_i have a small support, only a few functions overlap. Therefore the matrix $M'_{i,j} = (\varphi_i \cdot \varphi_j)$ is sparse. Since for a local operator (recall Definition 2.1 on page 21) V the support of $\operatorname{supp}(V\varphi_i)$ can

volume dimension	3D
mesh element type	hexahedra
basis type	Lagrange
basis order	cubic
dimension of the matrix	5.672
nb. of elements	32.171.584
nb. of structural nonzeros	526.136

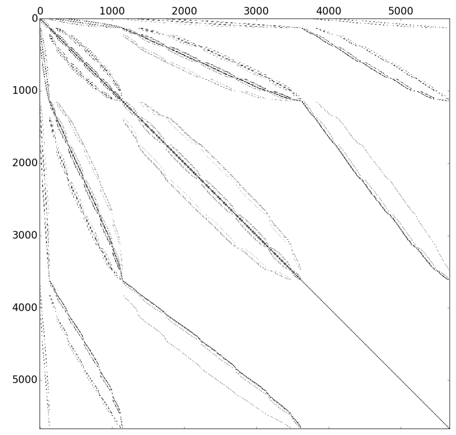


Figure 3.10: Structural nonzeros of the Poisson equation discretized by the 3D FEM cubic Lagrange basis.

differ from $\operatorname{supp}(\varphi_i)$ only on a set of measure zero, the matrix M_{ij} has the same pattern as M'_{ij}. In the other words, a basis with a small support generates sparse matrices and a local operator does not spoil this property. □

The assumptions of the theorem are fulfilled by a vast majority of solved problems using FEM, including the matrices arising from the discretization of the Poisson equation (recall (3.29) on page 60), whose items are given by

$$A_{ij} = -\frac{1}{4\pi}\left(\sum \nabla\varphi_i \cdot \nabla\varphi_j\right), \tag{3.60}$$

since ∇ is a local operator.

Bases with the small support lead to typical structures of nonzeros in matrices (an example can be seen in Fig. 3.10), which allow us to define the following term:

Definition 3.41 (Matrix structural nonzeros). *Let φ_i be a finite element basis and let U, V be local operators, that yield the matrix $M_{ij} = (U\varphi_i \cdot V\varphi_j)$. Then the* (structural) nonzeros *of M are the set of all such index couples i, j that φ_i and φ_j overlap. The* number of (structural) nonzeros*, abbreviated as* NNzs*, denotes the size of this set.*

Note that for a particular operator, even some structural nonzeros can be “accidentally” zero.

Special storage formats exist for sparse matrices that make calculations with sparse matrices much more efficient (see e.g. [139, 118]). Similarly, special sparse matrix algorithms (from many e.g. [140]) lower the memory consumption and the computational time of programs dealing with sparse matrices.

For common sparse matrix formats (e.g. *Compressed sparse row*), the storage and computational demands of many operations (e.g. the matrix-vector multiplication) depend not (only) on the dimension of the matrix, but primarily on the number of structural nonzeros [118], since only nonzero elements of matrices are stored. Therefore, it is desirable to keep the number of nonzeros as low as possible.

In some cases it is useful to express the number of nonzeros relatively to the matrix size:

Definition 3.42 (Fill-in). *A matrix* fill-in *is defined as the*

$$\text{number of structural nonzeros}(M)/\dim(M)\,. \tag{3.61}$$

Projection to a finite element basis

Functions (in our case e.g. wave functions, the charge density, potentials, etc.) can be represented during a computation in two different ways. Results of finite element computations are given as coefficients of the FEM basis (the DoFs). However, this form is not possible for all functions: e.g. the charge density is given as a sum of square roots of wave functions, therefore it does not belong to the finite element space. The potentials of atomic cores, that are expressed using radial bases, also are not expressed in the FEM basis.

The functions, that are not given as DoFs, are represented in FENNEC numerically by their values in given points. For this purpose, the quadrature points (see Definition 3.33 on page 78) are used with advantage, since they allow numerical integration of these functions.

However, it is still necessary to be able to map a general function to the finite element basis, i.e., a *projection into the FE space* from the underlined functional space needs to be defined.

Definition 3.43 (Projection into a FE space). *Given a FE space $\mathcal{W}_h(\Omega)$ with its basis φ_i, which approximates a function space $\mathcal{W}(\Omega)$, a* projection into $\mathcal{W}_h(\Omega)$ *is a function*

$$P_{\mathcal{W}_h^\Omega} : \mathcal{W}(\Omega) \to \mathcal{W}_h(\Omega)\ , \tag{3.62}$$

such that it holds for

$$\psi \in \mathcal{W}(\Omega) \qquad \textit{and} \qquad \psi_h = P_{\mathcal{W}_h^\Omega}(\psi) = a_i\varphi_i\ , \quad a_i \in \mathbb{C}\ , \tag{3.63}$$

that

$$\forall \psi_h' \in \mathcal{W}_h(\Omega) : ||\psi_h - \psi|| \leq ||\psi_h' - \psi||\ . \tag{3.64}$$

The simplest is the L^2 projection, where the norm in Eq. 3.64 is the L^2 norm [124]:

Definition 3.44 (L^2 projection). *The L^2* projection *is a projection $P_{\mathcal{W}_h^\Omega}$ into a FE space with a basis φ_i, for which it holds, that*

$$\forall i : \left(\varphi_i\ \cdot\ (\psi - P_{\mathcal{W}_h^\Omega}(\psi))\right) = \left(\varphi_i\ \cdot\ \left(\psi - \sum_j a_i\varphi_j\right)\right) = 0\ . \tag{3.65}$$

In cases where it is substantial to approximate the derivatives well, a H^1 projection can be used instead — then the used inner product in (3.65) includes not only values of the multiplied functions, but their derivatives too.

A cheap way for evaluating a projection exists for a nodal (e.g. Lagrange) basis: the coefficient (DoF) of the i_{th} basis function can be taken as the value of the function in the corresponding node. This approach also requires the ability to evaluate the function in the nodes of the basis, which is not possible for functions given numerically in quadrature points without further approximation.

3.3 Isogeometric analysis

Commonly used finite element bases have only the $\mathcal{C}^0$ continuity and thus they have discontinuous derivatives on element boundaries. This can be insufficient for some purposes or the resulting precision could suffer (e.g. for forces calculations, that includes differentiation). Therefore it is convenient to have an option to obtain solutions with the continuous first or even the second derivatives.

It is possible to construct a finite element basis of a higher continuity than $\mathcal{C}^0$, but the construction of such a basis is not trivial and to obtain the desired level of continuity of derivatives, high degree polynomials are required (e.g. [103, 141]). Therefore Sfepy and FENNEC employed another approach to obtain bases with globally continuous derivatives: the *Isogeometric analysis*, abbreviated as IGA.

The IGA is an approach for solving partial differential equations, that can be thought of as a variant of the FEM, using *NURBS* (non-uniform rational B-splines) both for the computational domain description and for the unknown field approximation, instead of the usual polynomial bases.

The IGA basis is global on each NURBS patch [142] and thus can have there an arbitrary level of continuity. To facilitate the IGA implementation into existing finite element codes, the techniques of the Bézier extraction [99] or the Lagrange extraction [143] allow local assembling over elements just as in the FEM, however, the necessity of the transformation between the local and the global bases results in greater computationally demands compared to FEM. Another drawback is a lesser sparsity of the assembled matrices, since in IGA bases with higher continuity, more elements overlap than in the common FEM bases.

3.3.1 NURBS basis

As mentioned above, in IGA, unlike in FEM, the geometry of a computational domain is not defined by a mesh, but by using *NURBS* objects. A single object described by NURBS curves/surfaces/solids is called a *patch*. The same approach to the geometry description of objects is used in modern CAD softwares[11]. Although not in electronic structure calculations, this property is a substantial advantage in many other fields where the FEM is employed, since it allows transferring the geometry directly from a CAD to a computational software without the meshing step, which can be all but not trivial (especially for complex structures) and which can bring substantial numerical errors into the computation.

The idea of NURBS came from the fields of interpolation and computer graphics, where NURBS bases were used as blending functions allowing to join smoothly polynomials that interpolate parts of interpolated data sets, or segments of curves or surfaces in computer graphics. We will define the NURBS in the usual way (e.g. [100]): first the *Bézier curves* will be defined, then their generalization: the *B-splines* and finally the *NURBS* themselves.

Bézier curves are defined using *Bernstein basis polynomials.*

[11]Computer-aided design – a software used for design, construction, and engineering.

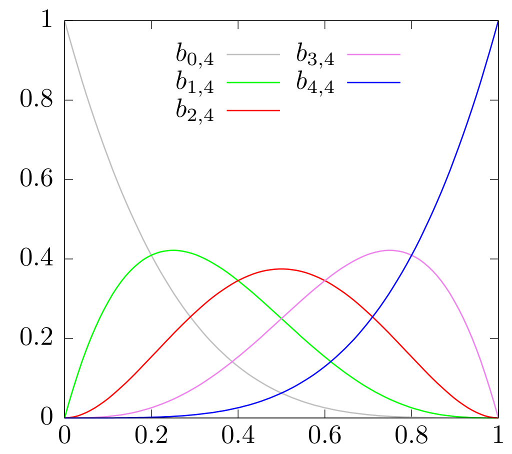


Figure 3.11: Bernstein basis polynomials of the fourth order.

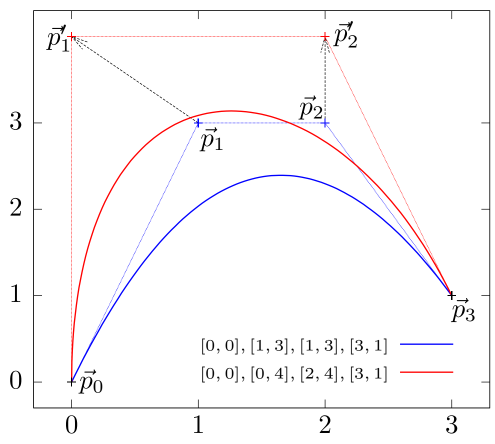


Figure 3.12: Example of reshaping of a Bézier curve using control points.

Definition 3.45 (Bernstein basis polynomials)**.** *The Bernstein basis polynomials of the degree n are polynomials on the interval $[0, 1]$ defined as*

$$\beta_{(i,n)}(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad i \in \{0, \ldots, n\} . \tag{3.66}$$

The Bernstein basis polynomials (see Fig. 3.11 for an example) of an arbitrary degree form the *partition of unity* on $[0, 1]$ (see Definition 3.24 on page 72): this claim can be easily proven by induction. Therefore, they can be used for blending in the same way, as the Lagrangian basis used for the interpolation on hexahedral elements (see Chapter 3.2.2). A Bézier curve is obtained by blending its *control points*, see Fig. 3.12.

Definition 3.46 (Bézier curve)**.** *The degree p* Bézier curve *in a vector space $\mathbb{V}$ is the function $\zeta : (0, 1) \to \mathbb{V}$ determined by $p + 1$* control points $P = \{\vec{p}_i : \vec{p}_i \in \mathbb{V}, i \in \{0 \ldots n\}\}$, *defined as the linear combination of these control points with the degree p Bernstein polynomial basis*

$$\zeta(t) = \sum_i \beta_{(i,n)}(t) \vec{p}_i . \tag{3.67}$$

Bézier curves are common in many fields of computer graphics (e.g. TrueType fonts are defined using this type of curve[12]). However, Bézier curves have a substantial drawback for more complex geometries: either the geometry is described using a single curve — then a change a control point position affects the whole curve. Or the geometry is described using several concatenated Bézier curves — then the continuity of derivatives of the compound curve is lost. Some level of continuity can be achieved by imposing constraints on the adjacent control points, but such constraints propagate local changes to more distant control points.[144] Therefore,

[12]`https://developer.apple.com/fonts/TrueType-Reference-Manual/`

using Bézier curves, one has either the locality of changes, or the smoothness of the resulting geometry, not both.

To overcome the above mentioned problems (and a few other issues, see [107]), *B-spline curves*[13] have been introduced into the fields of computer graphics and geometry description. Splines are defined as piecewise polynomial curves [145]. The term "B-spline" is a bit overloaded, as it denotes both the most local basis of the spline functional space with regard to the required degree and smoothness of the curve (see e.g. [108], the "B" in the name stands for basis), as well as the curves spanned by such a basis, similarly to Bézier curves.

The key component of the theory of B-splines is the *knot vector*.

Definition 3.47 (Knot vector)**.** *The* knot vector $\mathbf{k}$ *of length* $\dim(\mathbf{k})$ *is a non-decreasing sequence of real numbers* $\{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_{\dim(\mathbf{k})}\}$ *of length* $\dim(\mathbf{k})$*. The members of the knot vector are called* knots*.*

Note that the subsequent knots can be equal. This option will be later useful for the Bézier extraction.

We will use sets of knot vector in the following text: to make the notation clear, we will use $\mathbf{k}_i$ for the i-th vector from a given set of vectors, and k_i for the i-th (scalar) member of the vector $\mathbf{k}$ when confusion may arise. For a grid of vectors K made from vectors $\mathbf{k}_i$ the notation $k_{i,j}$ denotes the j-th item of the i-th vector.

Each knot vector determines a corresponding *B-spline basis*:

Definition 3.48 (B-spline basis)**.** *The* B-spline basis *of the order n for the knot vector* $\mathbf{k}$ *with the length* $\dim(\mathbf{k})$ *is a set of functions*

$$B_{\mathbf{k},n} = \left[\beta_{(\mathbf{k},n,i)},\ \ \textit{for } i \in \{1, \ldots, \dim(\mathbf{k}) - n - 1\}\right] , \tag{3.68}$$

recurrently defined as a convolution of a B-splines basis of degree $n-1$ by the Cox-de Boor formula*:*

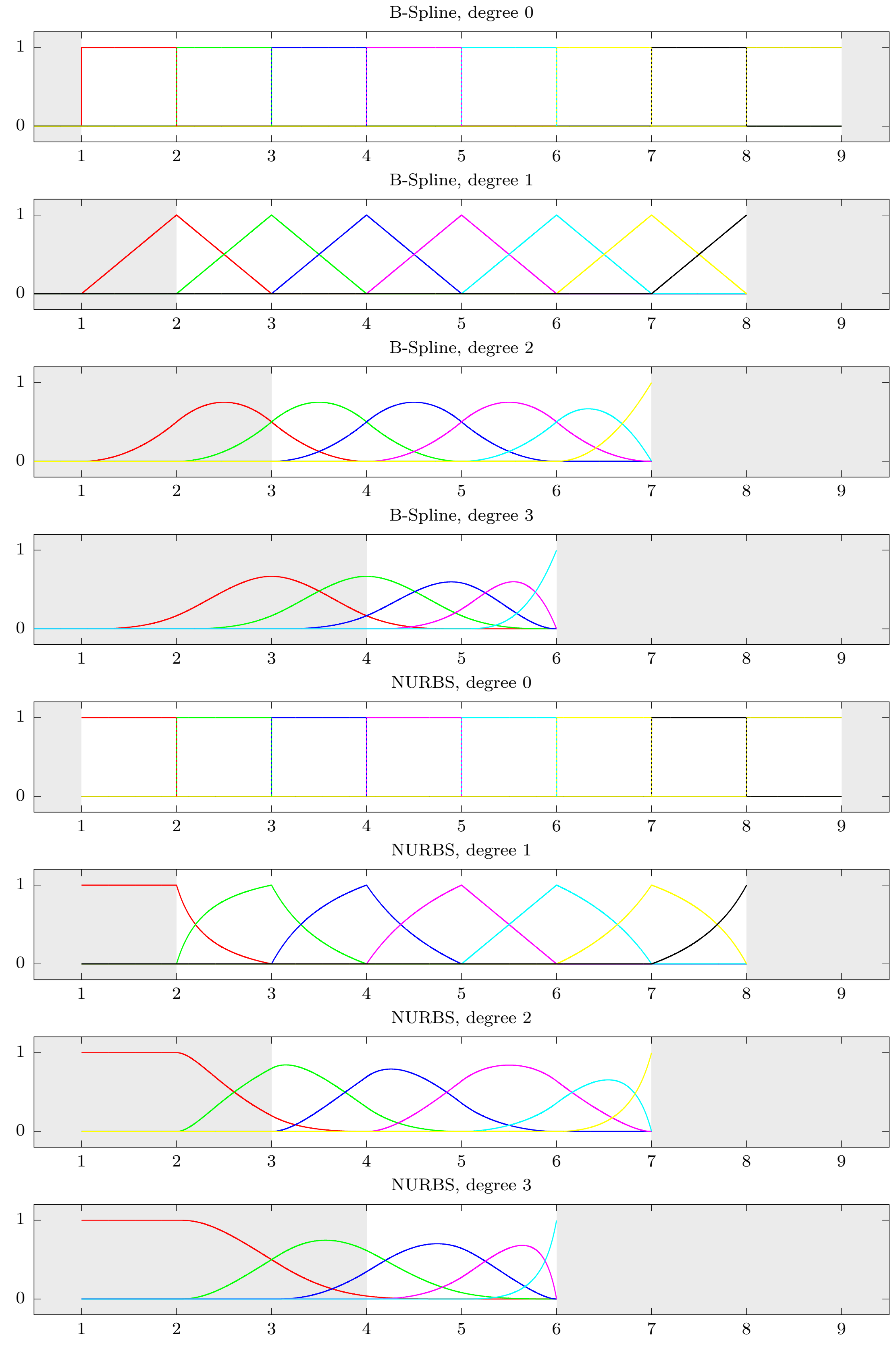
$$\begin{aligned} \beta_{(\mathbf{k},0,i)}(t) &= 1 \quad \text{for } \mathbf{k}_{(i)} \le t \le \mathbf{k}_{(i+1)} , \\ &= 0 \quad \text{otherwise} , \\ \beta_{(\mathbf{k},n,i)}(t) &= \frac{t - \mathbf{k}_{(i)}}{\mathbf{k}_{(i+n)} - \mathbf{k}_{(i)}} \beta_{(\mathbf{k},n-1,i)} + \frac{\mathbf{k}_{(i+n)} - t}{\mathbf{k}_{(i+n+1)} - \mathbf{k}_{(i+1)}} \beta_{(\mathbf{k},n-1,i+1)} . \end{aligned} \tag{3.69}$$

Note that while there are $n+1$ overlapping basis functions in the interior of the knot vector for a basis of the degree n, the first n and the last n knot spans are covered by less basis functions. Therefore, these knots are commonly called the *outer knots* and only the interior knot span is used for the definition of the B-spline curve. It is the knot span, where the B-spline basis forms the partition of unity [146]. An example of a B-spline basis for an uniform (in its interior) knot vector can be seen in Fig. 3.13.

Definition 3.49 (B-spline curve)**.** *The* B-spline (curve) *of the degree n with the basis $B_{\mathbf{k},n}$ in a vector space $\mathbb{V}$ is the function* $\xi(x) : \left[\mathbf{k}_{(n+1)}, \mathbf{k}_{(\dim(\mathbf{k})-n-1)}\right] \to \mathbb{V}$,

[13]Isaac Jacob Schoenberg, inventor of the splines, named the curves by the tool used for drawing smooth curves used by shipbuilders.

Figure 3.13: An example of recursive convolutions of B-splines and NURBS bases of degree n for the knot vector $\mathbf{k} = \{1, 2, \dots, (9-n-1), (9-n), \dots\}$, where the last knot $9-n$ is n-times repeated, and the symmetrical NURBS weights given by $w(i,n) = \min(i, \dim(\mathbf{k}) - n - i)^2$, which yield the following pattern: $\{1, 4, 9, \dots, 9, 4, 1\}$. The grayed parts of the graphs are either the outer knots, knot spans not used for B-spline curves, or the knot span not covered by the knot vector.



determined by $\dim(\mathbf{k}) - n - 1$ control points $P = \{\vec{p}_i \in \mathbb{V},\, i \in \{1, 2, \dots, \dim(\mathbf{k}) - n - 1\}\}$ *(or in matrix notation* $P \in \mathbb{V}^{(\dim(\mathbf{k})-n-1)}$*), and defined by the following linear combination of the control points and the basis:*

$$\xi(t) = \sum_{i=n+1}^{\dim(\mathbf{k})-n} \beta_{(\mathbf{k},n,i)}(t)\vec{p}_i = B^T_{\mathbf{k},n}(t)P\,. \tag{3.70}$$

B-splines can be easily extended to several dimensions as B-spline surfaces, B-spline solids (etc. . .) using a tensor product of B-splines. Such a tensor product is called the *(B-spline) patch.* First, we define a domain for the patch basis functions.

Definition 3.50 (B-splines patch domain)**.** *Let K be a grid from a set of knot vectors $\mathbf{k}_j$, that spans the bases $B_{\mathbf{k}_j,\mathbf{n}_{(j)}}$ of degrees $\mathbf{n} \in \mathbb{N}^j$. Then the B-splines patch domain $\mathbb{D}_{K,\mathbf{n}}$ is the Cartesian product of the inner knot spans of the basis*

$$\mathbb{D}_{K,\mathbf{n}} = \left(k_1^{(1+n_1)}, k_1^{(\dim(\mathbf{k}_1)-n_1)}\right) \times \left(k_2^{(1+n_2)}, k_2^{(\dim(\mathbf{k}_2)-n_2)}\right) \times \dots$$
$$\dots \times \left(k_j^{(1+n_j)}, k_j^{(\dim(\mathbf{k}_j)-n_j)}\right)\,. \tag{3.71}$$

Using the domain, we can define *B-splines patch basis*:

Definition 3.51 (B-spline patch basis)**.** *Given a grid from j knot vectors $K = \{\mathbf{k}_l\}$, indexed by a multiindex $\mathbf{i}$, the* B-spline patch basis *of orders $\mathbf{n} \in \mathbb{N}^j$ is defined as the tensor product of the appropriate B-spline bases:*

$$B_{K,\mathbf{n}} = B_{\mathbf{k}_1,n_1} \otimes B_{\mathbf{k}_2,n_2} \otimes \dots \otimes B_{\mathbf{k}_j,n_j}\;, \tag{3.72}$$

where the members of the basis are the functions $\beta_{K,\mathbf{n},\mathbf{i}} : \mathbb{D}_{K,\mathbf{n}} \to \mathbb{V}^{\dim(\mathbf{n})}$, which are expressed as

$$\forall \mathbf{i} : \beta_{(K,\mathbf{n},\mathbf{i})}(\mathbf{t}) = \prod_j \beta_{(\mathbf{k}_j,n_j,i_j)}(t_j)\;. \tag{3.73}$$

The B-spline patch is then just a generalization of the B-spline curve for several dimensions:

Definition 3.52 (B-spline patch)**.** *The* B-spline patch *for the basis $B_{K,\mathbf{n}}$ determined by a grid of control points $\{\vec{p}_\mathbf{i}\} = P \in \mathbb{V}^{\dim(B_{K,\mathbf{n}})}$ is defined as the function $\xi : \mathbb{D}_{K,\mathbf{n}} \to \mathbb{V}$ by*

$$\xi(\mathbf{t}) = \sum_\mathbf{i} \beta_{(\mathbf{k}_1,n_1,i_1)}(t_1)\beta_{(\mathbf{k}_2,n_2,i_2)}(t_2)\dots\beta_{(\mathbf{k}_j,n_j,i_j)}(t_j)P_{i_1,i_2,\dots,i_j} =$$
$$= \sum_\mathbf{i} \beta_{(K,\mathbf{n},\mathbf{i})}(\mathbf{t})\vec{p}_\mathbf{i} = \left(\prod_j B_{(\mathbf{k}_j,n_j,)}(t_j)\right)^T P\,. \tag{3.74}$$

Because the B-spline curves cannot describe a circle exactly, they have been further generalized to *NURBS — non-uniform rational B-splines.* Here, the basis is defined not only by the knot, but also by *weights* of the knots (see Fig. 3.13 for an example of the difference):

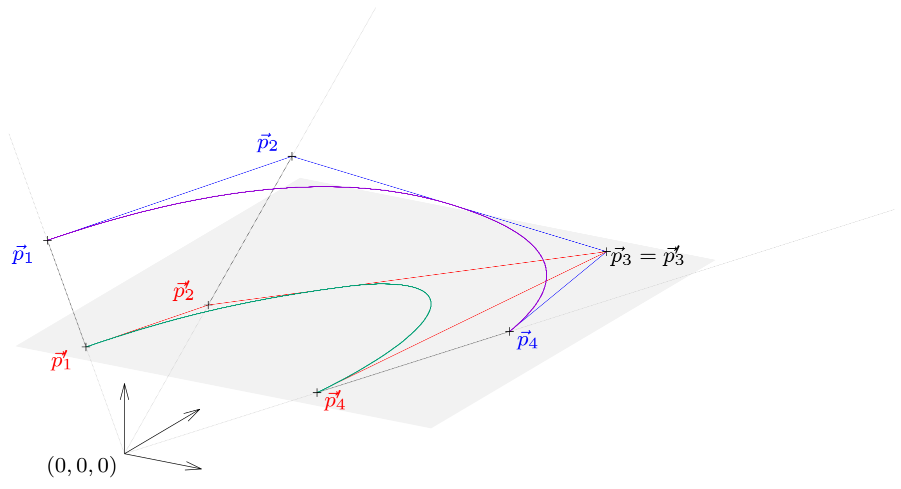


Figure 3.14: A NURBS in a n-dimensional space can be expressed as a projection of a B-spline in a $(n+1)$-dimensional space to a hyperplane defined by $x_{n+1} = 1$ (i.e. perpendicular to the extra axis), where each weight of the NURBS basis is transformed to a coordinate in the extra dimension of the corresponding control point.

Definition 3.53 (NURBS)**.** *The* NURBS basis *of the order n for a given knot vector $\mathbf{k}$ and weights $\mathbf{w} \in \mathbb{R}^{\dim(\mathbf{k})}$ is defined as*

$$\eta_{(\mathbf{k},\mathbf{w},n,i)}(t) = \frac{w_i \beta_{(\mathbf{k},n,i)}(t)}{\sum_l w_l \beta_{(\mathbf{k},n,l)}}(t)\,. \tag{3.75}$$

The NURBS curve *is the spline spanned by the NURBS basis in the same way as the Bézier or B-spline curves.*

The NURBS curves can be generalized to several dimensions as *NURBS patches* in a similar way as B-splines. However, since the basis functions contain the normalization term (the delimiter in 3.75), the *NURBS patch basis* is not a straightforward tensor product of NURBS (curve) bases as in the case of B-splines. To build a NURBS patch basis as a tensor product, one must employ the property, that a NURBS curve can be viewed as a B-spline curve in the space of the dimension higher by one [147] — see Fig. 3.14). It also simply follows from this fact, that the NURBS basis also forms the partition of unity.

For the sake of simplicity, we will follow a simpler and more common approach of defining a multidimensional NURBS basis, which is not formally the tensor product of one-dimensional bases.

Definition 3.54 (NURBS patch basis)**.** *Given a knot vectors and the appropriate spline patch basis as in Definition 3.26, and a Cartesian product of weights as the matrix $W = \mathbf{w}_1 \times \mathbf{w}_2 \ldots \times \mathbf{w}_j$, the* NURBS patch basis *$N_{K,W,\mathbf{n}}$ is a set of functions $\eta(\mathbf{x})_{(K,W,\mathbf{n},\mathbf{i})} : \mathbb{D}_{K,\mathbf{n}} \to \mathbb{V}$ indexed by the multiindex $\mathbf{i}$ and defined as*

$$\eta_{(K,W,\mathbf{n},\mathbf{i})}(\vec{t}) = \frac{W_{\mathbf{i}} \prod_j \beta_{(\mathbf{k_j},n_j,i_j)}(t_j)}{\sum_{\mathbf{i}'} W_{\mathbf{i}'} \prod_j \beta_{(\mathbf{k}_j,n_j,i_j')}(t_j)}\,. \tag{3.76}$$

The definition of the NURBS patch differs from the B-spline patch definition only in the use of NURBS patch basis.

Definition 3.55 (NURBS patch). *Given the NURBS patch basis $N_{K,W,\mathbf{n}}$ and a set of control points $\{\vec{p}_\mathbf{i}\} = P \in \mathbb{V}^{\dim\left(N_{(K,W,\mathbf{n})}\right)}$, the NURBS patch is the function $\mu : \mathbb{D}_{K,\mathbf{n}} \to \mathbb{V}^{\dim(\mathbf{n})}$ defined as*

$$\mu\left(\mathbf{t}\right) = \sum_\mathbf{i} \eta_{(K,W,\mathbf{n},\mathbf{i})}(\mathbf{t})\vec{p}_\mathbf{i} \,. \tag{3.77}$$

3.3.2 Implementation of isogeometric analysis

In the case of electronic structure calculations, where a rather simple volume domain needs to be described — a box is often sufficient — only a single patch is commonly needed.[14] Therefore in later development we will consider only one patch.

In isogeometric analysis, the NURBS patch basis defined in the previous section is used for the discretization of both the geometry and the fields. The patch domain can be seen as equivalent to a reference element. The coordinates $\vec{x}$ on the patch domain take the role of reference coordinates and (3.77) serves as the reference mapping. A degree of freedom is associated with each basis function of the patch and a function $\psi : \Omega \to \mathbb{C}$ is discretized as

$$\psi(\vec{x}) \doteq \sum_\mathbf{i} \mathbf{v}_\mathbf{i}\, \eta_{(K,W,\mathbf{n},\mathbf{i})}(\vec{x}), \quad \mathbf{v} \in \mathbb{C}^{\dim\left(N_{K,W,\mathbf{n}}\right)} \,, \tag{3.78}$$

where $\mathbf{v}$ is the vector of the degrees of freedom.

Bézier extraction

The NURBS patch basis is a global basis — supports of the basis functions can cover the whole patch. The *Bézier extraction* technique can be used to localize the NURBS patch basis onto "elements", defined by non-zero knot spans, that can be used for an integral evaluation and assembling in the FEM sense.

The result of the Bézier extraction is a local basis, defined on each knot span "element", that is independent on the knots, weights and control points of the patch geometry: thus, the same basis is shared by all knot grid elements, which makes the implementation of the IGA very similar to the FEM. Only the main idea of the Bézier extraction will be presented here, see [99, 149] for the full explanation.

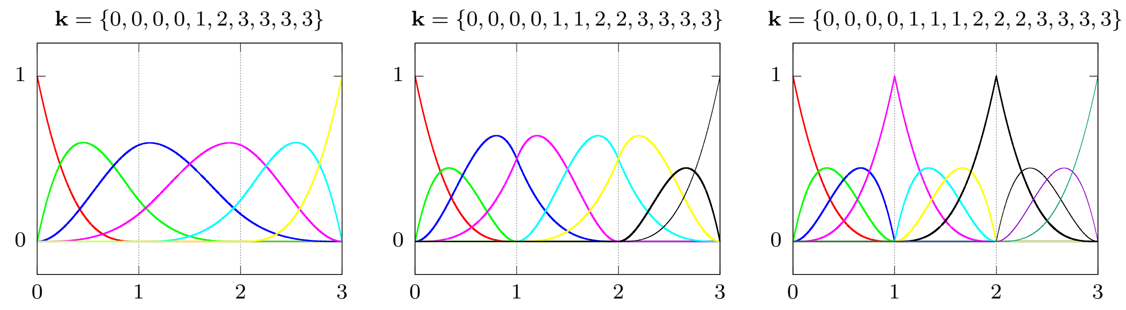
It can be easily shown, that the order n Bézier curves are just a special case of the B-spline curves with the following specific knot vector:

$$\mathbf{k}_{\text{Bézier}} = \{\underbrace{0,\ldots,0}_{p+1},\underbrace{1,\ldots,1}_{p+1}\} \,. \tag{3.79}$$

Since the NURBS can be described as higher-dimensional B-splines, the same idea can be used also for the NURBS.

[14]Some techniques use patches for local refinement, see e.g. [148], such an approach could be used for electronic structure calculation with advantage. However, SfePy does not yet support this technique.

Figure 3.15: Example of the Bézier extraction — the NURBS of the third degree with uniform weights.



We can see in Figure 3.15, how repeating inner knots leads to the formation of a basis local to each knot span. The Bézier extraction is based on the fact, that it is possible to insert knots into a knot vector without changing the geometric or parametric properties of the curve by computing the new set of control points by the knot insertion algorithm [99]. Each multiplicity of a knot in the knot vectors lowers the continuity of the basis by one degree and makes the basis functions more localized. If the multiplicities of the interior knots equal to the degree of the NURBS, the basis is fully localized, like in the FEM. Such a basis can be transformed to the interval $[0, 1]$, then it is equal to the Bernstein polynomial basis (see (3.66)).

The key point of the Bézier extraction is the description of the transformation of the basis function when a knot is (de)multiplied. The transformation — called the *Bézier extraction operator* — can be exploited to divide a NURBS patch into "finite elements with Bernstein basis", to evaluate expressions using this Bernstein basis on a reference element, and then, using the inversion of the Bézier extraction operator, to transform the results back to the original NURBS space. Thus, the Bézier extraction operator can be seen as a reference mapping between the local Bernstein basis and a (spatial part of a) global NURBS basis. For the description of the operator and further details see [99].

This approach allows us to treat the IGA nearly in the same way as the FEM — the computation differs only in the use of different shape functions and in the use of the mapping between the local and global basis. The shape functions are now given by the tensor product of Bernstein polynomials on a reference square or cube. Both the coordinates and the reference mapping are compound: we have the mapping from the reference element to a given knot span element and a mapping from the knot span element to a real volume within the NURBS patch.

The use of the Bernstein basis is not necessary in this approach. The Bézier extraction has been an inspiration for the *Lagrange extraction*, that maps the B-splines directly to the Lagrange nodal basis [143], which allows even greater unification of FEM and IGA programming code bases: in this case, the shape functions are the same, only the reference mappings differ. However, this technique is not implemented in SfePy and thus nor in FENNEC.

Note, that the idea of the Bézier extraction allows constructing a basis with a decreased continuity using multiple knots in the knot vector.

3.4 Convergence analysis of various FEM and IGA bases

The Lagrange basis, described in Section 3.2.2, is the most common, however not the only possible basis for the FEM. Another option is e.g. the Bernstein polynomial basis defined in the previous section, or the hierarchical (Lobatto) basis, which will be introduced later. Moreover, even for a given basis type (e.g. Lagrange), there are several choices: the element geometry, the basis (polynomial) order etc. Therefore, the most suitable basis in our context of the electronic structure calculations should be found.

The purpose of this section is to compare the bases offered by the SfePy package [150] with respect to our application. This section has two parts. In the first part, we summarize the properties of FEM and IGA bases and their differences. The second part of the section contains the convergence analysis of the bases performed on using calculations of the nitrogen atom electronic structure.

Please bear in mind, that this brief summary is far from a fully exhaustive enumeration of all possible variants of the finite element method and all their variants and modifications, it is rather a brief analysis of commonly used disrete function spaces and their main properties, especially with regard to their use within electronic structure calculations and to the possibilities provided by the finite element framework SfePy, which has been used as the FE framework for FENNEC. The wide range of possible FEM bases and their exact description can be found in the references (e.g. [103]).

3.4.1 Classification of bases

The following properties are common to (nearly) all possible finite element bases and follow from the previous topics.

- ▷ **Small support** The FEM and IGA basis functions have the small support. It is the reason for the sparsity of the resulting matrices, as it has been shown in Section 3.2.3.

- ▷ **Non-orthogonality** FEM bases are non-orthogonal: a scalar product of two different basis functions is not zero. Scalar products of the basis functions form the *mass matrix*:

  Definition 3.56 (Mass matrix)**.** *The* Mass matrix *corresponding to the finite element basis φ_i is the matrix M defined by*

  $$M_{ij} = \int_\Omega \varphi_i \varphi_j^+ = (\varphi_i \cdot \varphi_j)\,. \tag{3.80}$$

  For an orthonormal basis, the mass matrix would be the identity matrix. However, neither the NURBS basis (as can be easily seen from the non-negativity of NURBS), nor the commonly used FEM bases are orthonormal nor orthogonal.

- ▷ **Polynomiality** In general, shape functions on a finite element can be constructed using any functions and not just polynomials (e.g. [151]). However, the vast majority of finite element bases are polynomial. Only one non-polynomial basis will be considered in this work: the NURBS basis used in the IGA. However, since NURBS bases are projections of polynomial B-spline bases, they share many characteristics of the polynomial bases (e.g. the degree).

- ▷ **Partition of unity** Both shape functions and FEM bases usually form the partition of unity, meaning a constant function can be represented exactly.

Differences of finite element bases

The following properties are characteristic for each particular basis kind.

- ▷ **Dimension** The dimension of basis functions — whether they are suitable to describe functions on curves, surfaces, or volumes (or higher-order spaces).

- ▷ **Reference element shape** A “topological” shape of a finite element determines the type of the underlying reference element. The most commonly used shapes for 3D finite elements are tetrahedra and hexahedra.

  Tetrahedral elements offer a great meshing flexibility and the possibility to refine a mesh without ill-shaped elements or hanging nodes. As a trade-off, the (Lagrange) shape functions on hexahedra have higher order terms (e.g. tri-linear instead of just linear, see Fig. 3.7), and therefore provide (as we will see) a substantially better approximation of discretized functions.

  This classification is not directly applicable to IGA patches, however, since the IGA basis is given by a tensor product, it can be viewed as a special case of a hexahedral mesh with possibly curved faces: the Bézier extraction even emphasizes the analogy.

- ▷ **Basis type** Shape functions can be constructed in many ways. SfePy offers two standard types: the Lagrange (or nodal) basis and the Lobatto (or hierarchical) basis.

  The *Lagrange basis* has been introduced in Section 3.2.2. This type of basis allows cheap projections into the basis (see Section 3.2.3).

  The disadvantage of the Lagrange basis is the fact, that the basis functions are not *hierachical*: a higher-order basis does not include the functions from the bases of lower orders. This property is not desirable in the case when the *p-adaptivity* is employed: when elements with different basis orders are used in the FE space. In such a case a *hierarchical* basis can be used with advantage. The construction of the Lobatto basis — a hierarchical basis implemented in SfePy — can be found e.g. in [115, 152].

  Another types of bases are the NURBS bases described in the previous section, or serendipity elements (see Section 3.2.2).

  An example of 2D Lagrange bases is shown in Fig 3.6, see [153] for examples of hierarchical bases.

- ▷ **Basis order** Since all considered bases are polynomial, we can classify them according to the highest degree of the polynomial that is included in them. Generally, the higher the order of the basis is, the more functions belong to an element and overlap, which results in the greater fill-in of resulting matrices. On the other hand, higher degree polynomials usually offer a better approximation of functions and thus better convergence, so for the best computational efficiency, the sweet-spot should be found. We will denote linear, quadratics, etc. bases as $\mathcal{P}^1$, $\mathcal{P}^2$... and tri-linear, tri-quadratics etc. as $\mathcal{Q}^1$, $\mathcal{Q}^2$... An example of Lagrange bases of various orders is in Fig. 3.6.

- ▷ **Continuity** As mentioned in Section 3.2.3, bases can be classified by the continuity of derivatives of their basis functions. Whereas all the available FEM bases in SfePy belong to $\mathcal{C}^0$, the single patch IGA bases can be easily constructed with a desired degree of continuity: an IGA basis of order n can belong to any class from $\mathcal{C}^0$ to $\mathcal{C}^{n-1}$ (see the Bézier extraction in the previous section).

- ▷ **Fill-in and sparsity pattern** An important property of a basis is the number of overlapping functions, that depends on the order of the basis. This property determines the amount of sparsity of the resulting matrices and their sparsity pattern (see Fig. 3.10 on page 84 for an example).

  Only adjacent element basis functions overlap in the FEM. On the contrary, in the one dimensional case, a member of the $\mathcal{C}^n$ IGA basis overlaps $\max(n+1)$ "elements" (see Fig. 3.15 on page 93). Therefore, higher-continuity-degree (IGA) bases have a substantially greater fill-in.

3.4.2 Convergence and efficiency analysis of FEM and IGA bases

In this section, the suitability of the various types of bases for electronic structure calculations by FENNEC will be examined. For this purpose, sample computations using FENNEC have been performed and the convergence of the DFT cycle has been analyzed.

Metric

To find the best suitable basis for electronic structure calculations, first, the right metric should be found.

All the considered metrics are based on a single iteration of the DFT self-consistent cycle (see chapter 2.5 on page 36) and do not include the number of iterations of the DFT cycle needed to converge. This decision has two reasons. First, the numbers of iterations do not differ for bases with a similar precision. Second, a more precise approximation leads to fewer iterations than a less precise one. It is partially caused by the chosen initial charge density, that is constructed using the radial atom density, but our experience with the calculations shows, that a more "physical" setup (including the accuracy of an approximation) leads usually

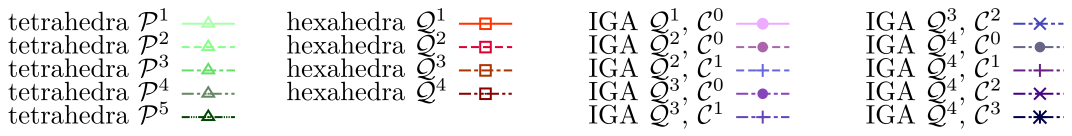

tetrahedra $\mathcal{P}^1$
tetrahedra $\mathcal{P}^2$
tetrahedra $\mathcal{P}^3$
tetrahedra $\mathcal{P}^4$
tetrahedra $\mathcal{P}^5$
hexahedra $\mathcal{Q}^1$
hexahedra $\mathcal{Q}^2$
hexahedra $\mathcal{Q}^3$
hexahedra $\mathcal{Q}^4$
IGA $\mathcal{Q}^1$, $\mathcal{C}^0$
IGA $\mathcal{Q}^2$, $\mathcal{C}^0$
IGA $\mathcal{Q}^2$, $\mathcal{C}^1$
IGA $\mathcal{Q}^3$, $\mathcal{C}^0$
IGA $\mathcal{Q}^3$, $\mathcal{C}^1$
IGA $\mathcal{Q}^3$, $\mathcal{C}^2$
IGA $\mathcal{Q}^4$, $\mathcal{C}^0$
IGA $\mathcal{Q}^4$, $\mathcal{C}^1$
IGA $\mathcal{Q}^4$, $\mathcal{C}^2$
IGA $\mathcal{Q}^4$, $\mathcal{C}^3$

Figure 3.16: Average time of an iteration of the DFT cycle with regard to the matrix size (left) and to the number of nonzeros (right) — a comparison of tetrahedra and hexahedra.

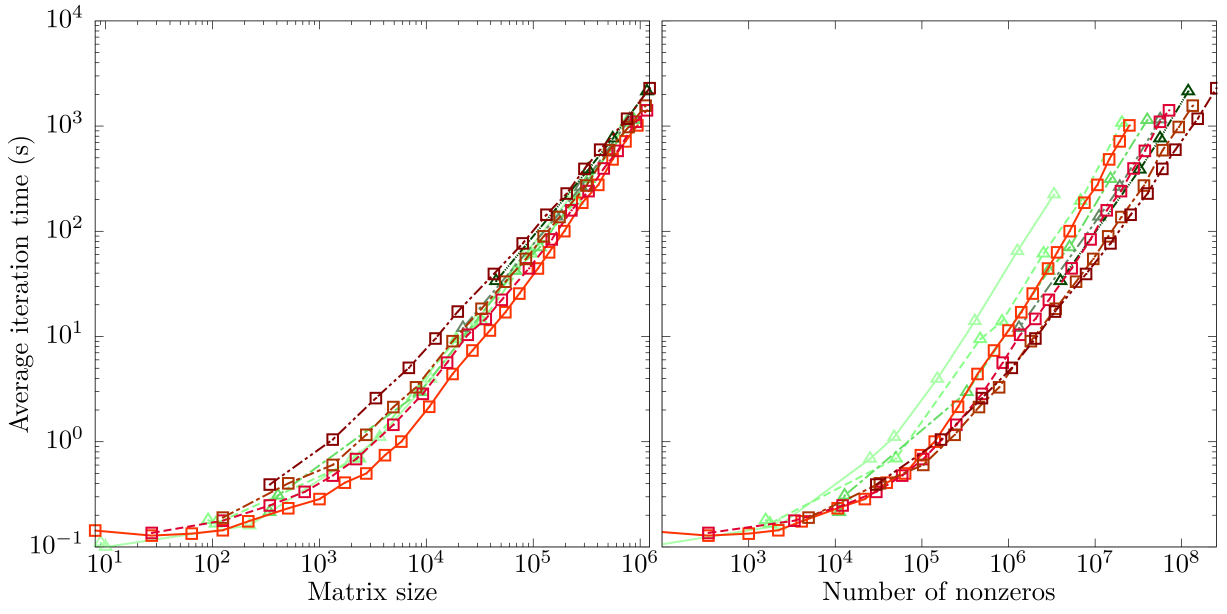
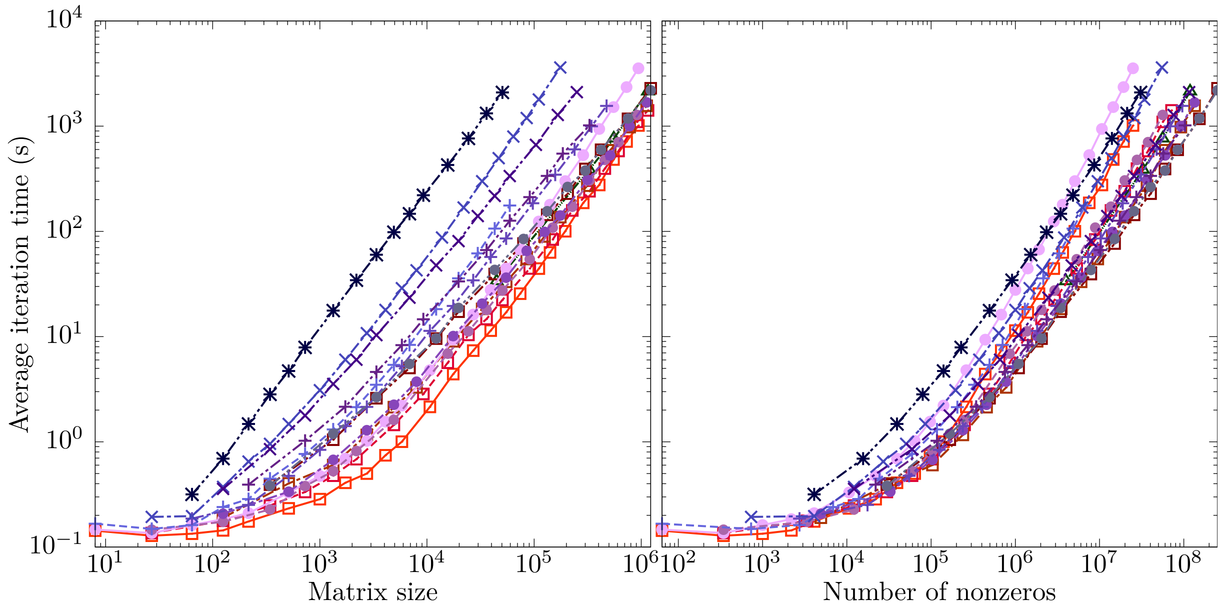


Figure 3.17: The time of an iteration of the DFT cycle to compute the nitrogen atom with regard to the matrix size (left) and to the number of nonzeros (right) — a comparison of FEM and IGA.



to a better convergence of the DFT cycle. Therefore, judgments based on the number of iterations for different accuracies of approximation could be misleading.

For measuring the computational demands for a given basis, several metrics can be used. The natural one seems to be the computation time on a reference hardware. However, such a metric is too implementation-dependent: using different eigenvalue solvers or better optimization of the crucial part of computing, etc. would affect the results, just as the platform-specific properties (sizes of caches, implementation of mathematical libraries, etc.). Thus, this metric will be used primarily only for confirming that the other chosen metrics are reasonable — see Fig. 3.16 and 3.17.

The next possible metric could be the number of finite elements. Nonetheless, this metric is unsatisfactory: different basis types yield different numbers of basis functions for the same mesh, and thus, for the same number of finite elements (see Fig. 3.6). Hence, this metric is not able to distinguish such cases.

The elapsed time for electronic structure calculations using the FEM depends predominantly on the performance of the underlying linear algebra solvers. Since the matrices resulting from the discretization are sparse, the performance of the solvers depends not only on the dimension of the resulting matrices but on the number of nonzeros of the matrices, too. Two metrics suggest themselves naturally:

number of degrees of freedom (matrix size) the number of degrees of freedom (abbreviated as DoF's) is the number of basis functions used in the computation. This number is equal to the dimension of the resulting matrices and determines memory requirements for storing vectors.

number of (structural) nonzeros the number of (structural) nonzeros (abbreviated as NNzs), which determines both the number of scalar multiplications needed for one matrix-vector multiplication and the memory requirements for matrix storage.

The obtained elapsed times in Fig. 3.16 indicate that the number of DoF's seems to better predict the computational time for different shapes of the bases, but it does not predict well the computation time ratio for different degrees of bases on hexahedral meshes. On the other hand, the number-of-nonzeros metric underestimates the required time for tetrahedra.

The DoF's metric does not perform very well for the case of IGA bases with a higher degree of continuity of derivatives (see Fig. 3.17), which is caused by the fact, that these bases have a too low ratio $^{\text{number of DoF's}}/_{\text{NNzs}}$ (see Fig. 3.18) — an increasing continuity of IGA bases causes a greater fill-in of matrices since NURBS basis elements overlap more "elements" (see Fig. 3.15 on page 93). Therefore, the component of the complexity depending on the number of nonzeros prevails in this case.

Another observation is that the DoF's metric is better asymptotically for the FEM bases, while the NNzs metric yields better results for smaller matrices: this phenomenon is caused by the used eigenvalue solvers and thus it is implementation-dependent. To conclude, none of the suggested metrics by themselves are fully satisfactory, but their combination covers the important aspects of the bases comparisons. Thus, both metrics will be used in the following analysis.

We can also notice that the performance of the current implementation of the IGA is noticeably slower than the FEM, especially for a higher degree of continuity of derivatives. We are not sure if this is solely due the more demanding evaluation of the NURBS basis as compared to polynomials, or also with a possible under-optimal implementation of the IGA in SfePy.

The dependence of the number of nonzeros on the matrix size can be seen in Fig. 3.18. Since the number of nonzeros depends on the matrix size and on the number of overlapping basis functions (which is determined by the basis type), the resulting dependence is nearly linear.

Criteria, considered bases and setup of the evaluation

To pick the best basis for an electronic structure calculation, the following bases have been evaluated. First the tetrahedral and hexahedral elements have been compared using Lagrange polynomial bases of the degrees up to five or four, respectively. Then the Lagrange and Lobatto bases have been compared on hexahedral elements (which yielded better results in the previous comparison).

We have already described, that SfePy offers two techniques that can "improve" the basic FEM method: h-adaptivity (mesh refinement) and IGA: those were compared with the "basic" FEM to demonstrate their advantages. Since they cannot be used together yet, they have been compared with a "standard case" and with each other to pick the best possibility.

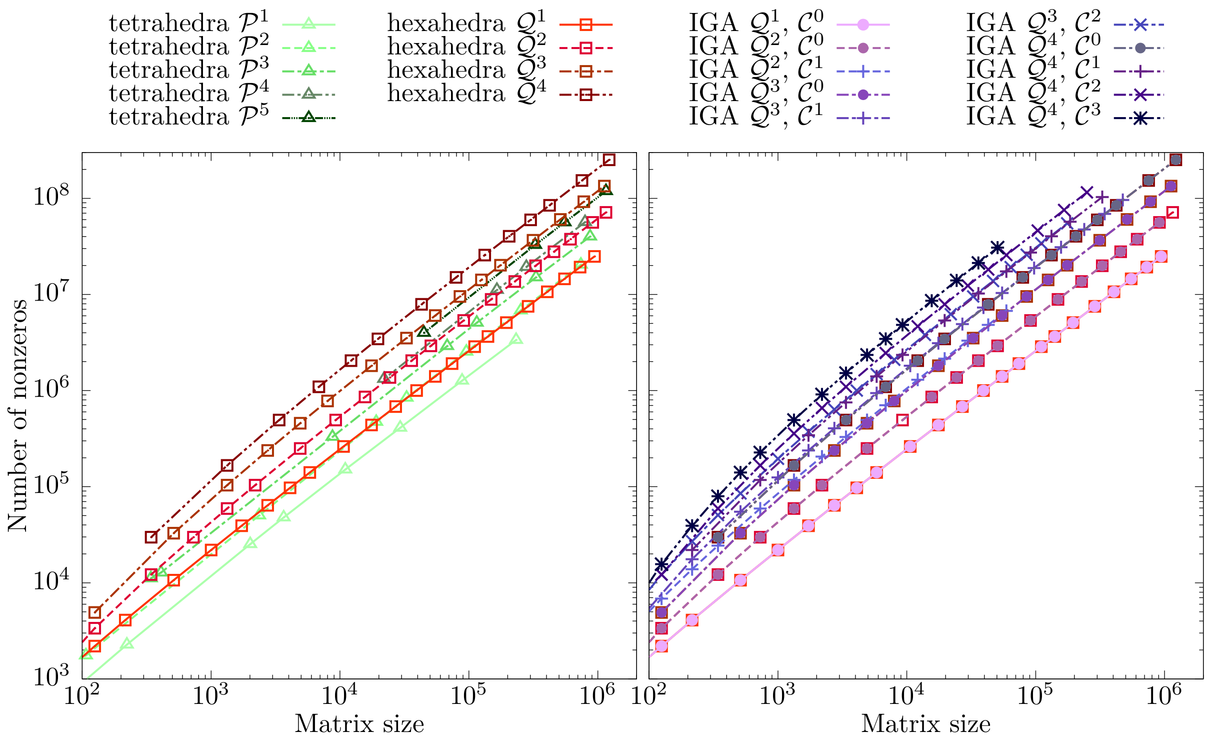


Figure 3.18: The dependence of the number of nonzeros on the matrix size – a comparison of tetraheadra and hexahedra (left) and FEM and IGA (right).

Setup of the analyssis The uniform hexahedral meshes covered a cube of a side of 32 atomic units and were uniformly divided to $n \times n \times n$ elements.

The same setup has been used for the IGA — the patches have been generated using uniform knot grids, which yielded the patches equivalent to uniform hexahedral meshes. The uniform tetrahedral meshes have proven to be uncompetitive, therefore, they were not included in the comparison. However, the advantage of tetrahedra is that those can be easily refined. To reflect this behavior, adaptively refined tetrahedral meshes covering the same volume as above were used for comparison with the hexahedral meshes.

For the adaptively refined hexahedral mesh case, the uniform cube mesh covering the same volume as above with the element edge size 4 a.u was generated. The mesh was then refined by a varying number of steps, see Section 3.2.1 for the details of the refined mesh creation.

A nitrogen atom was placed in the center of each mesh, the initial charge density was set to the density obtained from a radial computation and the DFT self-consistent cycle was then run. If the cycle converged, the resulting eigenvalues and the charge density were compared with the results from the radial computation.

We analyzed the convergence results to find the most suitable basis for the electronic structure calculations. The ideal basis has to be not only the most efficient (in terms of the metrics mentioned above), but it also has to have the following properties:

DFT cycle stability The fundamental property required is that a basis must not to spoil the convergence of the DFT cycle (by numerical instabilities) or of the eigenvalue solver. Otherwise the computation time is substantially prolonged or even the DFT cycle fails to converge.

These criteria were fulfilled by nearly all the considered bases: only several cases of a refined tetrahedral mesh had problems with the convergence.

(discretization) stability The computed quantities should converge with respect to the number of degrees of freedom (the size of the basis). We considered the speed of convergence of the resulting charge density and the first eigenvalue. No significant differences were observed between the convergence results obtained from the first and the second eigenvalues except the refined hexahedral mesh case.

consistency The computed quantities should be a good approximation of the real ones. Because the exact solution of the problem on the cube are not known, the results from a radial computation (obtained by the LDAT code for creating pseudopotentials [20]) are used as the reference values. They necessarily differed a bit, due to different boundary conditions, however its precision is sufficient for our purpose.

monotonicity The monotonicity of the convergence with respect to the mesh element size is another desirable property that allows approximating better the computational error.

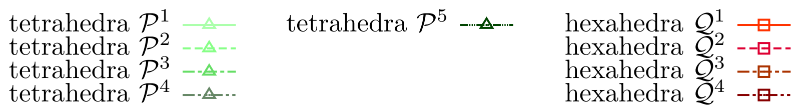

tetrahedra $\mathcal{P}^1$
tetrahedra $\mathcal{P}^2$
tetrahedra $\mathcal{P}^3$
tetrahedra $\mathcal{P}^4$
tetrahedra $\mathcal{P}^5$
hexahedra $\mathcal{Q}^1$
hexahedra $\mathcal{Q}^2$
hexahedra $\mathcal{Q}^3$
hexahedra $\mathcal{Q}^4$

Figure 3.19: The error of the Nitrogen atom charge density with regard to number of DoFs (left) and NNzs (right) — a comparison of hexahedra and tetrahedra.

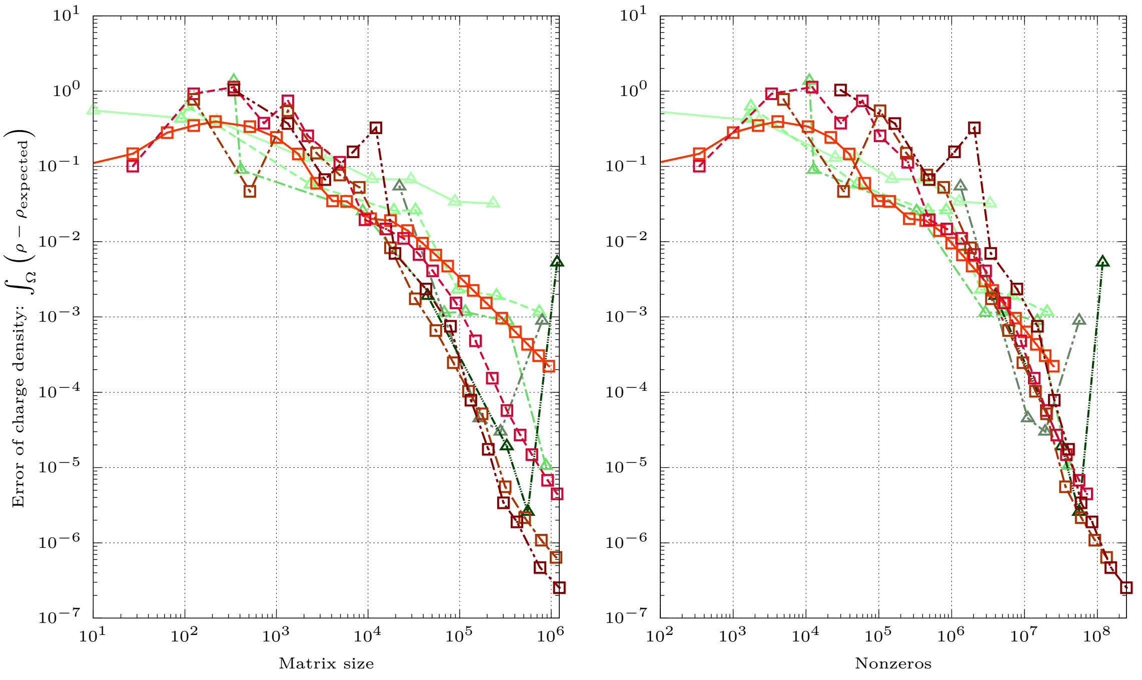
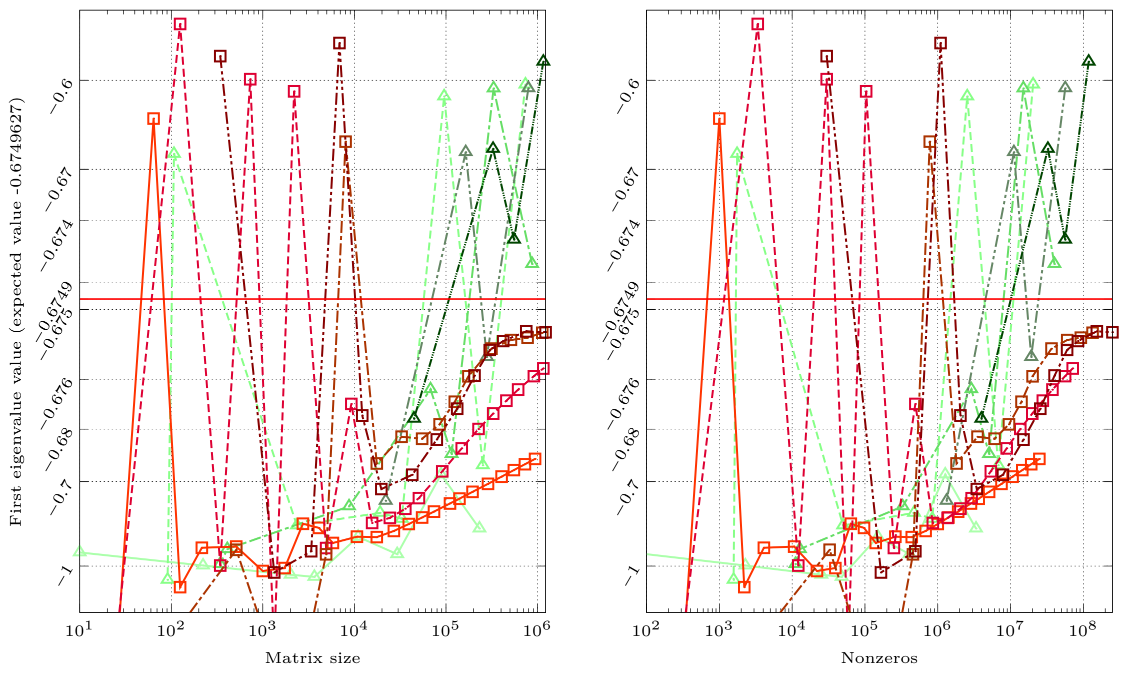


Figure 3.20: The first eigenvalue of the Nitrogen atom with regard to number of DoFs (left) and NNzs (right) — a comparison of hexahedra and tetrahedra. The solid red line denotes the expected value −0.6749627 obtained from a radial computation.



Comparison of hexahedra and tetrahedra

The results shown in Fig. 3.19 indicate that none of the considered bases spoils the DFT convergence for a sufficiently rich (at least 10.000 DoFs) basis. There are some missing results for large tetrahedral elements of a higher order, because matrices generated by those setups caused that the eigenvalue solver did not converge.

The convergence of eigenvalues in Fig. 3.20 is much worse in the case of tetrahedral bases than for any of the hexahedral bases, and it is not monotone. The oscillations, similar to those that the tetrahedral meshes suffer, can also be observed on the coarse hexahedral meshes. However, the hexahedra meshes convergence stabilizes much earlier. Therefore, we can conclude that for the purposes of the DFT self-consistence calculations, the tetrahedral bases are not suitable.

The better performance of hexahedral elements comes probably from higher polynomial order and thus the higher precision of, e.g. the tri-cubic shape functions on hexahedra than the cubic functions on tetrahedra (which have comparable numbers of DoFs for comparable element sizes).[15]

The hexahedral meshes proved so decisively advantageous in comparison with the tetrahedral meshes that the latter were rejected in the early stages of the FENNEC development. The purpose of the above comparison was to find a suitable element/basis type, not to do a theoretical analysis of the finite elements. The obtained results were sufficient for this aim, so no further effort was devoted to re-examine the phenomena.

Comparison of the polynomial orders

Figures 3.20 and 3.19 illustrate also the impact of the degree of the hexahedral bases on the convergence results: the convergence improves with increasing the polynomial order up to three. The basis of the third and the fourth degrees have nearly the same quality with respect to the matrix size, however, the basis of the third degree is better in terms of the number of nonzeros.

If we consider that and that smaller elements for comparable numbers of degrees of freedom provide a greater freedom in the meshing, and the better performance of third-degree bases in the case of refined meshes (see Fig. 3.25 bellow), we can conclude that the third-degree polynomials seem to be the best from the considered polynomial orders.

Lobatto versus Lagrange bases

Only small differences can be observed for the Lobatto and Lagrange bases. They yield nearly identical results for charge density convergence (see Fig. 3.21), whereas the resulting first eigenvalues slightly differ (see Fig. 3.22) for higher-order bases. This is rather expected behavior since the Lobatto and Lagrange bases are the same for the first degree and their differences grow with their degree. The results of the

[15]An unverified hypothesis is, that a part of the oscillations on tetrahedra may come from the fact, that the nitrogen atom has only the radial symmetric s orbitals and p orbitals, that are in some sense *perpendicular* to each other. Therefore, a hexahedral meshes maybe could better retain the symmetry of the wave functions than a tetrahedral mesh.

hexahedra $\mathcal{Q}^1$
hexahedra $\mathcal{Q}^2$
hexahedra $\mathcal{Q}^3$
hexahedra $\mathcal{Q}^4$
hexahedra Lobatto $\mathcal{Q}^1$
hexahedra Lobatto $\mathcal{Q}^2$
hexahedra Lobatto $\mathcal{Q}^3$
hexahedra Lobatto $\mathcal{Q}^4$

Figure 3.21: The error of the Nitrogen atom charge density with regard to number of DoFs (left) and NNZs (right) — a comparison of the nodal Lagrange and hierarchical Lobatto bases.

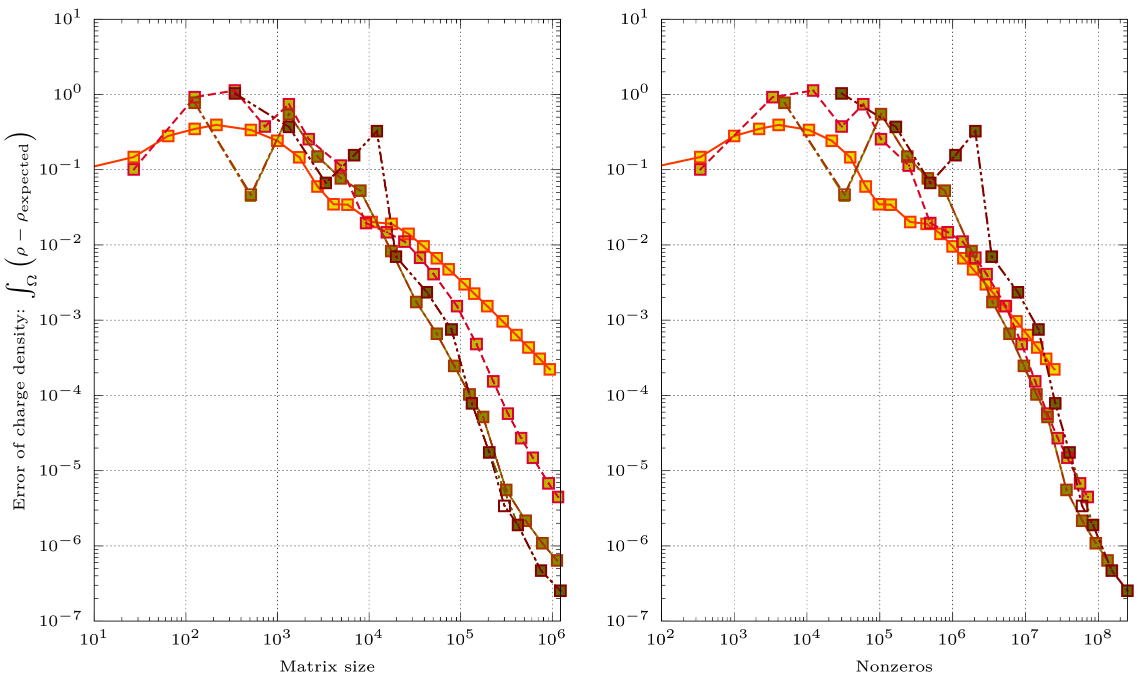
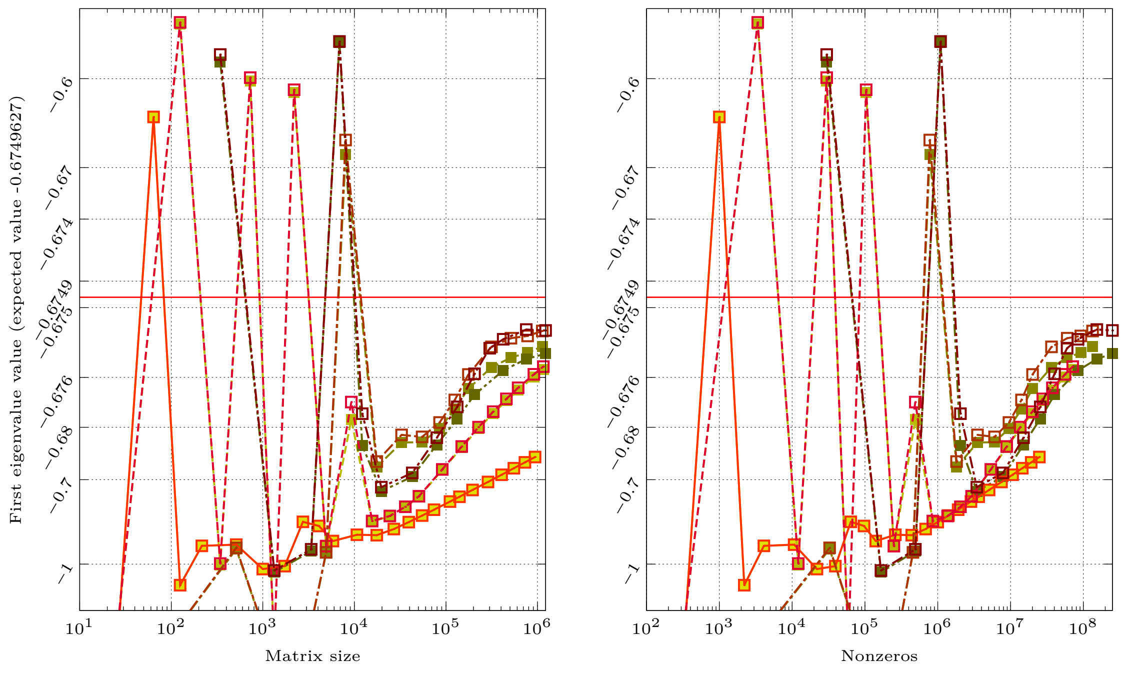


Figure 3.22: The first eigenvalue of the Nitrogen atom with regard to number of DoFs (left) and NNZs (right) — a comparison of the nodal Lagrange and hierarchical Lobatto bases. The solid red line denotes the expected value −0.6749627 obtained from a radial computation.



Lagrange bases are slightly better both in the accuracy (the error with regard to the reference value) and the convergence speed, therefore if p-adaptivity (different orders of shape functions in different finite elements) is not employed, a nodal basis seems to be a slightly better choice. However, this observation should be further verified on more complex problems, since the hierarchical bases are known to be better conditioned than Lagrange ones.[154, 155]

Comparison of FEM and IGA

The $\mathcal{C}^0$ IGA bases and their Lagrange counterparts (see Figs. 3.23 and 3.24) yield nearly the same results: the FEM results are almost obscured in the graphs. The IGA $\mathcal{C}^0$ bases are in fact Bernstein polynomial bases (see Section 3.3.1). Both the Bernstein and Lagrange bases are tensor products of polynomials (see section 3.2.2), therefore, they have the same pattern of structural nonzeros, which may explain the nearly identical results.

A richer picture offers the comparison of FEM and IGA bases with respect to the degree of continuity of their derivatives: the more continuous bases offer faster and more monotone convergence and especially more exact results than the $\mathcal{C}^0$ computations. The best bases seem to be again the ones of the third degree, with the maximally possible $\mathcal{C}^2$ continuity. The monotonicity and smoothness of the convergence curves of the IGA bases with continuous derivatives, compared to $\mathcal{C}^0$ cases, seem to be substantially better as well.

Comparison of uniform and refined meshes

The last comparison will demonstrate the advantages and disadvantages of the refined meshes. In Figs. 3.25 and 3.26, a smaller number of data points for the refined meshes is depicted. The presented results show the convergence with regard to the refinement level of a single uniform mesh, which limits the number of the possible meshes. More configurations could be generated by refining another “starting” mesh, but those would converge to a slightly different results (due to having boundary elements of different sizes and thus different boundary conditions in some sense).

We can see a slightly larger error of the eigenvalues for the refined cases. This is caused by a coarse mesh on the border of the volume domain, which results in a slightly worse approximation of wave functions tails. The selected algorithm for generating finer meshes refines only the already well-described center of the domain, so the results begin to stagnate earlier than in uniform mesh cases.

The second-degree polynomials on the refined meshes perform substantially better than the third-degree polynomials for a relatively small price of a slightly less precise eigenvalues. The second-degree polynomials on the small elements near the atomic center are able to describe the wave function precisely and there is a substantial saving of degrees of freedom in the elements on the border, which cover only smooth and easily describable tails of wave functions.

However, the ratio of “border” and ”inner’ elements is smaller for large structures, and the chemical bonds can stretch the wave functions more away from atomic centers in such cases. Thus, the second-order bases that in this case yield good

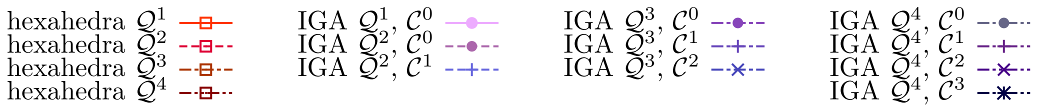


Figure 3.23: The error of a Nitrogen atom charge density with regard to number of DoFs (left) and NNZs (right) — comparison of IGA and FEM.

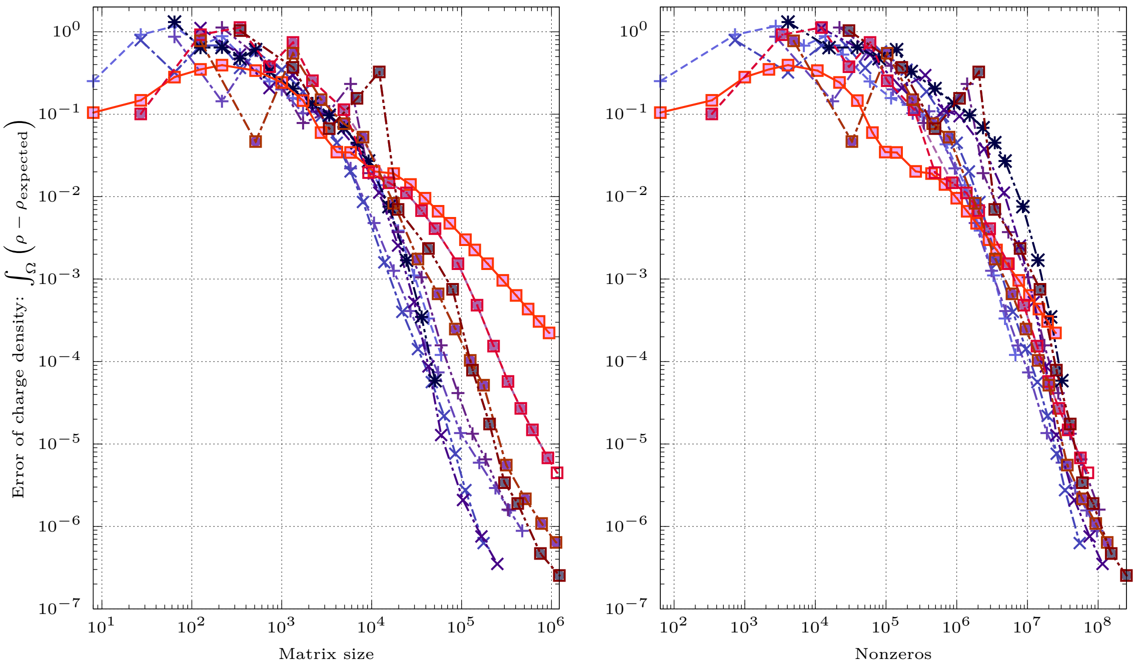
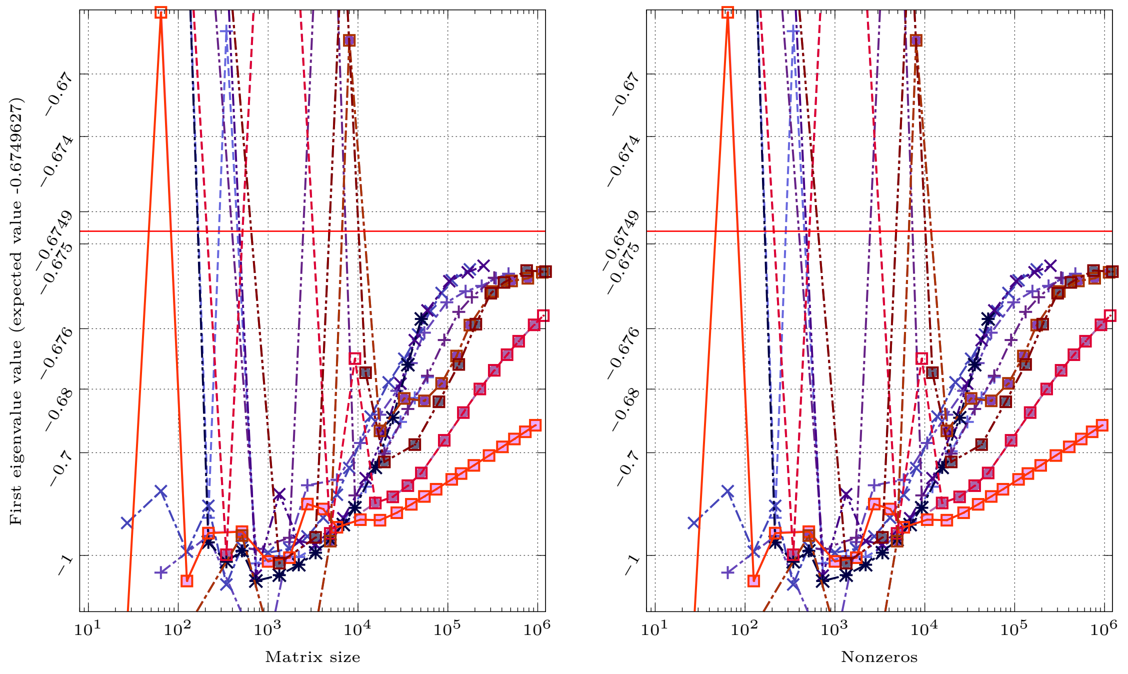


Figure 3.24: The first eigenvalue of a Nitrogen atom with regard to number of DoFs (left) and NNZs (right) — a comparison of IGA and FEM. Red line denotes the expected value −0.6749627 obtained from a radial computation.



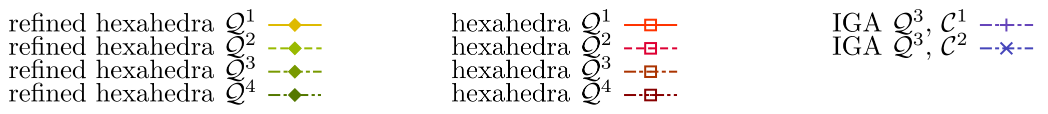


Figure 3.25: The error of a Nitrogen atom charge density with regard to number of DoFs (left) and NNZs (right) — a comparison of uniform and refined meshes.

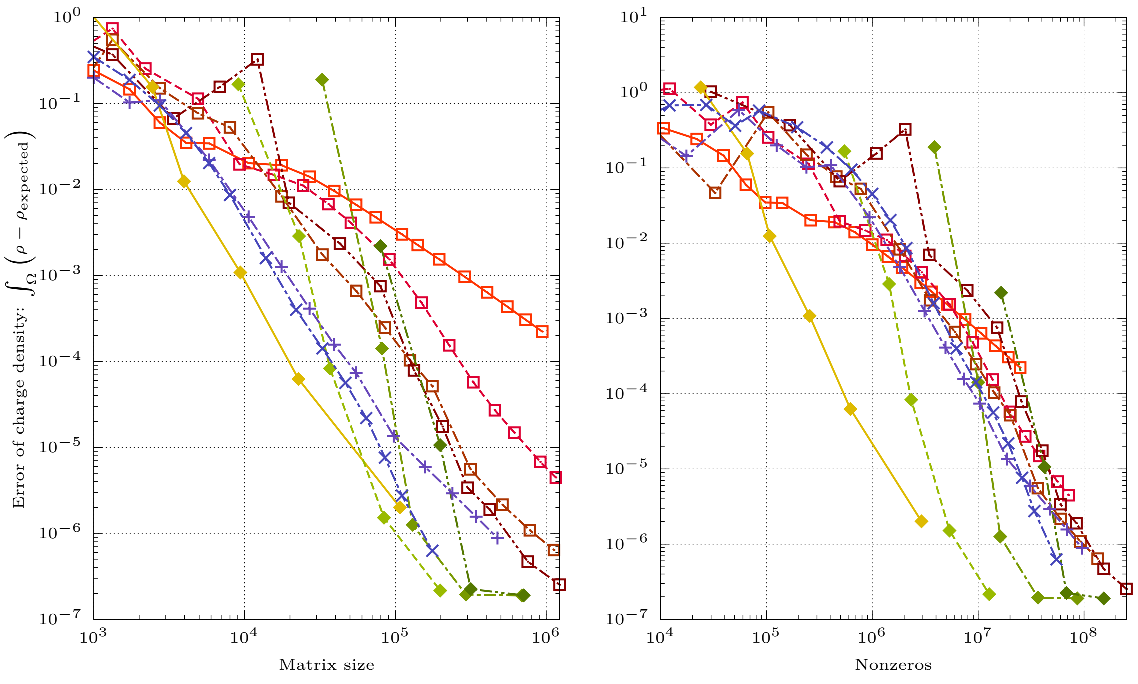
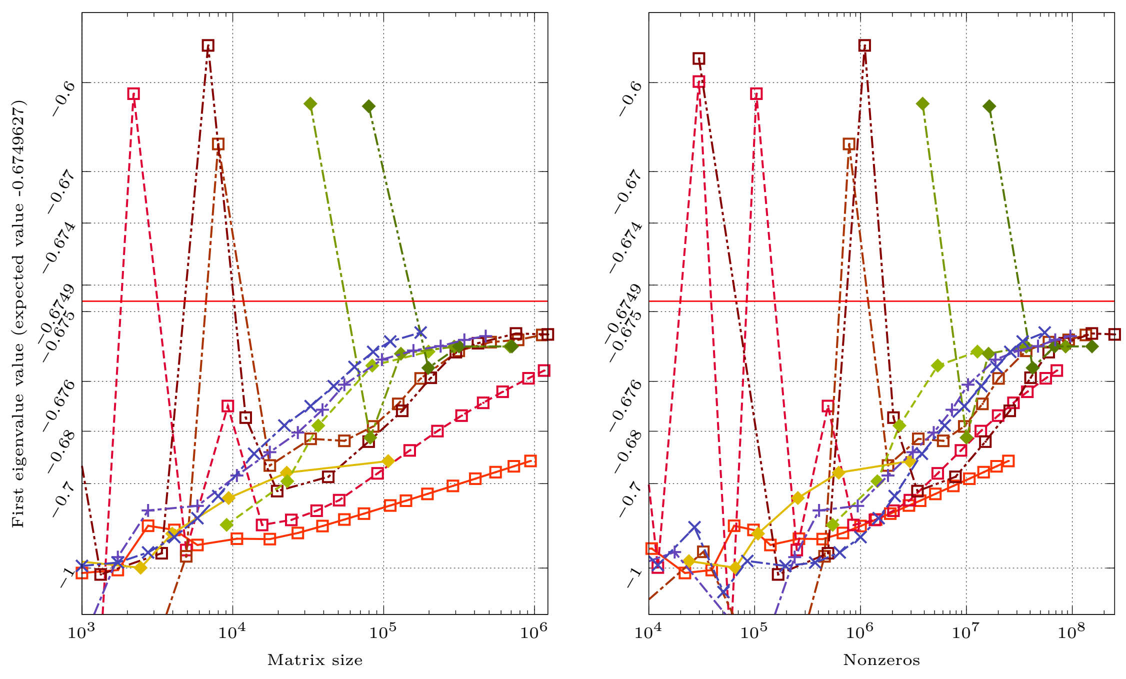


Figure 3.26: The first eigenvalue of a Nitrogen atom with regard to number of DoFs (left) and NNZs (right) — a comparison of uniform and refined meshes. Red line denotes the expected value −0.6749627 obtained from a radial computation.



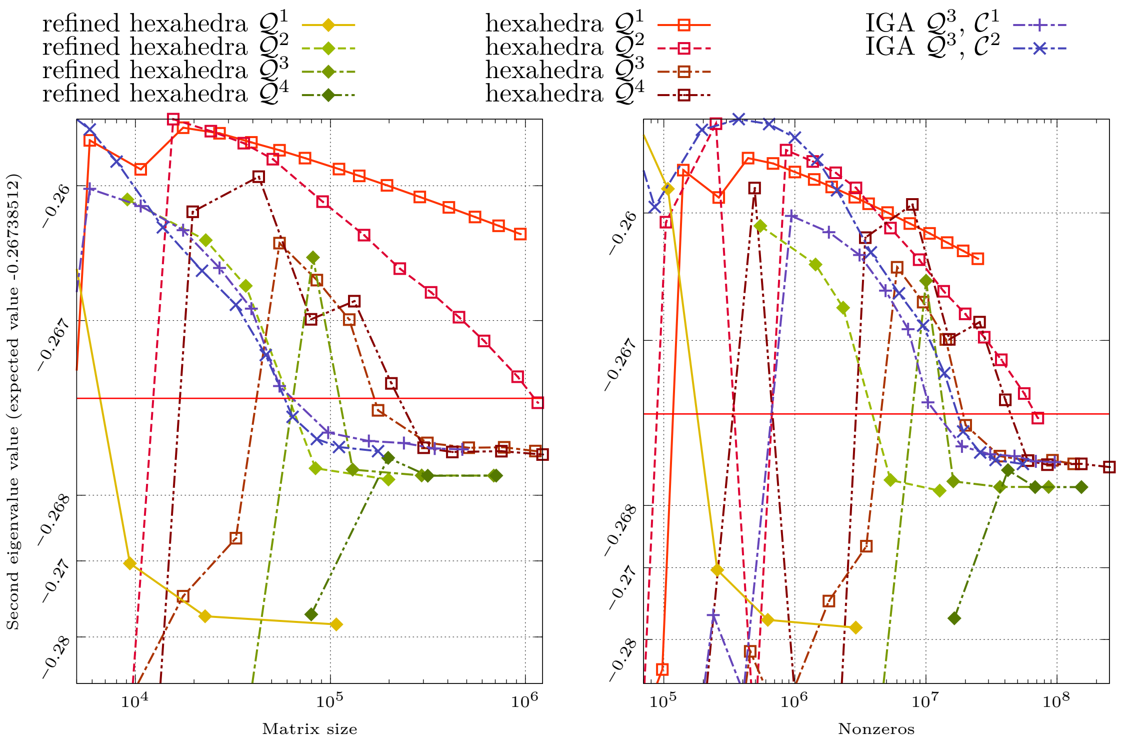


Figure 3.27: The second eigenvalue of a Nitrogen atom with regard to number of DoFs (left) and NNZs (right) — a comparison of uniform and refined meshes. Red line denotes the expected value −0.26738512 given from a radial computation.

results, could (and according to our experience sometimes do) yield not so good results in practice.

Since the third-degree polynomials offer a slightly better precision, the decision which degree to choose could be done with the property of the studied material on mind: the third degree seems to be the more robust choice, whereas the second-degree polynomials on he refined meshes could offer substantial speedup in some cases, but their accuracy should be verified.

From the very good performance of the second-degree polynomials on the refined meshes, it follows that the p-adaptivity — different orders of shape functions in different elements — might bring a substantial decrease of computational demands. It is one of the viable options for our further research, and would require the implementation of the p- or hp-adaptivity into the SfePy package.

Comparison of refined FEM and uniform IGA

Since the number of matrix nonzeros is a substantially better metric for the IGA than the number of DoF's (see Fig. 3.17), it can be concluded from the results in Fig. 3.25 and 3.26 that the refined quadratic FEM (both for the eigenvalues and the charge density) and the cubic FEM (for the charge density) outperform the IGA. The performance benefit for the refined meshes over the IGA was confirmed by the comparison of the second eigenvalues in Fig. 3.27.

The performance advantage of the both refined mesh cases suggests that the best results might be obtained using refined IGA computations. Unfortunately, such an option is not implemented in SfePy yet.

3.4.3 Conclusions of the analysis

As we have seen, for electronic structure calculations the hexahedral elements are a much better choice than the tetrahedral elements. The Bernstein and Lagrange polynomial bases yield the same results, whereas the hierarchical Lobatto basis is a bit inferior for higher degrees of polynomials, however, this claim should be further verified on more complex electronic structures since the theoretical results claim the opposite.

The best degree for the basis polynomials seems to be three for uniform meshes. For a refined hexahedral mesh, the third degree is a robust choice, but the second-order polynomials could offer a substantial speedup in some cases. The higher degree of continuity brings a major advantage for computations, however the lack of possibility to refine the IGA patches in our current implementation makes the refined FEM the best choice.

Therefore, in the current state of development, the best choice for a common computation seems to be to use the cubic hexahedral FEM elements. For large structures with high computational demands it can be a viable option to employ the second-degree polynomials, but the accuracy should be further verified. The IGA computation could serve as a useful tool for checking, whether the discontinuity of the derivatives does not spoil the computed results.

If the further development of SfePy includes incorporating refined elements in the IGA, the advantages of the IGA will probably prevail, thus, it is one of the promising improvements and aims of our further research. Likewise, an implementation of the full hp-adaptivity of the FEM meshes would be very desirable, as the current h-refinement implementation does not fully cover the different requirements for the precision in various regions of the computation domain.

An implementation of serendipity elements might be another good topic, as those elements have fewer degrees of freedom for the same polynomial degree than the Lagrange bases, especially for higher-degree bases. There is of course a drawback of the serendipity elements: they poorly treat the mesh deformation [105]. However, our adaptively refined hexahedral meshes have only the undeformed elements, so the serendipity elements would remain complete [105].

4. Solving the Kohn-Sham equations using the finite element method

Contents

This chapter is dedicated to the application of the finite element method for solving the Kohn-Sham equations (2.44). Unlike the Poisson equation, the Kohn-Sham equations contain singularities, which lead to numerical difficulties. We use a method of pseudopotentials to overcome the problem, however, the approach brings another problem into play: pseudopotentials harm the sparsity of the resulting matrices. To retain the sparsity of the resulting matrices, a special form of pseudopotentials (called *separable* or *Kleinman-Bylander*) is used, which allows to formulate the discretized equations as a *rank-k-update* eigenvalue problem.

The chapter begins with a derivation of the weak form of the Kohn-Sham equations (2.44) and their discretization using the FEM (using a common approach, the derivation follows [156]). The end of the first section is dedicated to the properties of these discretized equations and related numerical problems.

The second section describes the pseudopotentials, their basic properties, and their l-dependent and separable form. The separable form is used to transform dense matrices coming from the Kohn-Sham equations to a sparse rank-k-update problem. This section also includes a mathematical formulation of transforming pseudopotentials into a separable form and a brief description of a corresponding algorithm.

The last section of this chapter contains a brief study aimed at finding a suitable number of projectors used for creating a separable pseudopotential.

The beginning of this chapter is mainly an application of known facts to a given problem: the idea of eliminating singularities of all-electron calculations by replacing atomic cores by pseudopotentials [157] was originally published nearly a hundred years ago [158, 159] and can be found in many textbooks, e.g. [10]. The used separable form of pseudopotentials was originally proposed in [160]. Later parts of the chapter, primarily the pure algebraic formulation of separable pseudopotentials using B-orthogonality and their application to the finite element method are one of the key novelties of the newly developed method. Also, the study in the third chapter is an original work.

4.1 Discretization of the Kohn-Sham equations

In order to use the finite element method for solving the Kohn-Sham equations (2.44), the equations must be transformed into the weak form (see (3.8)) first and then discretized by the finite element method. We will follow a similar approach as in Section 3.1.

4.1.1 Weak form of the Kohn-Sham equations

Denoting

$$V = V_{\mathrm{H}} + V_{\mathrm{xc}} + V_{\mathrm{ext}}\,, \tag{4.1}$$

the Kohn-Shamm equations (2.44) can be expressed as

$$\left(-\frac{1}{2}\nabla^2 + V\right)\psi = e\psi\,. \tag{4.2}$$

To transform (4.2) to the weak form (see Eg. (3.8)), let us multiply the equation by a test function $\phi \in \mathcal{W}_0(\Omega)$ and integrate over the domain Ω:

$$\int_\Omega \left(-\frac{1}{2}\nabla^2\psi\phi^+ + V\psi\phi^+\right) = \int_\Omega e\psi\phi^+\,. \tag{4.3}$$

By applying the first Green identity ((3.15) on page 58) to the first term of (4.3) we obtain

$$\int_\Omega \left(\frac{1}{2}(\nabla\psi)\cdot(\nabla\phi^+) + (V\psi)\phi^+\right) - \int_\Omega \frac{1}{2}\nabla\cdot((\nabla\psi)\phi^+) = \int_\Omega e\psi\phi^+\,. \tag{4.4}$$

The application of the Gauss's theorem ((3.17) on page 59) results in

$$\int_\Omega \left(\frac{1}{2}(\nabla\psi)\cdot(\nabla\phi^+) \;+\; (V\psi)\phi^+\right) \;-\int_{\partial\Omega} \frac{1}{2}(\nabla\psi)\phi^+\,\mathbf{n} = \int_\Omega e\psi\phi^+\,, \tag{4.5}$$

where $\partial\Omega$ denotes the boundary of Ω. The test function is zero on the boundary, so the second term vanishes:

$$\int_\Omega \left(\frac{1}{2}(\nabla\psi)\cdot(\nabla\phi^+) + (V\psi)\phi^+\right) = e\int_\Omega \psi\phi^+\,. \tag{4.6}$$

4.1.2 Discretization of the Kohn-Sham equations by FEM

Having the zero boundary finite element space $\mathcal{W}_{h0}(\Omega)$ (see Definition 3.11 on page 59), the solution ψ and the test function ϕ from $\mathcal{W}_0(\Omega)$ in (4.6) can be approximated using a finite element basis φ_i of $\mathcal{W}_{h0}(\Omega)$ with the dimension $n = \dim \mathcal{W}_{h0}(\Omega)$ as

$$\psi \doteq \sum_i u_i \varphi_i \,, \qquad \phi \doteq \sum_j v_j \varphi_j \,, \qquad i, j \in 1, \ldots, n \,. \tag{4.7}$$

Substituting the approximations (4.7) into the weak Kohn-Sham equations (4.6) yields

$$\begin{aligned}\int_\Omega \left(\frac{1}{2} \left(\nabla \sum u_i \varphi_i \right) \cdot \left(\nabla \sum v_j \varphi_j^+ \right) + \left(V \sum u_i \varphi_i \right) \sum v_j \varphi_j^+ \right) = \\ e \int_\Omega \sum u_i \varphi_i \sum v_j \varphi_j^+ \,, \end{aligned} \tag{4.8}$$

or in the scalar product notation, denoting $\int_\Omega \varsigma \varphi^+ = (\varsigma \cdot \varphi)$,

$$\begin{aligned}\frac{1}{2} \Big(\nabla \sum u_i \varphi_i \cdot \nabla \sum v_j \varphi_j \Big) + \Big(V \sum u_i \varphi_i \cdot \sum v_j \varphi_j \Big) = \\ e \Big(\sum u_i \varphi_i \cdot \sum v_j \varphi_j \Big) \,. \end{aligned} \tag{4.9}$$

Satisfying (4.9) for all $\phi \in \mathcal{W}_{h0}(\Omega)$ is equivalent to satisfying it for all φ_i from the basis of $\mathcal{W}_{h0}(\Omega)$. This yields n linear equations for $j = 1 \ldots n$

$$\frac{1}{2} \Big(\nabla \sum u_i \varphi_i \cdot \nabla \varphi_j \Big) + \Big(V \sum u_i \varphi_i \cdot \varphi_j \Big) = e \Big(\sum u_i \varphi_i \cdot \varphi_j \Big) \,. \tag{4.10}$$

This can be rewritten, due to the linearity of the used operators, as

$$\sum u_i \left(\frac{1}{2} (\nabla \varphi_i \cdot \nabla \varphi_j) + (V \varphi_i \cdot \varphi_j) \right) = e \sum u_i (\varphi_i \cdot \varphi_j) \,. \tag{4.11}$$

Such system of equations can be written in the matrix form

$$L\mathbf{u} + V\mathbf{u} = eM\mathbf{u} \,, \tag{4.12}$$

where the particular matrices are

$$L_{ij} = \int_\Omega \frac{1}{2} (\nabla \varphi_i)(\nabla \varphi_j^+) \,, \tag{4.13}$$

$$V_{ij} = \int_\Omega (V \varphi_i) \varphi_j^+ \,, \tag{4.14}$$

$$M_{ij} = \int_\Omega \varphi_i \varphi_j^+ \,. \tag{4.15}$$

The discretization of the Kohn-Sham equations thus leads to the *generalized eigenvalue problem*.

4.1.3 Singularity of Coulombic potentials

The FE formulation derived in the previous section has a fundamental issue: the eigenvalue problem (4.12) can be solved, but it does not yield acceptable results. Let us first identify the issue.

The problematic part of the Kohn-Sham equations lies in the, on the first sight very simple, Coulombic potential V_{ext}. The potential consists of the sum of the electrostatic potentials of the atomic nuclei (recall (2.16))

$$V_{\text{ext}}(\vec{x}) = V_{\text{ion}}(\vec{x}) = \sum_{j=1\ldots\text{number of atoms}} -\frac{z_j}{\|\vec{x}-\vec{c_j}\|} = \sum V^j_{\text{ion}}(\vec{x})\,, \tag{4.16}$$

and therefore it has a singularity at the place of each atomic nucleus. Such a singularity is very undesirable for many reasons:

- ▷ the Gauss integration scheme is precise for polynomials — see Section 3.2.2. But the integration of a rational function is neither precise nor convergent in the scheme. Thus, the result of the numerical integration of such a function is very dependent on the "random" (with regard to the atomic center) position of integration points. When doing a geometry optimization with a singular potential, the computed energy of the system would vary not due to physical laws, but due to misuse of the numerical integration.

- ▷ Any approximation of the potential singularity requires a very rich basis and consequently a high number of degrees of freedom, which greatly increases the computational demands.

- ▷ The potential singularity attracts electrons. As a consequence, some electrons are strictly localized near the singularity and thus they have (according to the Heisenberg uncertainty principle, see e.g. [62]) very high energies. Such electrons (called *core electrons*) are so firmly tied to the atomic nucleus, that the effect of their surrounding on them is nearly negligible — they do not create bonds with other electrons.

  On the other hand *valence electrons*, which create bonds with other atoms, do not have such a strict localization and their energies are several orders of magnitude smaller. However, these energies are the quantities that define the properties of a calculated material: the difference of the energy of the bonded and non-bonded electron determines the strength of the bond and the sign of the difference determines the attractivity or repulsivity of the bond.

  To obtain the most energy efficient configuration of a sample, the configuration with the lowest total energy (given by (2.37)) has to be found. However, this energy is predominantly determined by the (nearly unchanging) energy of the core electrons. Thus, we have to subtract two large numbers to obtain the differences in the small valence energies[1]. Such an approach has unavoidable numerical problems: first, computers use floating point numbers with a finite

[1]The comparing of total energies is only a demonstration of the problem. In fact, the lowest energy state can be more easily determined using forces, however, the problem of large energies is principal, it spoils any similar approach.

number of decimal digits, and second, even the smallest numerical error (which is unavoidable) in the energies of core electrons totally overshadows the differences of the valence energies.

From the above discussion it clearly follows that the problem of potential singularities needs to be treated to make the Kohn-Sham equations solvable in practice. Several approaches have been developed to overcome this problem, e.g. the *frozen core* approach (core electrons do not change their wave functions nor energies during the computation) or adding special basis functions that describe well the solutions around the singularity. Pseudopotentials are one of the most successful methods for removing the singularities: their use eliminates core electrons completely from the computation, and thus not only it solves all the above mentioned issues, but it also reduces the number of the degrees of freedom in the computation. [158, 159, 15].

4.2 Pseudopotentials

The computationally efficient discretization of the Kohn-Sham equations used by FENNEC is based on *pseudopotentials*, which have been first constructed by Fermi [158] and Hellman [159]. The basic properties of pseudopotentials will be summarized in this section.

The following description of pseudopotentials does not strictly follow the process of their construction (which can be found e.g. in [15, 161]; th proof of the correctness of the approach can be found in [162]), but it only emphasizes the main idea.

Pseudopotentials are constructed using *screened all-electron potentials*. The term *all-electron* means a potential that includes the all, i.e. both core and valence, electrons of an atom, while *screened* means that the potential includes a Hartree and an exchange-correlation potential of all electrons in the system.

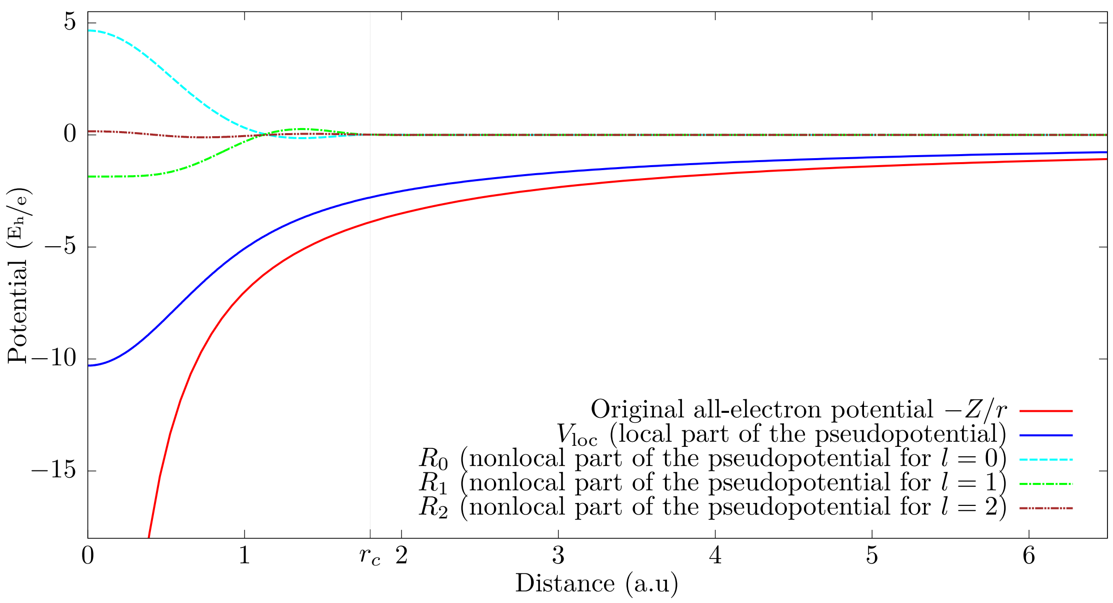


Figure 4.1: Example of a nitrogen pseudopotential with $r_c = 1.8$a.u.

However, the resulting pseudopotential needs to be *unscreened*, i.e. with the Hartree and the exchange-correlation potential of valence electrons subtracted. These potentials (their sum is called *unscreen potential* V_{unscreen}) in the computed system depend on the actual charge density and are included in the Kohn-Sham equations separately. Thus, this section will be focused on the resulting unscreened pseudopotential and its properties as those are important for the practical solvability of the resulting matrix problem.

4.2.1 Fundamental properties of pseudopotentials

According to the Born-Oppenheimer approximation, we can treat atomic nuclei and electrons separately. Since core electrons have so high energies (due to atomic nuclei potentials) that they are nearly not influenced by valence electrons, we can even separate (or decouple, see equation (2.18) on page 25) core and valence electrons. Having two objects with an opposite charge in one place — atomic nuclei and core electrons — both of them separated from valence electrons, a natural idea is to merge them to a single object, called *atomic core.*

The effect of an atomic core on valence electrons can be described by an (unscreened) potential V_{core}. Such a potential consists of an electrostatic potential of atomic nuclei and electrostatic (Hartree) and exchange-correlation potentials of core electrons.

The core electrons are strictly localized near the atomic nuclei and sum of their charges can be considered as spherically symmetric.[2] Therefore, the core potential V_{core} is nearly equal to the classical Coulomb potential in distances greater than a given distance r_0 from the atomic center (according to conventions, atom nuclei have a negative charge whereas electrons have a positive one)

$$V_{\text{core}} \approx \frac{Z'}{r} \qquad \forall r > r_0\,, \qquad \text{where } Z' = Z_{\text{nuclei}} + \textit{number of core electrons.} \tag{4.17}$$

The radius r_0 is in fact from a theoretical point of view infinite, since there is some nonzero probability, that even a core electron could be found at any distance from its atom nucleus. Therefore, the equality (4.17) does not hold exactly. However, the difference from the true Coulomb potential is negligible already in a small distance from the atomic center, much smaller than the range of valence electrons is.

Treating the atomic core as a single rigid object (this approach is called *frozen core approximation*) resolves the problem with high energies of core electrons because the Kohn-Sham equations with frozen atomic cores describe only the valence electrons. However, the problem with the potential singularity in the atomic center remains. Therefore the pseudopotential approach has been developed: a potential of each atomic core is replaced by an artificial potential V_{ps} having no

[2]It might not be perfectly symmetric in some cases, but the violation of the symmetry is mostly negligible. If it is not, the electron with the symmetry violated should be included among valence electrons. Moreover, the symmetry supposes that the division of the electrons into core and valence ones does not separate electrons from the same electron shell belonging to the same l-space (see Def. 4.7).

singularity, that is called *pseudopotential* and emulates the original core potential, i.e. the energies of wave functions are retained, as well as some other important properties of the original potential (e.g. scattering properties [162]).

Solutions of the Kohn-Sham equations with pseudopotentials are called *pseudo-wave functions*, and an atomic core with a pseudopotential instead of a potential is called *pseudocore*. However, the prefix pseudo is often omitted, if there is confusion is not likely.

The pseudopotential should satisfy the following requirements[161]:

1. A pseudocore should have the same electrostatic charge as the (original) atomic core. From the Poisson equation for the atomic core (see (5.14)) it follows that from some distance, the *cut-off radius* r_c, from the atomic center a pseudopotential has to be equal to the corresponding core potential.

$$V_{\text{ps}}(r) = V_{\text{core}}(r) \approx \frac{Z'}{r} \qquad \forall r > r_c\,. \tag{4.18}$$

   Within the radius r_c the $V_{\text{ps}}(r)$ and $V_{\text{core}}(r)$ differ. Note that the equality with the Coulomb potential does not hold exactly: as it has been stated, even the atomic core potential is not a true Coulomb potential. Moreover, the radius r_c can be shorter than the distance r_0, up to which a non-negligible charge of the core electrons reaches. Such an overlap of charge density of core electrons is called *core tail*.

2. On r_c, a pseudopotential must be smoothly connected to the potential of the atomic core. To achieve this, the value and derivatives of the *screened pseudopotential* must equal those of the screened potential of the atomic core on r_c. Pseudopotentials used in FENNEC are constructed in such a way that the first and the second derivatives agree.

3. The pseudo-wave functions should "emulate" the (original) wave functions. Therefore the values and the derivatives[3] (in our case the first and the second) of pseudo-wave functions and the corresponding wave functions should match on r_c.

4. The charge of each pseudo-wave function inside r_c should match the charge of its corresponding wave function:

$$\int_0^{r_c} \psi^+_{\text{pseudo}} \psi_{\text{pseudo}} = \int_0^{r_c} \psi^+ \psi\,. \tag{4.19}$$

   Pseudopotentials with this property are called *norm conserving*. Although not all kinds of pseudopotentials are norm conserving, the property is highly desirable (otherwise we would have to make not so easy corrections accounting for the missing or excessing charge inside r_c).

The assumption that core electrons do not interact with the surrounding environment is nearly exact, but it is not completely true: a reaction of core

[3]Commonly logarithmic derivatives are used, as they are norm-independent.

electrons can be observed in some cases (e.g. core-level shifts can occur [163]), and it cannot be generally neglected. Therefore the newly developed method uses the environment reflecting pseudopotentials [164, 20] that are capable of including effects of valence electrons on atomic cores.

Properties of pseudopotentials

Since a pseudopotential combines an atomic nucleus and its core electrons together, the solution of the Kohn-Sham equations with pseudopotentials contains only the valence electrons of the system. This fact substantially lowers the number of requested eigenvalues and eigenvectors and so substantially reduces the computational demands of used iterative solvers.

Smoothing of the wave functions is another desirable effect of pseudopotentials. Because the core electrons disappear from the solution and so the valence electrons do not have to be orthogonal to them (see the Pauli exclusion principle (2.12) on page 24), the wave functions have fewer oscillations in the atomic core region. For example, the radial component of the $2s$ valence electron wave functions of nitrogen has a single node (an intersection with zero). The same wave function computed using pseudopotentials does not have to be orthogonal to the $1s$ core electron state and so it has no nodes at all. The more oscillation the function has, the steeper it is and so it needs a richer basis to be described accurately. Therefore, the use of pseudopotentials further reduces the computational demands.

There are many criteria on how to classify pseudopotentials, three the most important being:

transferability Each potential is constructed for wave functions of some energy, for which the pseudopotential acts exactly as the all-electron potential. The more the energy of a wave function differs, the greater is a difference between the effect of the all-electron potential and the pseudopotential.

Transferability of a pseudopotential denotes the energy range, for which the pseudopotential is acceptable for computing: the more transferable the potential is, the wider the range of energies is for which the pseudopotential is acceptable.

softness A pseudopotential is constructed in order to eliminate the singularity, to reduce the steepness of the potential. The softer (the less steep) the potential is, the fewer basis function are needed for its description. Unfortunately, since a softer pseudopotential is less similar to the original potential, it is usually less transferable.

radius r_c The cut-off radius r_c, where the pseudopotential starts to differ from the all-electron potential, is significant for the properties of the pseudopotential. Generally, the smaller r_c, the more similar the pseudopotential is to its corresponding all-electron potential: a pseudopotential with a smaller r_c is usually deeper, but it has better transferability. A good choice of r_c depends on the type of atom and the solved problem.

4.2.2 L-dependent pseudopotentials

The introduction of pseudopotentials cures all the problems mentioned earlier, but it causes another problem: a pseudopotential that satisfies the previous requirements cannot be constructed as local (see Definition 2.1 on page 21). This claim has not been proven, however, all attempts to make local pseudopotentials have been unsuccessful — there must be a nonlocal effect of the potential inside the r_c radius. Since only local (pseudo)potentials must be used to retain sparsity of the matrices arising from finite element discretization (see Theorem 3.5 on page 83), the pseudopotentials have to be reformulated so that the sparsity is preserved.

Below we introduce a common class of pseudopotentials called *l-dependent pseudopotentials*. The nonlocality of l-dependent pseudopotentials lies in the projections into l-subspaces. Details of constructing l-dependent pseudopotentials can be found in [165].

The spherical harmonic functions and l-spaces

It is natural to use spherical coordinates for describing spherically symmetric potentials of atomic cores. The spherical coordinates allow us to express $\mathbb{R}^3$ as a direct product of angular and radial spaces.[4]

Definition 4.1 (Radial space). *The* radial space $\mathbb{R}_0^+$ *is a subspace of* $\mathbb{R}$ *defined by* $\mathbb{R}_0^+ = \{x \in \mathbb{R} : x \geq 0\}$.

Definition 4.2 (Angular space). *The* angular space *is a space on the surface of a sphere:* $(-\pi/2, \pi/2)$ *(elevation θ)* $\times [0, 2\pi)$ *(azimuth φ)* $+\{[-\pi/2, 0], [\pi/2, 0]\}$ *(poles).*

The projections into l-spaces can be defined in the following way (the following definitions can be found e.g. in [166]).

Definition 4.3 (Legendre polynomials). The Legendre polynomials P_n *are solutions of the* Legendre equation

$$\frac{d}{dx}\left[(1-x^2)\frac{d}{dx}P_n(x)\right] + n(n+1)P_n(x) = 0. \tag{4.20}$$

Definition 4.4 (Legendre associated polynomials). *The* Legendre associated polynomials *are solutions of the* general Legendre equation

$$(1-x^2)\frac{d^2}{dx^2}P_l^m(x) - 2x\frac{d}{dx}P_l^m(x) + \left[l(l+1) - \frac{m^2}{1-x^2}\right]P_l^m(x) = 0 \tag{4.21}$$

and can be expressed as

$$P_l^m(x) = (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}\left(P_l(x)\right). \tag{4.22}$$

[4]To be fully correct, the ambiguity at $r = 0$ should be solved.

[5]source: `https://commons.wikimedia.org/wiki/File:Spherical_Harmonics.png`

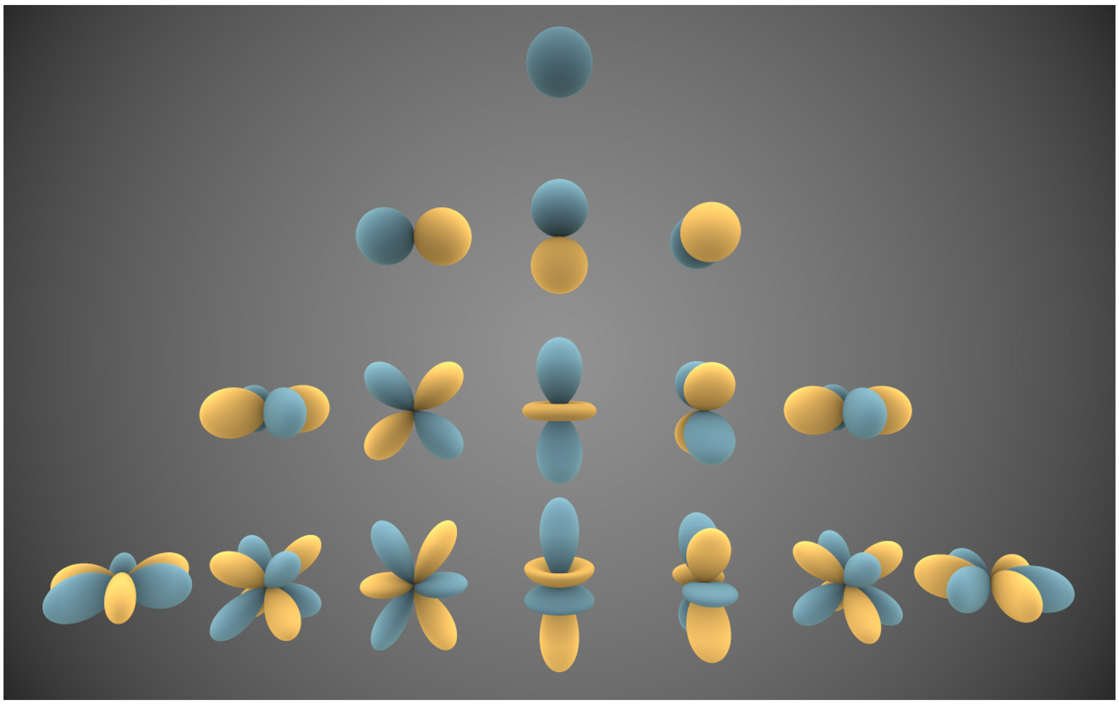


Figure 4.2: The spherical harmonics functions[5]

Definition 4.5 (Spherical harmonic functions)**.** *The* spherical harmonic functions *or* spherical harmonics *are the angular portion of the solution to Laplace's equation in spherical coordinates and can be found as eigenfunctions of the eigenvalue problem*

$$r^2\nabla^2 Y_{l,m} = -l(l+1)Y_{l,m}\,, \tag{4.23}$$

where r is the radius of the sphere. The spherical harmonic functions can be expressed as

$$Y_{l,m}(\theta,\varphi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(\cos\theta)e^{im\varphi}\,. \tag{4.24}$$

Note that the eigenvalues l of the spherical Laplace equation are nonnegative integers and the eigenvectors are the angular solutions of the stationary Schrödinger equation (2.10) in spherical potentials (see e.g. [167]). Moreover, they are orthogonal

$$\int Y_{l,m}^{+}Y_{l',m'} = \delta_{l,l'}\delta_{m,m'} \tag{4.25}$$

and form a full basis of the angular space. It is not difficult to prove (see e.g. [62]) the following theorem using the above-mentioned properties of the spherical harmonic functions:

Theorem 4.1 (Radial-angular decomposition theorem)**.** *Each solution of the stationary Schrödinger equation in a spherical symmetric potential can be written as a product of a spherical harmonics and a radial function*

$$\psi_{l,m,n}(r,\theta,\varphi) = Y_{l,m}(\theta,\varphi)\psi_{l,m,n}^{\text{Rad}}(r)\,. \tag{4.26}$$

Solutions described by the theorem are the well known *atomic orbitals* and the parameters l, m, n are called *quantum numbers*.

Sometimes it is useful to use the spherical harmonics in their real form:

Definition 4.6 (Real spherical harmonics)**.** *The* real spherical harmonics *(or* real spherical harmonic functions*)* $Y_{l,m}$ *are the functions defined by*

$$Y_{l,m} = \begin{cases} \frac{i}{\sqrt{2}} \left(Y_{l,m} - (-1)^m Y_{l,-m}\right) & \textit{if } m < 0\,, \\ Y_{l,0} & \textit{if } m = 0\,, \\ \frac{1}{\sqrt{2}} \left(Y_{l,-m} + (-1)^m Y_{l,m}\right) & \textit{if } m > 0. \end{cases} \tag{4.27}$$

The real spherical harmonics have the same properties as the complex ones, and moreover they form a basis of the space of real functions on the angular space.

Definition 4.7 (Angular l-spaces)**.** *Angular l-spaces $\mathcal{L}_l$ are the subspaces of the space of real functions on the angular space given by the orthogonal decomposition using the real spherical harmonic functions:*

$$\mathcal{L}_l = \operatorname{span}\{Y_{l,m} : m \in \{-l \ldots l\}\}\,. \tag{4.28}$$

Definition 4.7 gives us a hint how to build the projectors to a given l-space[6]:

Definition 4.8 (L-space projection)**.** *For a given $l \in \mathbb{N}$, the l-space projection is given by the following projector P_l:*

$$P_l(\psi(r, \theta, \varphi)) = \sum_{m=-l}^{l} \left(Y_{l,m} \times \int_{\theta,\varphi} Y_{l,m}^{+} \psi \right)\,. \tag{4.29}$$

The use of the times symbol ($\times$) should emphasize the fact, that functions of different coordinates are multiplied in (4.29): the spherical harmonics are functions of angles, while the result of the integral is a radial function.

L-dependent pseudopotentials

According to the radial-angular decomposition theorem 4.1, each wave function of an isolated atom is strictly localized in its l-space and so it can be subscripted by l. The value l is related to the angular momentum of the wave function [62]: the number of nodes of a spherical harmonic function is given by its l (see Fig. 4.2). Therefore, the higher is the l, the higher the energy of the first wave function (by energy) from this l-space is.

In more complex structures than just an isolated atom, the spherical symmetry of the charge density is broken. However, still the potential of an atomic nucleus (or a pseudopotential) causes that the angular parts of wave functions near an atomic core lie in a few l-spaces with the lowest l to minimize their energy. The

[6]Here a confusion can arise, as P_n is commonly used both for the projectors and for the Legendre polynomials. As the Legendre polynomials will not appear in the following discussion, the established notation will be used.

next important point is the observation, that the nonlocality is only needed to emulate the singularity of a potential. Therefore it is sufficient to limit the nonlocal effect of the pseudopotential only for the (spatial) part of the radial space, where the pseudopotential differs from its original potential.

This observation leads naturally to splitting the potential to a long-range local and a short-range nonlocal part

$$V_{\mathrm{ps}} = V_{\mathrm{loc}} + V_{\mathrm{nl}} \qquad \text{where } V_{\mathrm{loc}} \text{ is local and } \forall r > r_c : V_{\mathrm{nl}}(r) = 0\,. \tag{4.30}$$

The idea of an *l-dependent pseudopotential* lies in expressing the nonlocal part of the pseudopotential V_{nl} as radial functions, applied to a given l-projection of the wave functions.

Definition 4.9 (L-dependent pseudopotential)**.** *The L-dependent pseudopotential is a potential, given by* (4.30)*, where its long-range part V_{loc} is local and its nonlocal part V_{nl} is given by a set of radial functions $\{R_l\}$:*

$$V_{\mathrm{nl}} = \sum_l V_l = \sum_l R_l P_l = \sum_{l,m} R_l Y_{l,m} Y^+_{l,m}\,, \tag{4.31}$$

where

$$\forall l, \forall x > r_c : R_l(x) = 0\,. \tag{4.32}$$

Such a decomposition into l-spaces is, in fact, the decomposition of the operator in the basis of spherical harmonics $Y_{l,m}$, whose radial components are the same for the same l. Generally, such a decomposition has an infinite dimension (and so the sum is generally infinite). Thus, this approach is possible (or better it is efficient) only due to the specific nature of atomic core potentials, that pulls in a majority of the charge into the lowest l-subspaces.

Maybe the decomposition is more clearly written in the bra-ket notation, as it expresses explicitly that the projection is done via the scalar product (in the angular space) of spherical harmonic functions with the function it is applied on:

$$V_{\mathrm{nl}} = \sum_{l,m} R_l \, |Y_{l,m}\rangle\langle Y_{l,m}| \ .$$

The operator $|Y_{l,m}\rangle\langle Y_{l,m}|$ forms a dense matrix, which also illuminates the nature of nonlocality of the l-dependent pseudopotentials.

Construction of l-dependent pseudopotentials

A brief and inexact summary of the pseudopotentials construction will be described here, just for the sake of completeness. For the exact description, please see the mentioned literature.

The construction of an l-dependent pseudopotential for a given atom consists of creating R_l potentials for each l-space included in the potential. R_l are usually created by optimizing coefficients of a basis (e.g. often some kind of Gauss functions are used), constrained so that the requirements listed in Section 4.2.1 are fulfilled. The used basis and the used optimization criteria (e.g. the depth of the pseudopotential and its maximal steepness) differ according to a given method. This optimization is performed in each l-subspace independently on the other l-subspaces.

After the radial functions R_l are obtained, their average can be used as the local part of the potential (or the local part can be constructed in a more sophisticated way to better approximate the higher, in the pseudopotential not included l-spaces). The last step is to unscreen the potential: subtract the Hartree and the exchange-correlation potentials (see (2.41) and (2.42) on page 30) from the local part of the pseudopotential.

4.2.3 Separable pseudopotentials

The decomposition to the l-spaces allows us to restrict the nonlocality of the operator but does not solve the problem with the matrix sparsity. Moreover — the evaluation of P_l projectors requires a spherical integration. Such a task is possible, but it is computationally demanding and has substantial drawbacks if a FEM basis is used for the discretization (see the results in Section 5.4.2), so we need a more efficient formulation of l-dependent pseudopotentials. This appears to be a *separable pseudopotential*, which will be described in the following text.

Separable pseudopotentials are a well-known form of pseudopotentials [160], and the decomposition of an operator using B-orthogonality has been probably described as well since it is a not so surprising idea. Yet the connection of these two concepts leading to a pure algebraic description of separable pseudopotentials, as well as some of the applications of the presented description, e.g. extending the projectors basis by energy derivatives for a better transferability of separable pseudopotentials, cannot be found in literature (however the presented ideas were surely behind the scene).

B-orthogonality

Let us first visit the field of linear algebra. We will generalize the concept of orthonormality for an “indefinite scalar product”. Since we already discretized the equations, we will formulate the approach in a finite-dimensional vectors space, however, it is possible to use the same approach in an infinite-dimensional space prior to the discretization.

Definition 4.10 (B-scalar product)**.** *Given a vector space* $\mathbb{V}$ *with a scalar product* $(* \cdot *)$ *and a Hermitian operator* B*. Then* B-scalar product *is defined as*

$$(\mathbf{u} \cdot \mathbf{v})_B = (\mathbf{u} \cdot B\mathbf{v})\,. \tag{4.33}$$

It can be also easily proven that the B-scalar product for a positive definite matrix B fulfills the definition of a scalar product. For an indefinite matrix, the B-scalar product obeys the definition of the scalar product only on its “positive definite subspace”. Hence, we define a *Hermitian form* — a relaxed version of the scalar product, that is applicable even to indefinite matrices.

Definition 4.11 (Hermitian form)**.** *Let* $\mathbb{T}$ *is a field and* $\mathbb{V}$ *is a vector space. Then the Hermitian form is the function* $\chi(\mathbb{V} \times \mathbb{V}) \to \mathbb{T}$*, that has the following properties:*

1) $\chi(\mathbf{x}+\mathbf{y},\mathbf{v}+\mathbf{w}) = \chi(\mathbf{x},\mathbf{v}) + \chi(\mathbf{x},\mathbf{w}) + \chi(\mathbf{y},\mathbf{v}) + \chi(\mathbf{y},\mathbf{w})$

2) $\chi(a\mathbf{x}, b\mathbf{y}) = a\bar{b}\chi(\mathbf{x},\mathbf{y})$

3) $\chi(\mathbf{x},\mathbf{y}) = \overline{\chi(\mathbf{y},\mathbf{x})}$

Note that the definition of the Hermitian form for the case of real numbers does not differ from the definition of the bilinear form.

Lemma 1 (B-scalar product)**.** *The B-scalar product as defined in Definition 4.10 is the Hermitian form.*

A proof of the Lemma is only a technical verification of the definition of the Hermitian form (Definition 4.11).

Definition 4.12 (B-orthonormality)**.** *Given a Hermitian operator B, vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *are B-*orthogonal*, if*

$$(\mathbf{u}\,\cdot\,\mathbf{v})_B = 0\,. \tag{4.34}$$

*Given a set of vectors V, they are B-*orthonormal *if*

$$V^+(BV) = I\,, \tag{4.35}$$

where I is the identity matrix.

This definition is valid only for positive definite matrices, so we again define a relaxed version of the same concept:

Definition 4.13 (B-pseudoorthonormality)**.** *Given a set of vectors V and a Hermitian operator B, vectors V are B-*pseudoorthonormal *if there exists a diagonal matrix D that contains only numbers $\{-1,0,1\}$ on its diagonal*[7] *such that*

$$V^+(BV) = D\,. \tag{4.36}$$

Definition 4.14 (Definitness matrix)**.** *The matrix D from Definition 4.13 is called the* definitness matrix *of B and V.*

The construction of a separable pseudopotential is based on the following theorem:

Theorem 4.2 (B-orthogonal decomposition)**.** *Given a linear Hermitian operator B on a vector space $\mathbb{V}$ with a B-pseudoorthonormal basis V. Then*

$$((BV)D(BV)^+)V = B(D^2V) \tag{4.37}$$

where D is the definitness matrix of B and V.

[7]The minus one is allowed for the case of real numbers. In the case of complex numbers, the factor -1 can be dissolved into the basis. Note that since the operator is Hermitian, its eigenvalues are real and thus such a decomposition is possible.

Proof.

$$(BVD(BV)^+)V = BVD(V^+B^+V) = BVD(V^+BV)^+ = BVDD^+ = BD^2V \tag{4.38}$$

□

Since D^2 is the identity except for the elements of the basis V which are B-orthogonal to themselves (that are in the kernel (or null space) of the operator B), this theorem gives us the recipe how to decompose an operator B into a basis V.

Provided that we are able to select the subspace with a sufficiently low dimension, where we need the operator to be described, we can choose a basis of the subspace, B-orthonormalize it and decompose the operator using the theorem.

In this manner, we obtain an approximation of the operator, that will be exact on the given subspace, while the decomposed operator will be equal to zero for functions orthogonal to the given subspace. The construction of separable pseudopotentials just follows this idea.

Generating separable pseudopotentials from l-dependent pseudopotentials

We have already stated that the potentials of atomic nuclei (and pseudopotentials too) are so deep, that their influence (nearly) overrides the other potentials inside the radius r_c around the atomic core. As a result, not only the spherical symmetry of the core charge density is enforced by the atomic core inside the radius, but even valence wave functions do not differ too much from the shapes of wave functions of an isolated atom in this region.

Moreover, due to the Pauli principle (see (2.13)), wave functions belonging to a single l-subspace are orthogonal and so they form a basis in the radial space. Since they are orthogonal, each radial wave function has a different number of nodes, and, same as for the angular space, the fewer nodes a function has, the smaller its energy is. If we denote n the number of nodes of a radial wave function, the energies of wave functions with equal sums $l + n$ are similar: this fact is the essence of the chemical orbital notation e.g. $2s$, $1p$, etc.

Therefore, we have a natural choice for the radial space, where we need to describe the nonlocal part of a pseudopotential for a given l: we can take the first few lowest-energy wave functions of the isolated atom, use them to create a V_l-pseudoorthonormal basis and then apply the B-orthogonal decomposition theorem 4.2. These steps together form Algorithm 4.1 for generating the *separable pseudopotential* for a given l.

Definition 4.15 (Projectors). *The radial functions $\nu_{l,i}$ generated by Algorithm 4.1 are called* projectors.

We can summarize the previous development into a short theorem:

Theorem 4.3 (Separable pseudopotential). *Let R_l be a nonlocal radial part of a pseudopotential for a given l and let ψ_i be the radial wave functions computed*

Algorithm 4.1: Generating a separable pseudopotential

```
1: procedure Separable pseudopotential($R_l, V_{\text{loc}} + V_{\text{unscreen}}$)
2:     $\psi_i$ = Solve the Khon-Sham equations for $(R_l + V_{\text{loc}} + V_{\text{unscreen}}))$
3:     $\overline{\nu_{l,i}}, d_i = R_l$-orthonormalize($\psi_i$)
4:     $\nu_{l,i} = R_l\overline{\nu_{l,i}}$
5:     return $\nu_{l,i}$, $d_i$
```

with that pseudopotential and l. Then there exists a separable pseudopotential $\nu_{l,i}$ generated by Algorithm 4.1, so that it holds

$$\forall\psi \in \operatorname{span}\left(\{\psi_i\}\right) : R_l\psi = \sum_i \nu_{l,i} d_i(\nu_{l,i} \cdot \psi)\,. \tag{4.39}$$

The simple consequence of the theorem is that the radial nonlocal part R_l of a given l-dependent pseudopotential for a given l can be approximated as

$$R_l = \sum_i d_i \nu_{l,i} \nu_{l,i}^+\,, \tag{4.40}$$

or in the bra-ket notation

$$R_l = \sum_i d_i \left|\nu_{l,i}\middle\rangle\middle\langle\nu_{l,i}\right| \tag{4.41}$$

and the approximation is exact for the wave functions, from which the projectors $\nu_{l,i}$ have been constructed (whereas the approximation is effectively zero for all other (orthogonal) wave functions).

Application of separable pseudopotentials

Unlike the l-dependent pseudopotentials, where the decomposition was done in order to restrict the nonlocality to the angular part of the pseudopotential, the decomposition to the separable potential has a rather different purpose. Notice that there is a certain symmetry in the expression of the projectors P_l (see (4.31)) and so in the angular part of V_l:

$$V_l(\psi(r,\theta,\varphi)) = R_l(r)(P_l\psi)(r,\theta,\varphi) = R_l(r)\sum_{m=-l}^{l}\Big(Y_{l,m}\Big(Y_{l,m}^+ \cdot \psi\Big)\Big)(r,\theta,\varphi)\;, \tag{4.42}$$

or it might be more visible in the bra-ket notation:

$$|V_l\,|\psi\rangle = |R_l\rangle\,\left|Y_{l,m}\middle\rangle\middle\langle Y_{l,m}\right|\;. \tag{4.43}$$

Using the separable pseudopotential approach (4.40), a very similar symmetry appears in the radial part (4.41) as well. In the following expressions, $\times$ operator is used to emphasise the fact, that the multiplication results in a function with the domain given by a direct product of the domains of the coefficients of the product).

$$\left(\nu_{l,i} \times Y_{l,m}\right)(r,\theta,\varphi) = \nu_{l,i}(r) Y_{l,m}(\theta,\varphi) \tag{4.44}$$

Thus, the V_l operator can be written as:

$$\begin{aligned} V_l = \sum_i d_i \nu_{l,i} \nu_{l,i}^+ \times \sum_{m=-l}^{l} \left(Y_{l,m} Y_{l,m}^+\right) = \\ \sum_{i,m} d_i \left(\nu_{l,i} \times Y_{l,m}\right)\left(\nu_{l,i}^+ \times Y_{l,m}^+\right) = \sum_{i,m} d_i \left(\nu_{l,i} \times Y_{l,m}\right)\left(\nu_{l,i} \times Y_{l,m}\right)^+ \end{aligned} \tag{4.45}$$

Using the decomposition (4.45), the nonlocal — dense — part of the matrix of the discretized Kohn-Sham equations (4.14) can be expressed as

$$\begin{aligned} \left(\varphi_j \cdot V_{\mathrm{nl}} \varphi_k\right) = \left(\varphi_j \cdot \sum_l \sum_{i,m} d_i \left(\nu_{l,i} \times Y_{l,m}\right)\left(\nu_{l,i} \times Y_{l,m}\right)^+ \varphi_k\right) = \\ \sum_l \sum_{i,m} d_i \left(\varphi_j \cdot \left(\nu_{l,i} \times Y_{l,m}\right)\left(\nu_{l,i} \times Y_{l,m}\right)^+ \varphi_k\right) = \\ \sum_l \sum_{i,m} d_i \left(\varphi_j \cdot \left(\nu_{l,i} \times Y_{l,m}\right) \underset{\sim\sim\sim}{\left(\varphi_k \cdot \left(\nu_{l,i} \times Y_{l,m}\right)\right)}\right). \end{aligned} \tag{4.46}$$

The term underlined by a wavy line is a scalar number given by the inner product. Therefore, using the following property of inner product [168]

$$\left(\mathbf{u} \cdot a\mathbf{v}\right) = \overline{a}\left(\mathbf{u} \cdot \mathbf{v}\right), \tag{4.47}$$

where the overline denotes a complex conjugation, we can rephrase (4.46) as

$$\begin{aligned} \sum_l \sum_{i,m} d_i \left(\varphi_j \cdot \left(\nu_{l,i} \times Y_{l,m}\right)\left(\varphi_k \cdot \left(\nu_{l,i} \times Y_{l,m}\right)\right)\right) = \\ \sum_l \sum_{i,m} d_i \left(\varphi_j \cdot \left(\nu_{l,i} \times Y_{l,m}\right)\right)\overline{\left(\varphi_k \cdot \left(\nu_{l,i} \times Y_{l,m}\right)\right)}. \end{aligned} \tag{4.48}$$

The symmetry is clearly visible in the previous expression, thus, we can denote:

$$U_{i,j,l,m} = \left(\varphi_j \cdot \left(\nu_{l,i} \times Y_{l,m}\right)\right). \tag{4.49}$$

Now we can express the discretized Kohn-Sham equations (4.12) in a computationally efficient way as

$$L\mathbf{u} + V_{\mathrm{loc}}\mathbf{u} + UDU^+\mathbf{u} = eM\mathbf{u}, \tag{4.50}$$

where $U \in \mathbb{R}^{n \times k}$ and k is the total number of projectors from all atomic cores. Therefore, using separable pseudopotentials, we transform the dense generalized eigenvalue problem to the *sparse rank-k-update generalized eigenvalue problem.*

Definition 4.16 (Rank-k-update (generalized) eigenvalue problem). *A* rank-k-update problem *is an eigenvalue problem with dimension n of the form*

$$A\mathbf{u} + UDV\mathbf{u} = \lambda B\mathbf{u}\,, \tag{4.51}$$

where the matrix U has the shape $n \times k$, the matrix D is $k \times k$ and the matrix V is $k \times n$. For the generalized problem the matrix B is not the identity matrix.

The form of the rank-k-update problem is computationally efficient — we can even precompute $U \in \mathbb{R}^{m\times n}$ once at the start of the DFT cycle, since the pseudopotential does not depend on an actual charge density, and thus does not change during DFT iterations. Hence, we resolved the issue presented at the beginning of the section.

4.3 Finding a suitable size of the radial basis for generating separable pseudopotentials

In order to make the computation with separable pseudopotentials efficient, the dimension of the updates U (from (4.50)) must be kept as low as possible. Thus it is crucial to make a proper choice of the radial basis ψ_i for Algorithm 4.1. This section should answer the question, how to construct the projectors and how many of them are needed to describe a pseudopotential well.

Of course, the suitable number of projectors differs from case to case, so there is no general answer to the stated question. However, some guidelines for efficient use of FENNEC can be obtained from the data following below. Moreover, the methodology used in this section can be used for verifying that the number of projectors is suitable in a particular case. The following work is also a methodical preparation for a quality evaluation of the pseudopotentials generated using a newly developed code for pseudopotential construction, written by my colleague Robert Cimrman.

4.3.1 Test setup

To evaluate the suitable number of projectors, the charge density of given systems was computed with a high number of projectors to obtain the reference charge density. The adaptive hexahedral mesh (see Section 3.2.1) with cubic elements was used, with the edge length of the smallest elements equal to 0.5 a.u. Then (using the same initial charge density) the number of projectors was decreased progressively and the L^2 norms of the differences of the resulting charge densities from the reference one were being evaluated.

Since the state of a system is given by the charge density (see the First Hohenberg-Kohn theorem 2.2), a small error in the charge density is sufficient to have a computation precise “enough”. This claim (which is stated “intuitively” here, since its exact formulation would require sophisticated error analysis of the whole computation chain) is further confirmed by the work presented in the next chapter: see Fig. 5.7 on page 154 for the relationship between the error in the charge density and the error in the force.

All the used pseudopotentials were generated using LDAT routines for generating environment reflecting l-dependent pseudopotentials [20]. However, the pseudopotentials were not adapted for the charge density of the computed samples (see [20]) if it is not said otherwise: the purpose of the test is not to demonstrate the quality of fine-tuned pseudopotentials, but just to evaluate the effect of raising or lowering of the number of projectors in "real cases" where the ability of a pseudopotential fine-tuning is often limited (e.g. during geometry optimizations, where the energies of wave functions change with changes of the geometry).

The conversion from the l-dependent to separable form was done using routines included in FENNEC by Algorithm 4.1.

A number of systems were evaluated in this way: e.g. various dimers (e.g. nitric oxide and titanium nitride), carbon dioxide, and a few more complex molecules (dicyanocyanamide, tetrafluoromethane, fluorouracil). Stretched dimers have wave functions with different energies, therefore, the dimers were computed with various interatomic distances to detect a possible loss of transferability during the separable pseudopotentials generation.

The molecules were selected "randomly", taking into account the chemical elements included in the material samples, which we plan to compute in our further research. Some of them were picked with an assumption that a given phenomenon to be demonstrated could appear.

Choice of projectors

In the previous section (4.2.3) it has been explained, that a natural choice is to take the first few radial solutions of the Kohn-Sham equations (2.44) (for a given atomic l-pseudopotential) as the basis for the decomposition. In this section, we try to find an answer to the question, what number of projectors is commonly sufficient in the calculations.

Moreover, the following approach for the radial basis construction will be put into the test: we already mentioned the property of transferability of a pseudopotential. Since the separable form of the pseudopotential limits to a great extent the number of wave functions, that are "seen" by the pseudopotential (see the discussion below (4.41)), its transferability can be even lower than the transferability of the original pseudopotential. Especially for a single projector, the transferability is often very low.[169]

Therefore, it seems to be a viable approach to enrich the basis by a derivative of the atomic wave function with respect to its energy. Similar approaches have already been published, e.g. Chou suggests including more wave functions with different energies [170], but we are not aware of a publication using the energy derivatives.

4.3.2 Test results

The results of the performed numerical tests show some common patterns: therefore we will not present all the performed computations, but only the examples of the patterns observed during our tests.

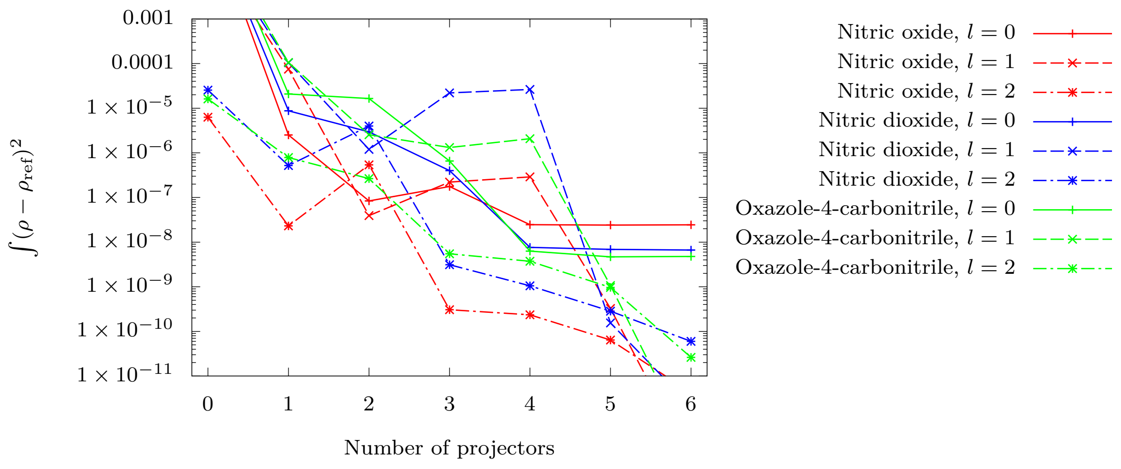


Figure 4.3: Analysis of the charge density convergence with regard to the number of projectors for the Nitrogen pseudopotential in various compounds.
111

Nitrogen pseudopotential and geometry of the molecule The first tested pseudopotential was for Nitrogen, which had appeared unproblematic in our previous practical computations (e.g. it provided the correct molecular distances, see Fig. 5.2 on page 148). We can see in Fig. 4.3, that already two projectors lead to sufficiently precise results.

The $l = 1$ part of the pseudopotential exhibits the highest sensitivity to the number of projectors, and moreover it does not converge monotonically. As it could be expected, the $l = 2$ part of the pseudopotential, whose eigenstates are not occupied, requires a smaller number of projectors to achieve the same precision.

Two similar molecules, NO and NO_2, were computed, to verify, whether the used mesh (a refined hexahedral mesh with regular hexahedrons, see Section 3.2.1) brings a "preferred direction bias” into the computation or not.

If that were the case, the results for those molecules would differ, since one of them is linear (has the angle 180° between the bonds) and the second is not. However, the observed differences were small, which indicates that the bias is not significant (at least in this particular case).

Titanium pseudopotential Titanium is a transition metal, and those are known to cause “computational difficulties”. This phenomenon — related probably to the fact that there is only a narrow energy gap between $3d$ ($l = 3$) and $4s$ ($l = 0$) electrons — was “proven” again by the following calculations of titanium dioxide and titanium nitride, in which titanium shows much larger sensitivity (in a computational sense) than both the elements from the previous case.

The first case in Fig. 4.4 for $l = 2$, denoted as titanium dioxide with the “bad” pseudopotential, is presented here to show results of an unsatisfactory pseudopotential. A very low transferability of the potential makes the computation unstable and the results were very different for any lower number of projectors and any used order of numerical quadrature.

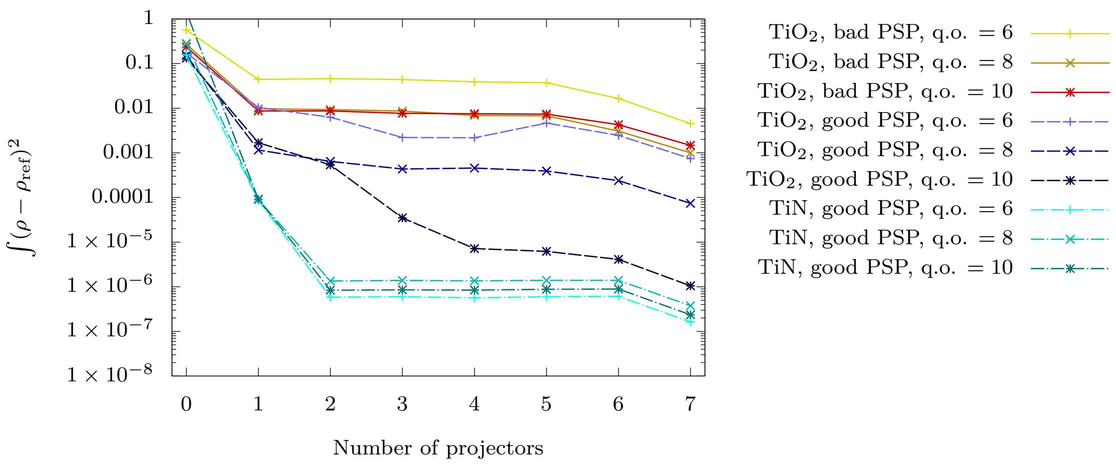


Figure 4.4: Analysis of the $l = 2$ nonlocal part of Titanium pseudopotentials and the effect of quadrature order on the precision of the computation.

Better results, presented as the "good pseudopotential" in Fig. 4.4 for $l = 2$, were obtained with a pseudopotential tuned for the titanium nitride (see the environment reflecting pseudopotentials in [20]). Only two projectors for each l have been necessary to achieve an acceptable precision in the case of titanium nitride. For a (by the charge distribution) relatively similar molecule of titanium dioxide (both of them have a covalent bond), substantially more projectors and a high quadrature order have to be used to achieve a similar precision, because the pseudopotential was tuned for TiN and not TiO2...

Note that in this case the states for l=2 are occupied and the results for l01 were similia.

These results suggest (or maybe better confirm), that the transferability of a pseudopotential is a very substantial property of the pseudopotential, and that the construction of a special (*environment reflecting*) pseudopotential for the particular atomic position [20] is one of the first approaches, that one could try if computational difficulties occur (e.g. when a slow convergence/divergence or unphysical results are observed).

Moreover, the results suggest the following hypothesis. Better results could possibly be obtained, if the projectors would be based not on the isolated atomic wave functions (which, however, reflect the environment, for which the pseudopotential has already been tuned, see [20]), but on the actual wave functions from a computed material (or if the projection basis would be enriched by such wave functions). This hypothesis will be examined by our further research.

Analysis of the quadrature order Another important observation can be demonstrated on the case of the titanium pseudopotentials. It is derived at the end of Section 3.2.2 that a suitable numerical quadrature for the density functional theory terms evaluated using cubic finite elements should be at least of the fourth order.

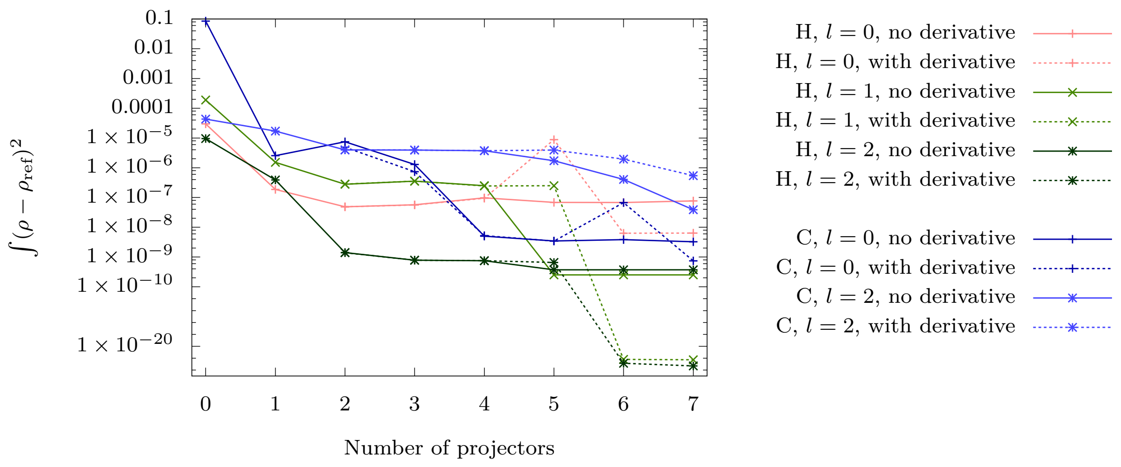


Figure 4.5: Hydrogen pseudopotentials in nitric acid and carbon pseudopotential in carbon oxide: projectors with and without the energy derivative. The logarithmic scale on the y axis is compressed below 1×10^{-10} to emphasize the important region.

However, if we consider the use of pseudopotentials, this is no longer valid. Spherical harmonic functions (4.23) are not polynomial, but rational functions (of coordinates and radius) and therefore the Gauss integration scheme is not exact for them (see Section 3.2.2). This behavior is especially important for pseudopotentials with a higher l, where the directionality of the potential and so the “rationality” of the projectors are more substantial.

Therefore, to use a higher-order quadrature to make the approximation more precise seems to be a viable approach. This hypothesis is supported by the presented sample calculations: see again Fig. 4.4. In the case of titanium oxide, the accuracy substantially increases with a given number of projectors if a higher quadrature order is used.

Surprisingly, this behavior did not appear in the case of titanium nitride, where mainly the first two projectors are substantial. There could be two reasons: first, the inaccuracy of the quadrature is random — a rational function may or may not be well-approximated by a polynomial. If only a few projectors are needed, there is a greater chance, that all of them can be well-approximated. And second: the error of the approximation is amplified by the loss of the orthogonality of the projectors, which is unavoidable in finite precision arithmetic for more projectors than two.

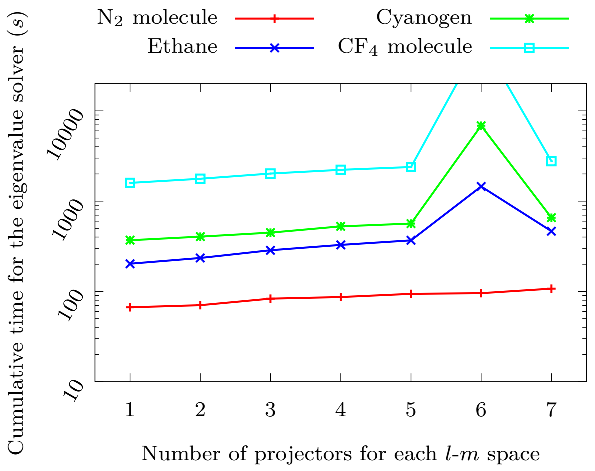
Projectors created using energy derivatives The idea of using energy derivatives for the construction of projectors was tested. To a given number of projectors for a given pseudopotential, made from the first n atomic wave functions, one “derivative projector”, constructed using a numerical (with $\delta = 0.1\mathrm{E_h}$) energy derivative of the ground state wave function ψ

$$\frac{\partial\psi(e)}{\partial e} = \frac{\psi(e+\delta) - \psi(e-\delta)}{2\delta}\,, \tag{4.52}$$

was added.

This approach yields surprisingly nearly identical results as adding one more (excited) wave function in most of the cases — apparently, both kinds of projectors act as a similar enrichment of the basis. Those cases are not presented as the differences are negligible and the lines would coincide.

Only in a few cases the “derivative projectors” performed better than the standard ones, e.g. for a hydrogen pseudopotential in the case of nitric acid, see Fig. 4.5. The results obtained with a carbon pseudopotential in carbon oxide were ambiguous, the derivative slightly improved the precision for $l = 0$, the results were nearly identical for $l = 1$, but for $l = 2$ including the energy derivatives worsened the results.



#projectors	CF_4	Cyanogen	Ethane	N_2
1	1591	369	203	67
2	1773	405	234	70
3	2023	474	285	83
4	2223	526	327	87
5	2390	565	367	94
6	61482	6849	1450	96
7	2776	655	465	107

Cumulative time for the eigenvalue solver(s)

Figure 4.6: Dependence of the sums of eigenvalue solver times from iterations of the DFT cycle on the number of projectors in each l-m space. The results were computed using eight cores of 2×6 core Xeon E5645 [171], clocked at 2.4Ghz with 800Mhz DDR3 memory. The adaptive hexahedral mesh (see Section 3.2.1) and Jadamilu eigenvalue problem solver (see Section 6.3.2) were used for the computation. The two- and more-projectors setups included the energy derivatives.

Performance analysis To decide how many projectors to use, the knowledge of the impact of the number projectors on the performance is important. Therefore, a brief analysis is presented below. We will focus on the performance of an eigenvalue solver, which consumes the majority of the computational time.

Unlike in the convergence analysis in Section 3.4.2, there is no theoretical parameter (like the matrix dimension or the number of nonzeros), that could be used as a metric for the performance of the eigenvalue solver, so we will compare real elapsed times. The resulting times for a given number of projectors for a single l-m space[8] are presented in Fig. 4.6.

The performance of the eigenvalue solver depends on many things: the used computer architecture (size and organizations of cache, implementation of memory controller etc. . .), the used solver, the used underlying BLAS library, etc. The elapsed times should be thus taken only in mutual relations. On the other hand,

[8]The total number of projectors in the system was $(1 + 3 + 5) * n *$ number of atoms.

the performance of any iterative solvers always depends on the matrix-vector multiplication performance, therefore, the presented results give a reasonable guess of the relative performance penalty for including more projectors into a computation.

The results are pretty consistent: the elapsed time for seven projectors is about twice as large than for a single projector (the only exception, the case of six projectors, will be discussed later). The implementation of the rank-k-updates could be improved by utilizing some block-sparse matrix storage format, however, the presented results show, that the performance of the rank-k-update matrix-vector multiplication is not a problem, or at least not the problem that should be addressed now.

Remark on ghost states The exceptionally high computation times for six projectors in Fig. 4.6 are caused by *ghost states*. By the term ghost states we mean the non-physical electron states, caused as another unwanted side-effect of numerical errors. The reasons of their appearance can be various, e.g. so-called *spurious eigenvalues* from eigenvalue solvers (e.g. [172]), that are mainly due to the loss of orthogonality during solver iterations [173]. Eigenvalue solvers often incorporate their own methods for the suppression of the spurious eigenvalues, e.g. the reorthogonalization in Lanczos solvers [173] or the purification [174].

In the current case, the source of the ghost states is similar: the B-orthogonalization of projectors (see Definition 4.10 on page 121). The orthogonalization is done in the radial space — and the inexactness of the Gauss integration scheme for rational functions causes a non-orthogonality of the resulting 3D projectors, which leads to the ghost states appearance.

The ghost states appear only for a single number of projectors in most cases: adding one or more projectors causes them to disappear again. Most often this number of projectors is six. The ghost states also appear more often in the cases, where the projectors include the energy derivatives. The reason might be a lack of a “natural orthogonality” of the derivative to the other projectors: atomic wave functions are orthogonal, while their energy derivatives are not, and the orthogonality apparently reduces the numerical errors from the finite precision arithmetic during the B-orthogonalization.

The problem could be solved by some kind of reorthogonalization, either in the radial or 3D space. This problem can be easily solved by not choosing the numbers of projectors that suffer from the issue. Thus, there was no need for a deeper examination of the matter and resolving the issue rigorously.

4.3.3 Conclusions and recommendations

We can see from the results that there is no general rule on how to determine the “best” number of projectors. The more a pseudopotential fits the computed case, the fewer projectors are usually needed. Two or three projectors yield often sufficiently precise results, however, it should be verified even with pseudopotentials tuned to a particular case [20] that adding a projector does not (significantly) change the results.

The use of a single projector can be recommended only if the required precision is low and/or when the l-space of the projector is unoccupied (this result is in agreement with the recommendations of the other authors [169]). For the same reason, it is not recommended to use pseudopotential formats, that allow only a single projector for an l-space. The overestimation of the number of projectors is also not recommended since the performance suffers and the hazard that ghost states appear increases with the number of projectors.

The implementation of the matrix-vector multiplications with the rank-k-updates matrix has not been fully optimized yet. Thus, if the performance penalty for using too many projectors becomes an issue, there is a space for performance improvement. However, the performance of the solver itself is substantially more crucial than the number of projectors, so implementing more sophisticated algorithms for this multiplication is not necessary in the current state of development.

A computation with the projectors requires a higher order of the numerical quadrature. The results also suggest that the required number of projectors is not dependent on the orientations of atomic bonds with regard to the mesh. Adding a projector built from the energy derivative seems to have a similar effect on the precision as adding one projector built from an excited state. In some cases the energy derivative projectors perform better, but such cases are not common.

5. Geometry optimization, total energy and Hellmann-Feynman forces

Contents

Geometry optimization, *structure optimization* or *structure relaxation*, means an optimization of atomic positions with respect to the internal energy of the system. Such a task requires an efficient evaluation of two quantities: of the *total energy* of the system and of the derivatives of the total energy with respect to the position of the atomic sites, which mans (called after two famous physicist and authors of one useful theorem) *Hellmann-Feynman forces.*

In this chapter, efficient formulas for both the total energy and Hellmann-Feynman forces will be derived. Using the Kohn-Sham equations for the total energy evaluation is a well-known trick [175]) and it can be used for evaluating the Hellmann-Feynman forces, too. A theoretical novelty in this chapter is the application of the Hellmann-Feynman theorem in the finite element method in combination with pseudopotentials. The used approach and the resulting expressions have not been published in the literature yet.

The most difficult part of the calculation of the Hellmann-Feynman forces is the evaluation of the term arising from the gradient of nonlocal pseudopotentials because the derivatives of pseudopotentials include (among other terms) also the derivatives of the projection operators to l-subspaces which result in singularities. In this work, I have proposed a new expression for evaluating this part of the force, based on differentiating the wavefunctions instead of the projectors.

The new expression for Hellmann-Feynman forces in finite elements is verified by comparing equilibrium atomic positions (i.e. positions with zero forces) to minima of total energies, to results obtained via other computational approaches, and to experimental values (for nitric oxide, carbon dioxide, and tetrafluoromethane) and by comparing vibrational frequencies (of carbon dioxide). The convergence property of the expression is analyzed and the expression is compared to other previously published approaches to the non-local contribution to HF forces as regards accuracy and computational demands. This comparison shows the advantages of the new formula in accuracy or/and computational efficiency compared to so far published formulas.

The first section of this chapter contains several alternatives on how to express total energy in finite element discretization with pseudopotentials. The second section contains some theoretical background for the force calculations within density functional theory: the well known Hellmann-Feynman theorem and a discussion of the incomplete basis set correction.

In the third section, an efficient and numerically stable way for evaluating both the local and non-local parts of the Hellmann-Feynman forces within the finite element method is described. Special attention is dedicated to the non-local part, giving a brief overview of various approaches to calculations of nonlocal components of HF forces and illustrates the importance of incorporating all subcomponents of the nonlocal parts.

The last section presents the results of sample calculations that demonstrate the correctness of the formula and contains a brief convergence and error analysis in comparison with other approaches. The theoretical computational demands of the compared methods are analyzed too. The error of the computed Hellmann-Feynman forces is analyzed with respect to the stopping criteria of the DFT iterations and with respect to the mesh element size and number of basis functions, and the fluctuations of the results obtained by various methods are compared.

5.1 Total energy for pseudopotentials

In the second chapter, we derived the expression (2.37) for the *total energy* of the solution to Kohn-Sham equations (2.44). It can be formulated using the energy operator H_e, commonly called *Hamiltonian*:

$$\mathcal{E}_\rho = \sum_i \langle \psi_i | H_\mathrm{e} | \psi_i \rangle = \sum_{i=1}^{n} \int \psi_i^+ \frac{1}{2} \nabla^2 \psi_i + \int V_\mathrm{ext} \rho + \int E_\mathrm{H}(\rho) + \int E_\mathrm{xc}(\rho) \,. \quad (5.1)$$

To perform the geometry optimization correctly, two additional facts should be considered: First, moving the atomic sites, the energy $\mathcal{E}_\text{i-i}$ of repulsion among atomic cores will begin to change and it must be taken into account in optimizing the total energy of the system. Moreover, the smearing of the occupation of one-electron states (see equation 2.50 on page 34) has to be incorporated into the total energy expression.

Denoting the total energy including $\mathcal{E}_\text{i-i}$ by $\mathcal{E}_\mathrm{tot}$, we can express it as

$$\mathcal{E}_\mathrm{tot} = \mathcal{E}_\rho + \mathcal{E}_\text{i-i} = \sum_{i=1}^{n} w_i \int \psi_i^+ \frac{1}{2} \nabla^2 \psi_i + \int V_\mathrm{ext} \rho + \int E_\mathrm{H}(\rho) + \int E_\mathrm{xc}(\rho) + \mathcal{E}_\text{i-i} \,. \quad (5.2)$$

If we denote the parts of the total energy expression as

$$\mathcal{E}_{\text{kin}} = \sum_{i=1}^{n} w_i \int \psi_i^+ \frac{1}{2} \nabla^2 \psi_i \tag{5.3}$$

$$\mathcal{E}_{\text{ext}} = \int V_{\text{ext}} \rho \tag{5.4}$$

$$\mathcal{E}_{\text{H}} = \int E_{\text{H}}(\rho) \tag{5.5}$$

$$\mathcal{E}_{\text{xc}} = \int E_{\text{xc}}(\rho) \, . \tag{5.6}$$

the expression for total energy will be

$$\mathcal{E}_{\text{tot}} = \mathcal{E}_{\text{kin}} + \mathcal{E}_{\text{ext}} + \mathcal{E}_{\text{H}} + \mathcal{E}_{\text{xc}} + \mathcal{E}_{\text{i-i}} \tag{5.7}$$

5.1.1 The terms of the total energy

Energy of pseudopotentials Using pseudopotentials, the energy $\mathcal{E}_{\text{ext}}$ must be expressed accordingly to the equations 4.30 and 4.31 on page 120:

$$\mathcal{E}_{\text{ext}} = \sum_j \int V_{\text{loc}}^j \rho + \sum_{i,j,l} w_i \int \psi_i^+ R_l^j P_l^j \psi_i = \sum \mathcal{E}_{\text{loc}}^j + \sum_{j,l} \mathcal{E}_{\text{nl}}^{j,l} \tag{5.8}$$

where i runs over all occupied eigenvectors, j over all atomic sites and l over l-subspaces related to the given atomic site. We can denote

$$\mathcal{E}_{\text{loc}} = \sum V_{\text{loc}}^j \rho \tag{5.9}$$

$$\mathcal{E}_{\text{nl}} = \sum_{i,j,l} w_i \int \psi_i^+ R_l^j P_l^j \psi_i \, . \tag{5.10}$$

Hartree energy Because the expression for Hartree energy $\mathcal{E}_{\text{H}}$ contains double integral (let's recall its definition in equation 2.46 on page 31)

$$\mathcal{E}_{\text{H}} = \int E_{\text{H}} = \frac{1}{2} \int_{\mathbf{y}} \int_{\mathbf{x}} \frac{\rho(\mathbf{x})\rho(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|} \tag{5.11}$$

it is more convenient to evaluate it using the Hartree potential

$$\mathcal{E}_{\text{H}} = \frac{1}{2} \int V_{\text{H}} \rho \tag{5.12}$$

Note the factor one half: as the electron-electron repulsion acts on both the electrons, they share the energy of the force. Not to count the energy of an electron-electron pair twice, the resulting expression should be divided by two.

Ion-ion energy The energy of atomic nuclei repulsion is given by a Coulombic interaction:

$$\mathcal{E}_{\text{i-i}} = \frac{1}{2}\sum_{a\neq b} \frac{Z_a Z_b}{\|\mathbf{x}_a - \mathbf{x}_b\|} \,. \tag{5.13}$$

Pseudopotentials smooth the potentials of atomic nuclei by replacing them by atomic cores (since the atomic core represents the nucleus together with non-valence electrons and dissolves the singularity). Asymptotic behaviour is the same as (5.13), but for more precise total energy calculations, a real charge-potential energy should be taken into account.

Since the nonlocal parts of a pseudopotential are short-range and they vanishes in distances of other nuclei, only the local long-range potential is to be considered. Thus, core charges be put into effect using the Poisson equation

$$\rho^a_{\text{ion}} = -\frac{1}{4\pi}\nabla^2 V^a_{\text{loc}} \tag{5.14}$$

and the energy can be expressed as (e.g. [176])

$$\mathcal{E}_{\text{i-i}} = \frac{1}{2}\sum_{a\neq b}\int V^a_{\text{ion}}\rho^b_{\text{ion}} \,. \tag{5.15}$$

The number of terms in the sum in the equation 5.15 grows quadratically with the number of atoms. Defining

$$V_{\text{ion}} = \sum_b V^b_{\text{ion}} \qquad \rho_{\text{ion}} = \sum_a \rho^a_{\text{ion}} \,, \tag{5.16}$$

a simple transformation can be used

$$\frac{1}{2}\sum_{a\neq b}\int V^a_{\text{ion}}\rho^b_{\text{ion}} = \frac{1}{2}\left(\sum_{a,b}\int V^a_{\text{ion}}\rho^b_{\text{ion}} - \sum_a\int V^a_{\text{ion}}\rho^a_{\text{ion}}\right) = \frac{1}{2}\left(V_{\text{ion}}\rho_{\text{ion}} - \sum_a\int V^a_{\text{ion}}\rho^a_{\text{ion}}\right) \,, \tag{5.17}$$

which results in the expression with a linear dependency on the number of atomic cores.

5.1.2 Efficient computation of total energy

The evaluation of the energy of nonlocal pseudopotentials $\mathcal{E}_{\text{nl}}$ requires computationally demanding spherical integration, which should be avoided in FEM calculations. Moreover, it is desirable to avoid these terms in assembling the Hamiltonian matrix, too. The spherical integration could be avoided either using the rank-k-update approach, or by substituting the Kohn-Sham equations into the total energy expression.

Direct approach using separable pseudopotentials and Laplace matrix

The separable form of the nonlocal part of a pseudopotential V^l_{nl} is defined in chapter 4.2.3 by equation 4.45 as:

$$V^l_{\text{nl}} = \sum_{n,m} d_n \left|\nu_{l,n} \times Y_{l,m}\right\rangle\!\left\langle\nu_{l,n} \times Y_{l,m}\right| \,. \tag{5.18}$$

If we express a wavefunction ψ in a finite element basis φ_k

$$\psi = a_k \varphi_k \,, \tag{5.19}$$

we can write the energy of nonlocal parts of a pseudopotential V^l_{nl} as

$$\mathcal{E}_{\text{nl}}(V^l_{\text{nl}}) = \sum_i w_i \int \psi_i^+ R^j_l P^j_l \psi_i \tag{5.20}$$

$$= \sum_{i,k,n,m} w_i \int (a_{k,i}\varphi_k)^+ \, d_n \, (\nu_{l,n} \times Y_{l,m}) \int (\nu_{l,n} \times Y_{l,m})^+ a_{k,i}\varphi_k \tag{5.21}$$

$$= \sum_{i,k,n,m} w_i |a_{k,i}|^2 \int \varphi_k^+ d_n \, (\nu_{l,n} \times Y_{l,m}) \int (\nu_{l,n} \times Y_{l,m})^+ \varphi_k \,. \tag{5.22}$$

If we precompute the following set of vectors

$$u_{l,n,m,k} = \varphi_k^+ \, (\nu_{l,n} \times Y_{l,m}) = (\varphi_k \,\cdot\, \nu_{l,n} \times Y_{l,m}) \,, \tag{5.23}$$

we can express the nonlocal pseudopotential part of the total energy (5.20) as

$$\mathcal{E}_{\text{nl}}(V^l_{\text{nl}}) = \sum_{n,m,k,i} w_i |\mathbf{a}_{k,i}|^2 (\mathbf{u}_{l,n,m,k} \,\cdot\, \mathbf{u}_{l,n,m,k}) \,. \tag{5.24}$$

The first kinetic term of (5.2) can be easily expressed using the Laplace matrix L of the finite element basis:

$$\sum_i w_i \psi_i^+ \frac{1}{2}\nabla^2 \psi_i = \sum_{i,k} w_i \, (a_{i,k}\varphi_k)^+ \, \frac{1}{2}\nabla^2 a_{i,k}\varphi_k = \sum_i (\mathbf{a}_i \,\cdot\, L\mathbf{a}_i) \tag{5.25}$$

The Laplace matrix L can be already precomputed during the DFT selfconsistent cycle. Consequently, the kinetic energy expression can be easily evaluated without the need for assembling the matrix again in each iteration. All the remaining terms in (5.2) can be simply expressed as products of appropriate potentials with the charge density.

Total energy computing using Kohn-Sham equations

Another way to avoid the demanding spherical integration is to integrate Kohn-Sham equations (2.44):

$$\frac{1}{2}\nabla^2\psi_i + \sum_j V^j_{\text{loc}}\psi_i + \sum_{j,l} \int V^j_l P^j_l \psi_i + V_{\text{H}}\psi_i + V_{\text{xc}}\psi_i = e_i\psi_i \,, \tag{5.26}$$

with ψ_i^+

$$\int \psi_i^+ \frac{1}{2}\nabla^2\psi_i + \sum_j \int \psi_i^+ V_{\text{loc}}^j \psi_i + \sum_{j,l} \int \psi_i^+ V_{\text{nl}}^{j,l} P_l^j \psi_i + \int \psi_i^+ (V_{\text{H}} + V_{\text{xc}})\psi_i = e \int \psi_i^+ \psi_i \,. \tag{5.27}$$

Because the wavefunctions are normalized, the right side of the equation is equal to e_i — i.e. the *Kohn-Sham energies* of wavefunctions. Note that they are not the true energies of the electrons — although there are some relations in between (so, e.g. the eigenstates are occupied in the order of their Kohn-Sham energies) — the total energy of the system cannot be obtained by just summing the Kohn-Sham energies.

Since the Kohn-Sham equations hold for all theirs solutions, we can substitute each solution into equation (5.27) and perform a weighted (to incorporate smearing) sum of the resulting equations. We obtain

$$\sum_{i=1}^{n} w_i \int \psi_i^+ \frac{1}{2}\nabla^2\psi + \sum_j \int V_{\text{loc}}\rho + \sum_{i,j,l} w_i \int \psi_i^+ V_l^j P_l^j \psi_i + \int (V_{\text{H}} + V_{\text{xc}})\rho = \sum_i w_i e_i \tag{5.28}$$

Denoting

$$\mathcal{E}_{\text{Vxc}} = \int V_{\text{xc}}\rho \,, \tag{5.29}$$

The summed equations (5.28) can be expressed as

$$\mathcal{E}_{\text{kin}} + \mathcal{E}_{\text{loc}} + \mathcal{E}_{\text{nl}} + 2\mathcal{E}_{\text{H}} + \mathcal{E}_{\text{Vxc}} = \sum_i w_i e_i \tag{5.30}$$

and the equation (5.2) as

$$\mathcal{E}_{\text{kin}} + \mathcal{E}_{\text{loc}} + \mathcal{E}_{\text{nl}} + \mathcal{E}_{\text{H}} + \mathcal{E}_{\text{xc}} + \mathcal{E}_{\text{i-i}} = \mathcal{E}_{\text{tot}} \,. \tag{5.31}$$

Substracting them yields

$$-\mathcal{E}_{\text{H}} - \mathcal{E}_{\text{Vxc}} + \mathcal{E}_{\text{xc}} = e - \sum_i w_i e_i \tag{5.32}$$

and we obtain the final expression for the total energy:

$$\begin{aligned} \mathcal{E}_{\text{tot}} &= \sum_i w_i e_i - \mathcal{E}_{\text{H}} - \mathcal{E}_{\text{Vxc}} + \mathcal{E}_{\text{xc}} + \mathcal{E}_{\text{i-i}} = \\ &\sum_i w_i e_i + \int \left(E_{\text{xc}} - V_{\text{xc}} - \frac{1}{2} V_{\text{H}} \right) \rho + \frac{1}{2} \left(\int V_{\text{ion}}\rho_{\text{ion}} - \sum_a \int V_{\text{ion}}^a \rho_{\text{ion}}^a \right) \,. \end{aligned} \tag{5.33}$$

The final expression contains only few integrals that can be evaluated easily and without substantial computational demands. Unlike the “direct approach” (Eq. 5.2), this expression does not contain wavefunctions, but only the charge density. Therefore, its evaluation requires only scalar products fo vectors — not the matrix elements — and thus it is a more efficient way to evaluate the total energy. However, both of the approaches are useful, as equality of their results is a good check of the convergence of the DFT calculation.

Total energy using ionic charge

Pask and Sterne in [175] introduced another transformation for total energy. Their approach aims at periodical structures and its motivation is to avoid integrals that diverge in an infinite periodical space. Such an issue is not applicable in real space calculations for a finite sample, but the presented transformation offers numerical benefits even in real space.

First, let us define the electrostatic energy as

$$\mathcal{E}_{\text{es}} = \mathcal{E}_{\text{H}} + \mathcal{E}_{\text{loc}} + \mathcal{E}_{\text{i-i}} \tag{5.34}$$

Applying (5.17) and using the following "symmetry" or electron-ion forces:

$$\mathcal{E}_{\text{loc}} = \int V_{\text{ion}}\rho = \int V_{\text{H}}\rho_{\text{ion}} \tag{5.35}$$

the electrostatic energy can be expressed as

$$\begin{aligned}\mathcal{E}_{\text{es}} &= \frac{1}{2}\int V_{\text{H}}\rho + \int V_{\text{ion}}\rho + \frac{1}{2}\left(\int V_{\text{ion}}\rho_{\text{ion}} - \sum_a \int V^a_{\text{ion}}\rho^a_{\text{ion}}\right) \\ &= \frac{1}{2}\left(\int V_{\text{H}}\rho + \int (V_{\text{ion}}\rho + \rho_{\text{ion}}V_{\text{H}}) + \int V_{\text{ion}}\rho_{\text{ion}} - \sum_a \int V^a_{\text{ion}}\rho^a_{\text{ion}}\right) \\ &= \frac{1}{2}\left(\int (V_{\text{H}} + V_{\text{ion}})(\rho + \rho_{\text{ion}}) - \sum_a \int V^a_{\text{ion}}\rho^a_{\text{ion}}\right)\end{aligned} \tag{5.36}$$

Note that such expression, in fact, averages the two ways mentioned above to calculate $\mathcal{E}_{\text{loc}}$ term (see Eq. 5.35). Since these two ways diverge in opposite direction from the correct value of the total energy in practise, such method commonly offers lower numerical errors in a total energy calculation. Such an $\mathcal{E}_{\text{loc}}$ energy averaging can be done both in "Kohn-Sham way" of computing the total energy as well as in the case of applying the separable pseudopotentials.

5.2 Hellmann-Feynman forces

For an efficient structure optimization, the forces acting on atoms are needed besides the total energy of the system.

The forces acting on atomic nuclei (or on atomic cores, in case of pseudopotentials) — commonly called *Hellmann-Feynman forces* and abbreviated as *HF-forces* — can be calculated in the same way as in classical physics, i.e. as a negative gradient of the potential energy with respect to the positions of the atomic sites. The total energy $\mathcal{E}_{\text{tot}}$ plays the role of potential energy in this case.

It has been already stated (see Eq. 5.1) that the total energy of the system is determined by the "electronic" part of the Hamiltonian

$$H_e = \frac{1}{2}\nabla^2 + \sum_j V_{\text{loc}} + \sum_{j,l} V^{j,l}_{\text{nl}} + E_{\text{H}} + E_{\text{xc}} \tag{5.37}$$

and by the external potential, that yields the external energy $\mathcal{E}_{\text{i-i}}$. Denoting the gradient with respect to the position of the a-th atomic site by ∇_a, we can express the gradient of the equation (2.37) as

$$f = -\nabla_a \mathcal{E}_{\text{tot}} = -\nabla_a \left(\sum_i w_i \int \psi_i H_e \psi_i + \mathcal{E}_{\text{i-i}} \right) \tag{5.38}$$

Because a move of any atomic center results in a change of wavefunctions and eigenvalues, a direct calculation of the gradient would mean running the costly DFT convergence cycle both for the original system and for the system with the atomic core moved. Because such a computation would have to be done for each of the three axes and for each of the atomic sites in the system to get the complete set of forces, another approach has been found: Hellmann-Feynman theorem [177] makes the evaluation of the equation (5.38) much simpler.

Theorem 5.1 (Hellmann-Feynman theorem)**.** *Let x be a parameter which the Hamiltonian H_e is dependent on (e.g. a position of an atomic centre). Then it holds, for each ground state wavefunction ψ_i,*

$$\frac{de_{\psi_i}}{dx} = \frac{d \int \psi_i^+ H_e \psi_i}{dx} = \int \psi_i^+ \frac{dH_e}{dx} \psi_i \,. \tag{5.39}$$

This theorem can be rephrased as: the derivative of the total energy w.r.t. atomic position can be calculated as if the charge density remained frozen (i.e. unchanged while moving the atomic site).

The commonly cited proof of the theorem (see e.g. the original proof in [177]) considers the theorem as a direct consequence of the variational principle. This approach uses a bit vague assumptions, however. A meticulous reader thus may be interested in a fully mathematical exact proof of the theorem, as it is built in [178]. Nevertheless, the intuitive variational approach is sufficient for our needs and it is much more intelligible than the fully rigorous proof, so we will follow it.

Proof. By the chain rule, it holds

$$\frac{de_{\psi_i}(x)}{dx} = \frac{\partial e_{\psi_i}}{\partial x} + \frac{\partial e_{\psi_i}}{\partial \psi_i} \frac{d\psi_i}{dx} \,. \tag{5.40}$$

The variational principle (see theorem 2.1 on page 26) states, that $\frac{\partial \mathcal{E}_{\text{tot}}}{\partial \psi}$ is zero for a ground state, so the second term vanishes. Therefore, as ψ is not directly dependent on x, holds

$$\frac{d\mathcal{E}_{\text{tot}}(x)}{dx} = \frac{\partial \mathcal{E}_{\text{tot}}}{\partial x} = \int \psi^+ \frac{\partial H_e}{\partial x} \psi = \int \psi^+ \frac{dH_e}{dx} \psi \,. \tag{5.41}$$

□

5.2.1 Supplementary terms to the Hellmann-Feynman forces

The argumentation above relies on an important assumption: that the wavefunctions are the true ground state wavefunctions of a given Hamiltonian. In practice, when a discretization scheme is used in the calculation, the wavefunctions are not exact ground state wavefunctions, but only their projections to the given (e.g. finite element) subspace. Moreover, the charge density has some convergence error. Therefore, the Hellmann-Feynman theorem could be applied just with caution, since both of these facts may (or may not) spoil the particular force calculation. If they do, the correction terms that cure the issue should be applied. In the following text, the questions of the necessity of such corrections will be discussed for our case.

Incomplete basis set correction If the charge density described by a finite basis differs from the true charge density, the Hellmann-Feyman theorem does not hold and the gradients of the wavefunctions should be incorporated:

$$\nabla_a \mathcal{E}_\rho = \nabla_a \sum_i w_i \langle \psi_i | H | \psi_i \rangle = \sum_i w_i \Big(\langle \psi_i | H | \psi_i \rangle + 2\,\mathrm{Re}\, \langle \nabla_a \psi_i | H | \psi_i \rangle \Big) . \tag{5.42}$$

The extra (second) term is called *incomplete basis set correction*, abrreviated as *IBS*, or Pulay force[1] according to the name of the author who introduced the correction [179].

The need for IBS correction arise particularly in the case of basis dependent on structure geometry, as e.g. basis created from atomic orbitals in LMTO [14] or LAPW class methods [13, 180, 181]. In those methods, the corresponding basis functions are moved with the relevant atomic site to retain the capability of the basis to describe the wavefunctions. As it is described in [180], this basis transformation can be used for the approximation of the IBS term:

$$f_{\mathrm{IBS}} = -\sum w_i c_{i,j} c_{i,k} \left(2\,\mathrm{Re} \left\langle \frac{d\varphi_i}{da} \middle| H - e_i \middle| \varphi_j \right\rangle + \left\langle \varphi \middle| \frac{d\nabla^2}{da} \middle| \varphi_j \right\rangle \right) \tag{5.43}$$

For the case of the atomic-position-independent basis (e.g. planewaves) the approximation (5.43) of the IBS term is zero, since the gradient of the basis functions with respect to the change of atomic positions is zero. These facts lead to the commonly shared and cited belief that the IBS term should be considered for "non-fixed" bases only. This statement is nearly right in practice: e.g. planewaves calculations do not usually show significant error in forces. However, this argumentation is not completely correct.

Imagine a movable basis of atomic orbitals and the corresponding HF force calculated for a given atomic position. We can fix the basis in the space — i.e. to make it independent of the current position of atomic sites — move the atom slightly and perform the DFT cycle and evaluate the forces again. Of course, it will change nothing (except for the ion-ion interaction): the basis, the wavefunctions, and the "electronic" part of the forces — everything remains the same. The force will still

[1]Some authors denote only the approximation of the force given by (5.43) as Pulay force.

contain the error that requires IBS correction (5.43). It is not surprising since the expression (5.39) for the HF force is independent on any basis transformation carried out during geometry optimizations. What is changing during the geometry optimization iterations is just the approximation of the Hamiltonian, not the electronic part of the Hamiltonian itself.

The force expresses a measurable physical quantity that should be, for a given geometry, evaluated as precisely as possible, and it must be independent of any particular choice of the basis: if the way of discretization changed the value of the force, it would mean that the calculation is wrong.

It clearly follows from the reasoning above that the gradient in the (5.43) should not be necessarily taken with respect to the actual move of the basis. Optimally, it should be evaluated in terms of the basis transformation, which can describe the moved wavefunctions most precisely. The better the transformation is handled, the more exact force will be obtained — even though it could or could not match with the actual transformation of the basis performed during the structure optimization. For an atomic-position-dependent basis, selecting the same transformation for generating the IBS term as it is applied in the structure optimization process is a reasonable choice, which yields (5.43).

For planewaves, where the "completeness" of the basis does not change with respect to atomic positions, no need for incorporating any transformation into the force calculation commonly arise, which results in zero IBS term — providing that the IBS errors are effectively averaged out [182]. However, for a coarse planewave basis set (low energy cutoff) the IBS term could be noticeable (see [183]). The importance of the IBS correction grows if the volume of the first Brillouin zone changes during structure optimization, since the change of the volume causes changes in the energies of the basis wavefunctions.[182] In such cases, the IBS correction can reduce the force fluctuations, and therefore it can improve the convergence of the forces.[184]

Thus, the Eq. (5.43) should be understood in the following way: the IBS correction does not follow from the movement of the basis elements, but that the movement of the basis elements can be used as a good approximation to the IBS terms. It is not surprising that in the original article introducing the Puylay forces [179], the "right direction" of argumentation is applied, while the real essence of the problem is obfuscated in some later works of other authors.

FEM basis is built so that it is as independent of atomic positions as possible. So it could be expected to behave like planewaves. And, indeed, the numerical experiments confirm that the calculation of HF forces converges to the correct values (see e.g. Fig. 5.7 and 5.6). Therefore we assume that the IBS term is insignificant in our case and it can be neglected. Possible incorporating some form of IBS correction could result in lower requirements on the mesh fineness and, consequently, possibly lower computational demands, which is opened for further research at present time.

Non-self-consistency correction For the sake of completeness, another correction called *non-self-consistency correction* should be mentioned [185, 186]:

$$f_{\text{NSC}} = -\int (\nabla_a \rho)(V_{\text{ext}} - V'_{\text{ext}}) \tag{5.44}$$

It is aimed at the difference between the calculated and the actual ground states that is brought about by just a finite precision that the DFT self-consistent cycle always converge with. In the expression above, V_{ext} and V'_{ext} potentials correspond to the charge density computed during the last iteration and the ideal (hypothetically fully converged) ground state charge density, respectively. This correction should lower the requirements for the convergence precision of the DFT cycle, which could be useful particularly during the early iterations of geometry optimization.

However, our numeric experiments show a pretty nice correlation between the convergence error and the HF forces error (see Fig. 5.4). Therefore the benefit of this correction was not expected to be worth implementing at the current stage of the FENNEC development. It is planned as a rather independent task to be done e.g. by a student in his/her bachelor or master thesis.

5.3 Applying HF-theorem within the finite element method with pseudopotentials

According to (5.39) there is no implicit (via the dependence of the wavefunctions) dependence of the HF force on the position of the atomic centres. Therefore the gradient of the total energy contains only the terms with explicit dependence on atomic positions and its evaluation seems to be straightforward in principle, as have been published in [2]:

$$\nabla_a \mathcal{E}_{\text{tot}} = \int \nabla_a V^a_{\text{loc}} \rho + \sum_{l,i} w_i \int \psi_i^+ \nabla_a \left(V^{a,l}_{\text{nl}} P^a_l \right) \psi_i + \nabla_a \mathcal{E}_{\text{i-i}} \,. \tag{5.45}$$

However, in the case of pseudopotentials, this expression leads to the necessity of differentiating the projections to l-subspaces in which case the numerical errors arise due to singularities of the gradients of spherical harmonics at atomic centers. Therefore, this task requires special attention. First, we will discuss other – simpler – parts of the expression (5.45).

5.3.1 Local parts of Hellmann-Feynman force

The first term of (5.45) is the local part of electron-ion interaction (see Eq. 4.30 on page 120). The gradient of the local ion potential can be easily evaluated as the partial derivative of the potential in the radial space multiplied by the unit vector in direction from the atomic center, i.e. unit vector of the position relative to the atomic center:

$$\int \nabla_a V^a_{\text{loc}} \rho = \int \frac{\partial V^a_{\text{loc}}(r)}{\partial r} \frac{\mathbf{x} - \mathbf{c}_{\text{a}}}{r} \rho \,. \tag{5.46}$$

Only slightly more effort is required to evaluate the last term of (5.45) expressing the repulsion of atomic nuclei/cores. Without using pseudopotentials, this term

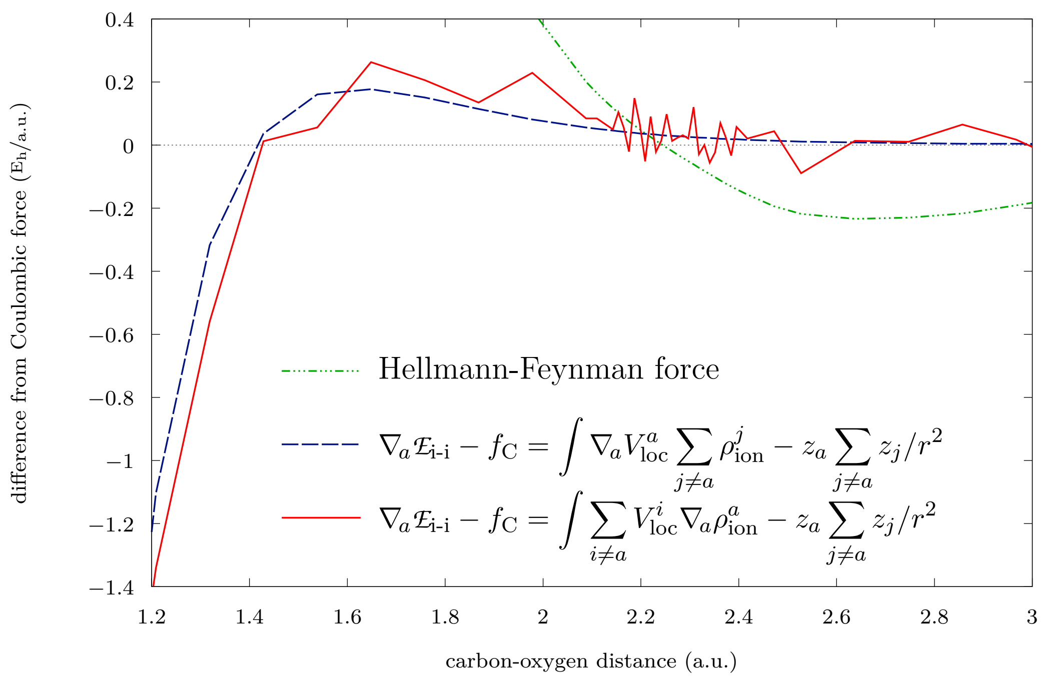


Figure 5.1: Two ways of calculating ion-ion repulsive forces on carbon (pseudopotential) cores of CO_2 molecule: Interatomic distances between the carbon atom and oxygen atoms (molecule is stretched symmetrically) are shown on the x axis and differences between the calculated force and the point-charge Coulombic force on the y axis. Note the numerical instabilities for the second method and the fact, that the core ion-ion forces differ substantially from the Coulombic force in common interatomic distances. (Stronger oscillations of the second method around the expected equilibrium distance are caused by finer sampling in this region).

would shrink to the classical Coulombic force between point charges. However, the pseudopotential smooths the charge of the nucleus and the actual force differs from the pure point-charges Coulombic term, as can be seen in Fig. 5.1. Therefore, the ion-ion force is derived by differentiating the ion-ion term of the total energy by expression proposed e.g. in [31].

Expressing the pseudocharge ρ^a_{ion} of the ion a as

$$-4\pi\rho^a_{\text{ion}} = \nabla^2 V^a_{\text{ion}} \,, \tag{5.47}$$

we can express the ion-ion energy as

$$\mathcal{E}_{\text{i-i}} = \frac{1}{2}\sum_{i\neq j}\int V^i_{\text{loc}}\rho^j_{\text{ion}} = \frac{1}{2}\sum_i \int V^i_{\text{loc}} \sum_j \rho^j_{\text{ion}} - \frac{1}{2}\sum_i \int V^i_{\text{loc}}\rho^i_{\text{ion}} \,. \tag{5.48}$$

Differentiation of $\mathcal{E}_{\text{i-i}}$ results in

$$\nabla_a \mathcal{E}_{\text{i-i}} = \frac{1}{2}\int \nabla_a V^a_{\text{loc}} \sum_{j\neq a}\rho^j_{\text{ion}} + \frac{1}{2}\int \sum_{i\neq a} V^i_{\text{loc}} \nabla_a \rho^a_{\text{ion}} \,. \tag{5.49}$$

The two terms in (5.49) express the forces corresponding to charge-potential and potential-charge interactions, respectively. The charge-potential and potential-charge forces must match, therefore (5.49) can be evaluated using just one of these terms. The second term of (5.49) suffers from much larger numerical errors than the first one since the gradient of ρ_{ion} is in fact the third derivative of V_{loc} — see (5.47) — and each differentiation introduces numerical errors into the computation, as can be seen in Fig. 5.1. That is the reason for using just the first term of (5.49) to evaluate the ion-ion force:

$$\nabla_a \mathcal{E}_{\text{i-i}} = \frac{1}{2}\int \nabla_a V^a_{\text{loc}} \sum_{j\neq a} \rho^j_{\text{ion}} + \frac{1}{2}\int \sum_{j\neq a} V^j_{\text{loc}} \nabla_a \rho^a_{\text{ion}} = \int \nabla_a V^a_{\text{loc}} \sum_{j\neq a} \rho^j_{\text{ion}} \,. \tag{5.50}$$

5.3.2 Nonlocal part of Hellmann-Feynman force

The most difficult term of the equation (5.45) is the middle one: the nonlocal part of electron-ion interaction. If we express a projections P_l to l-subspaces integrating over spheres (recall definition 4.8 on page 119) as

$$P_l\,|\psi\rangle = \sum_m Y_{l,m} \int_{\theta,\varphi} Y^+_{l,m}\psi = \sum_m |Y_{l,m}\rangle\langle Y_{l,m}|\psi\rangle \ . \tag{5.51}$$

then, denoting $V_l = V^{a,l}_{\text{nl}}$ and taking a sample wavefunction ψ, the corresponding force coming from a given l can be expressed as

$$\langle\psi|\nabla_a\left(V_l P_l\right)|\psi\rangle = \langle\psi|\left(\nabla_a V_l\right)P_l|\psi\rangle + \langle\psi|V_l\left(\nabla_a P_l\right)|\psi\rangle \ , \tag{5.52}$$

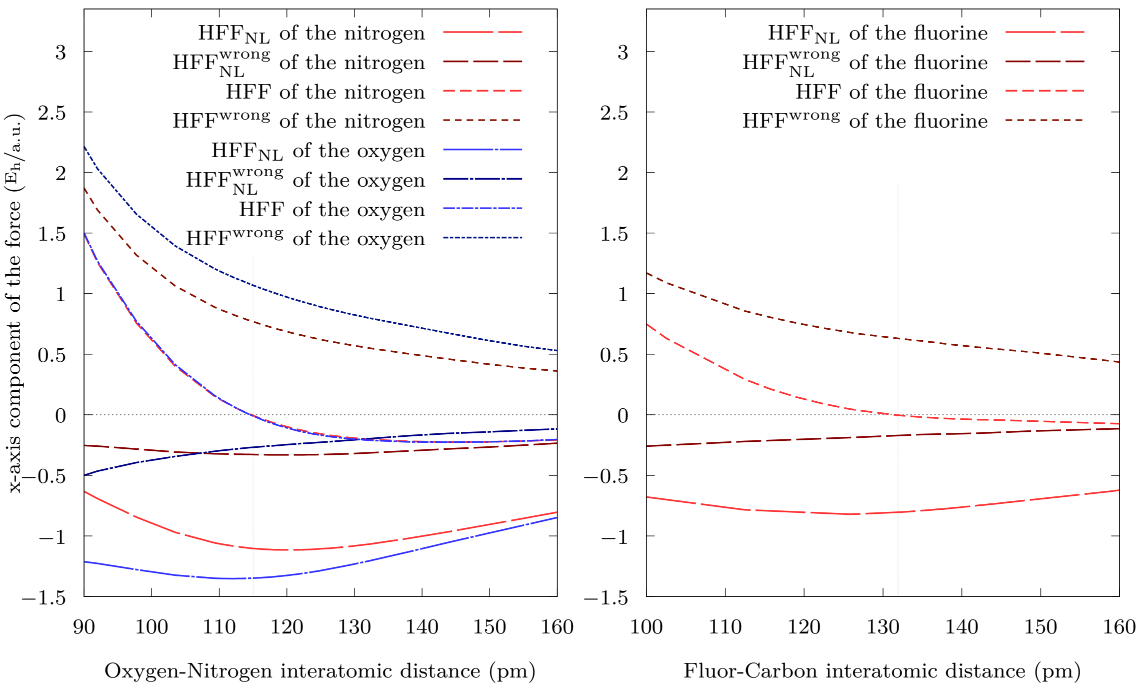
The first term in (5.52) expresses the force originating from the shift of the potential and the second term expresses the change of the charge density in the given l-subspace that occurs due to the shift of the center of $l-$projection.

While the first term of (5.52) can be evaluated by means of spherical projection relatively easily, the second one is more difficult. If we differentiate the projector P_l, we obtain the gradient of the spherical harmonic functions

$$\nabla_a P_l = \nabla_a \sum_m Y_{l,m} = \sum_m \left(|\nabla_a Y_{l,m}\rangle\langle Y_{l,m}| + |Y_{l,m}\rangle\langle\nabla_a Y_{l,m}|\right) \ , \tag{5.53}$$

with a singularity (for $l > 0$) at the origin. The approaches used so far have applied various strategies to overcome this difficulty. In the original paper describing the calculation of Hellmann-Feynman forces within DFT [2], the gradient of $Y_{l,m}$ is silently neglected. In some later works, as e.g. in Quantum Monte Carlo methods [187], where similar projections occur, authors explicitly claim that such terms can be neglected. However, in our case, this term forms a substantial part of the force and must be incorporated into the HFF calculation, as can be seen in Fig. 5.2.

Various ways have been used to avoid the necessity to calculate this problematic term in literature: the analytic derivatives of planewaves are used in [190, 191]; the Kohn-Sham equations are used in [192] to avoid the necessity to differentiate the nonlocal parts of the force, which however implies the necessity to differentiate the sum of Kohn-Sham energies. It would be neither simple nor computationally efficient in our case. Another alternative, evaluating the special integrals over atomic spheres like in the LAPW method [13], would be too computationally expensive within the general finite element basis.



$$\mathrm{HFF}_{\mathrm{NL}} = \sum_{i,l,m} w_i \left\langle \psi_i \right| \nabla_a \left(V_l P_l \right) \left| \psi_i \right\rangle \qquad \mathrm{HFF} = \nabla_a \mathcal{E}_{\mathrm{tot}} \text{ with } \nabla_a P_l \text{ included}$$

$$\mathrm{HFF}_{\mathrm{NL}}^{\mathrm{wrong}} = \sum_{i,l,m} w_i \left\langle \psi_i \right| \nabla_a \left(V_l \right) P_l \left| \psi_i \right\rangle \qquad \mathrm{HFF}^{\mathrm{wrong}} = \nabla_a \mathcal{E}_{\mathrm{tot}} \text{ with } \nabla_a P_l \text{ omitted}$$

Figure 5.2: The importance of the l-subspace projector derivatives in Hellmann-Feynman forces: Calculated HF-forces and their nonlocal components acting on the oxygen and nitrogen atoms in a nitric oxide molecule and on the fluorine atom in tetrafluoromethane, with and without the l-subspace projectors' derivatives. The expected interatomic distance (vertical dashed line) is 115pm for nitric oxide [188] and 131.91pm for CF_4 [189].

Probably the closest approach to our one has been published in [193, 194] for evaluating Hellmann-Feynman forces for the case of a separable (or Kleinmann-Bylander [160]) form of the pseudopotential $\nu_{l,n}$. That approach yields the following term for the nonlocal energy

$$\nabla_a \mathcal{E}_{\mathrm{nl}} = 2\,\mathrm{Re} \sum_i w_i \sum_{n,l,m} \Big\langle \psi_i \left| \nu_{l,n} Y_{l,m} \right\rangle \! \left\langle \nabla_a \left(\nu_{l,n} Y_{l,m} \right) \right| \psi_i \Big\rangle \,. \tag{5.54}$$

In some cases, the gradient of the projectors can be expressed analytically (e.g. see [195] or [196]), but for a general separable pseudopotential, the radial part must be differentiated numerically.

As the authors of [176] recommend, an improved precision can be achieved if the projector is decomposed using the vector spherical harmonic functions $Y_{l,m} r$ (where r is the radius):

$$\nabla \nu_{l,n} Y_{l,m} = \nabla \left(\frac{\nu_{l,n}}{r} Y_{l,m} r \right) = \left(\nabla \frac{\nu_{l,n}}{r} \right) Y_{l,m} r + \frac{\nu_{l,n}}{r} \nabla \left(Y_{l,m} r \right) \,, \tag{5.55}$$

because such decomposition makes the angular projector analytical in the origin. As we will see later in Fig. 5.8C, this approach is less suitable for the case of

l-dependent form of the pseudopotentials (5.54), because the radial part must be divided by r^2 and consequently the numerical accuracy decreases.

It seems, that RMG code [197] maybe use a method similar to our approach to avoid differentiating projectors. However, the papers describing the method, e.g. [198], cited by the RMG web page, contain just differentiating of projectors without further details of this issue.

5.3.3 Evaluating HFF using derivatives of wavefunctions

Since none of the methods mentioned above is fully satisfactory in our case, we propose another way to calculate the nonlocal terms of HF forces. Our approach is based on the simple assumption that the movement of an atomic core in any direction must result in the same force as the movement of the wavefunction (which is "frozen" due to the Hellmann-Feynman theorem) in the opposite direction. So instead of differentiating the operator V_l (as in Eq. 5.53), making use of the Hermiticity of the V_l operator we can state

$$\langle\psi|\,\nabla_a V_l\,|\psi\rangle = -\nabla_\psi\,\langle\psi|\,V_l\,|\psi\rangle = -\,\langle\nabla\psi|\,V_l\,|\psi\rangle - \langle\psi|\,V_l\,|\nabla\psi\rangle = -2\,\mathrm{Re}\,\langle\nabla\psi|\,V_l\,|\psi\rangle\ , \tag{5.56}$$

or, in the case of the separable form of a pseudopotential,

$$\nabla_a\mathcal{E}_{\mathrm{nl}} = -2\,\mathrm{Re}\sum_i w_i \sum_{n,l,m}\Big\langle\psi\,|\nu_{l,n}Y_{l,m}\rangle\langle\nu_{l,n}Y_{l,m}|\,\nabla\psi_i\Big\rangle\,. \tag{5.57}$$

Since the derivative terms of (5.56) and (5.57) are already expressed in the finite element basis, there are no extra computational demands for evaluating the projection into the basis and no additional inaccuracies arise since the analytic derivatives of the element basis functions are available. Moreover, the same expression can be easily used both for the l-dependent and separable pseudopotentials.

Note that expression 5.56 resembles a bit the incomplete basis set correction (or the Pulay term, see equation 5.42). Although the formal expressions seem to be (almost) the same, they express different things. Whereas in our case the differentiation represents a spatial shift of wavefunctions (in the fixed basis), the Pulay term means “how the wavefunction changes due to the change of the basis”. Thus the operator ∇ represent different differential operator in the two equations: since our basis does not depend on atomic positions, the Pulay term is zero in our case, but the expression (5.56) is not.

To summarize, the Hellmann-Feynman force within the finite element method (FEM) in real space can be simply expressed and evaluated as

$$\nabla_a\mathcal{E}_{\mathrm{tot}} = \int\rho\nabla_a V_{\mathrm{loc}} - 2\sum_i w_i\,\mathrm{Re}\int\nabla_\psi\psi_i^+\sum_{j,l}V_{\mathrm{nl}}^{j,l}\psi_i + \int\sum_{a\neq k}\nabla_a V_{\mathrm{loc}}^k\rho_{\mathrm{ion}}^a\,. \tag{5.58}$$

5.4 Verification and convergence analysis of the HF-forces expression

In this section, the newly developed expression will be verified by means of calculations of interatomic distances in several types of molecules and of the vibrational frequency of carbon dioxide. Moreover, the convergence of the method will be examined. Comparing the method with other approaches, the presence and the magnitude of unwanted oscillations (due to numerical errors) will be assessed. And finally, some attention will be paid to the performance of the method in comparison with other approaches to HF-force calculations.

5.4.1 Verification of the formula

For the verification of the formula, the forces in nitric oxide, carbon dioxide, and tetrafluoromethane molecules have been computed, varying the interatomic distance by scale factor β. Adaptivelly stretched hexahedral meshes (see section 3.2.1) with the base edge length of α a.u. were used for the calculations below. The distance from any atomic center to the domain boundary was at least 16 a.u.

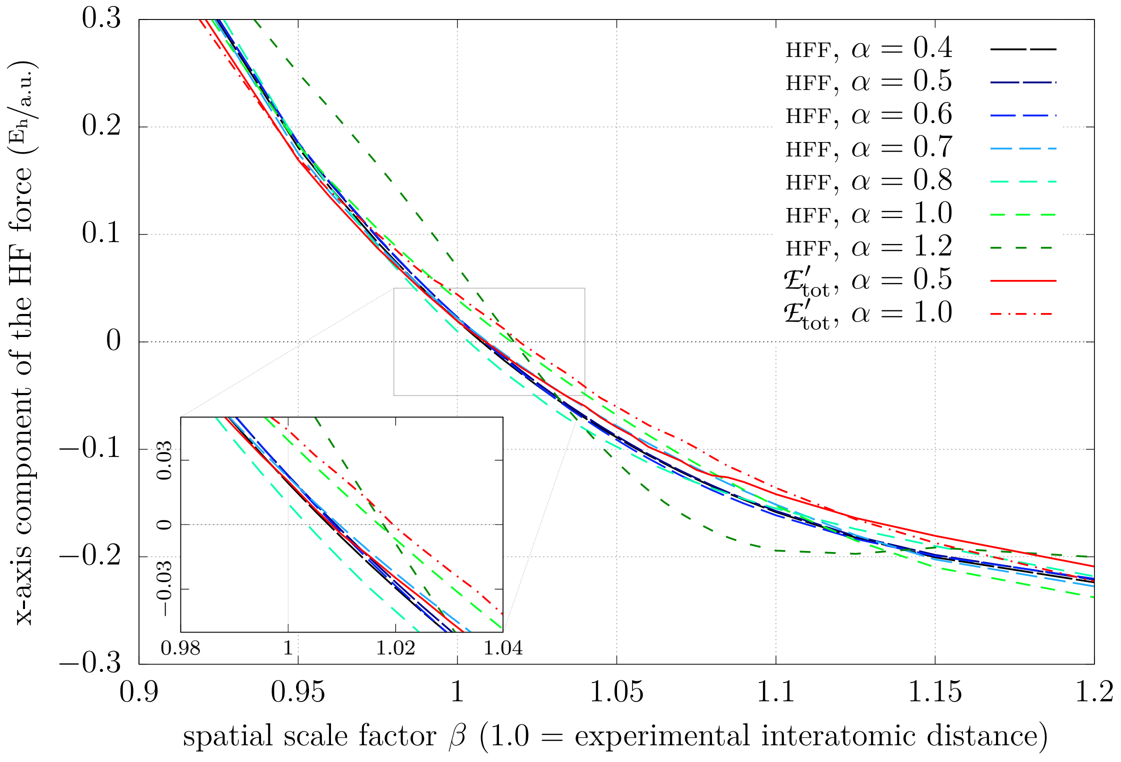


Figure 5.3: The calculated HF force acting on the first oxygen atom along the x-axis in carbon dioxide, varying with the mesh size (the parameter α is the linear size of the smallest element, in atomic units).
The red $\mathcal{E}'_{\text{tot}}$ curves correspond to HFF calculated by differentiating the total energy numerically, recalculating the wavefunctions for each atomic position (for comparison with the HF-theorem-based curves, where the wavefunctions are frozen).

The convergence of calculated HF forces in dependence on interatomic distance for the carbon dioxide molecule is shown in Fig. 5.3. The relative shift of the equilibrium interatomic distance with respect to the experimental value (116.3pm, see [188]) is 0.82%, which is comparable to other methods and is in the expectable

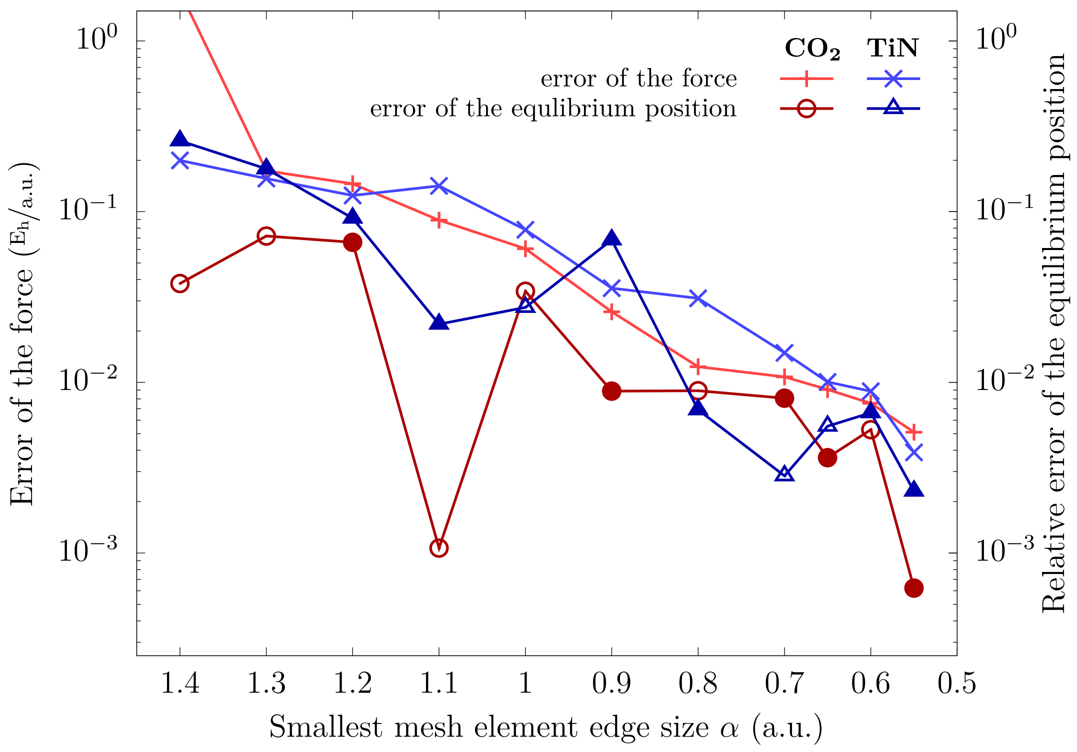


Figure 5.4: The convergence of HF interatomic force in CO_2 molecule and in diatomic TiN, with regard to the mesh element size. The reference value is the force calculated with the finest mesh ($\alpha = 0.45$). β denotes spatial scale factor for interatomic distances w.r.t. the equilibrium position. The error is computed as the L^2 norm of the function $f_\alpha(\beta) - f_{\text{ref}}(\beta)$ on interval $\langle 0.7, 1.4 \rangle$. Positive/negative errors of equilibrium positions are marked by full/empty symbols — circles or triangles. The reference value for the equilibrium distance of TiN is taken from the most precise calculation because the diatomic TiN cannot be compared with common data for crystalline TiN in this respect.

range if we consider applying DFT with a simple form of the XC-functional for a molecule.

Perhaps a bit surprising result is that the calculated HF forces are much smoother, i.e. evincing much smaller fluctuations than the numerical derivatives of calculated total energies. Since the magnitude of the fluctuations of total-energy-based calculation (via the numerical derivative) is greater than the difference between this calculation and calculated HF forces, we can conclude that the calculated HF forces agree with the numerical derivatives of total energy within the precision of our total-energy-based calculations and that our approach provides a numerically stable way for calculating HF forces.

The convergence of HF forces and the equilibrium position of atoms in CO_2 in dependence on the mesh element size is shown in Fig. 5.4. The reference values have been obtained for the finest mesh ($\alpha = 0.45$).

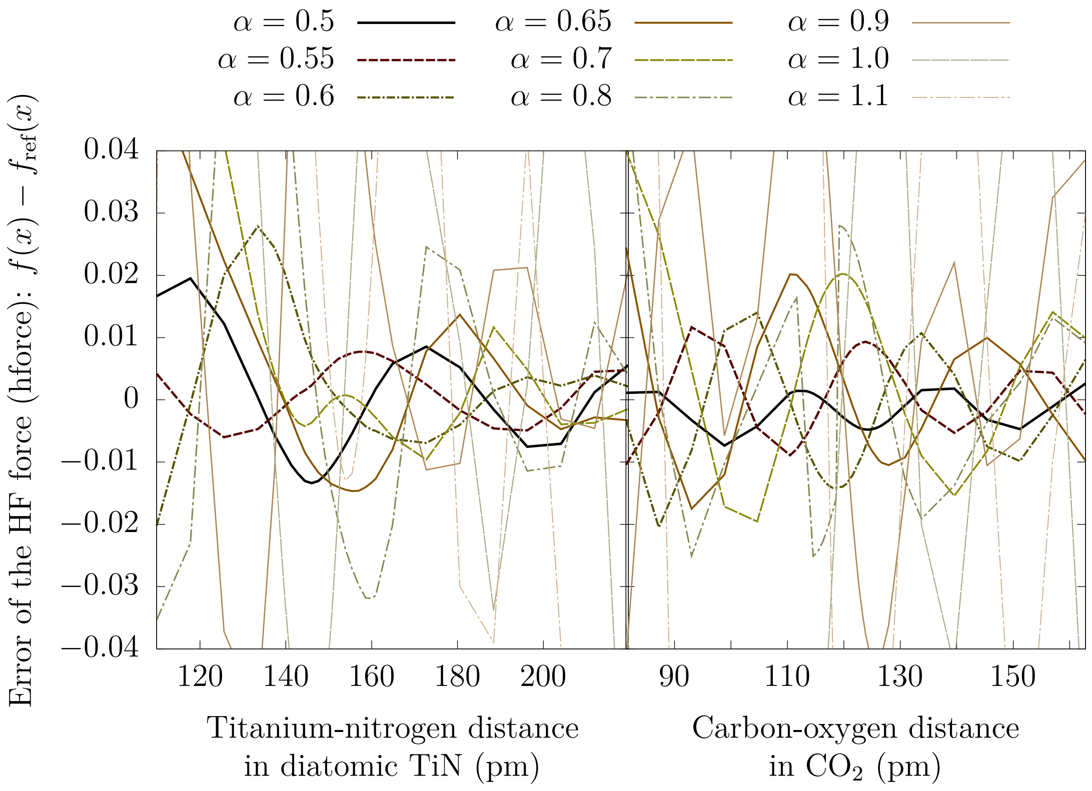


Figure 5.5: Comparison of the convergence of HF-forces for diatomic titanium nitride and carbon dioxide. The parameter α is the edge length of the smallest elements of the mesh. The reference force f_{ref} is the force calculated with the finest mesh ($\alpha = 0.45$).

The results of similar accuracy have been achieved for the cases of nitric oxide and tetrafluoromethane molecules, where the relative errors of expected interatomic distances were only 0.232% and 0.15% w.r.t. the expected values of 115 pm [188] and 131.91pm [189], respectively, and the deviation of the distance of zero HF forces compared to the minima of total energy was less than one per mille. The calculated HF forces for both molecules can be seen in Fig. 5.2 on page 148.

It should be emphasized that the experimental values above were used just as reference values for comparing the calculations mutually, without making a greater effort to obtain the best agreement with experimental values. Such an effort — including the possible use of more sophisticated exchange-correlation terms and pseudopotential tuning — would lead to even more precise and numerically more stable results at the cost of some additional computational demands.

The convergence of HF forces w.r.t. the interatomic distance in the diatomic titanium nitride has been verified, too, in order to test the convergence also for transition metals. Figs. 5.4, 5.5 show that the errors in the case of diatomic titanium nitride converge in a similar manner as in the case of carbon dioxide. The convergence with the mesh element size for TiN seems to be slower than for CO_2, but it should be noted that it always depends on the parameters of the used

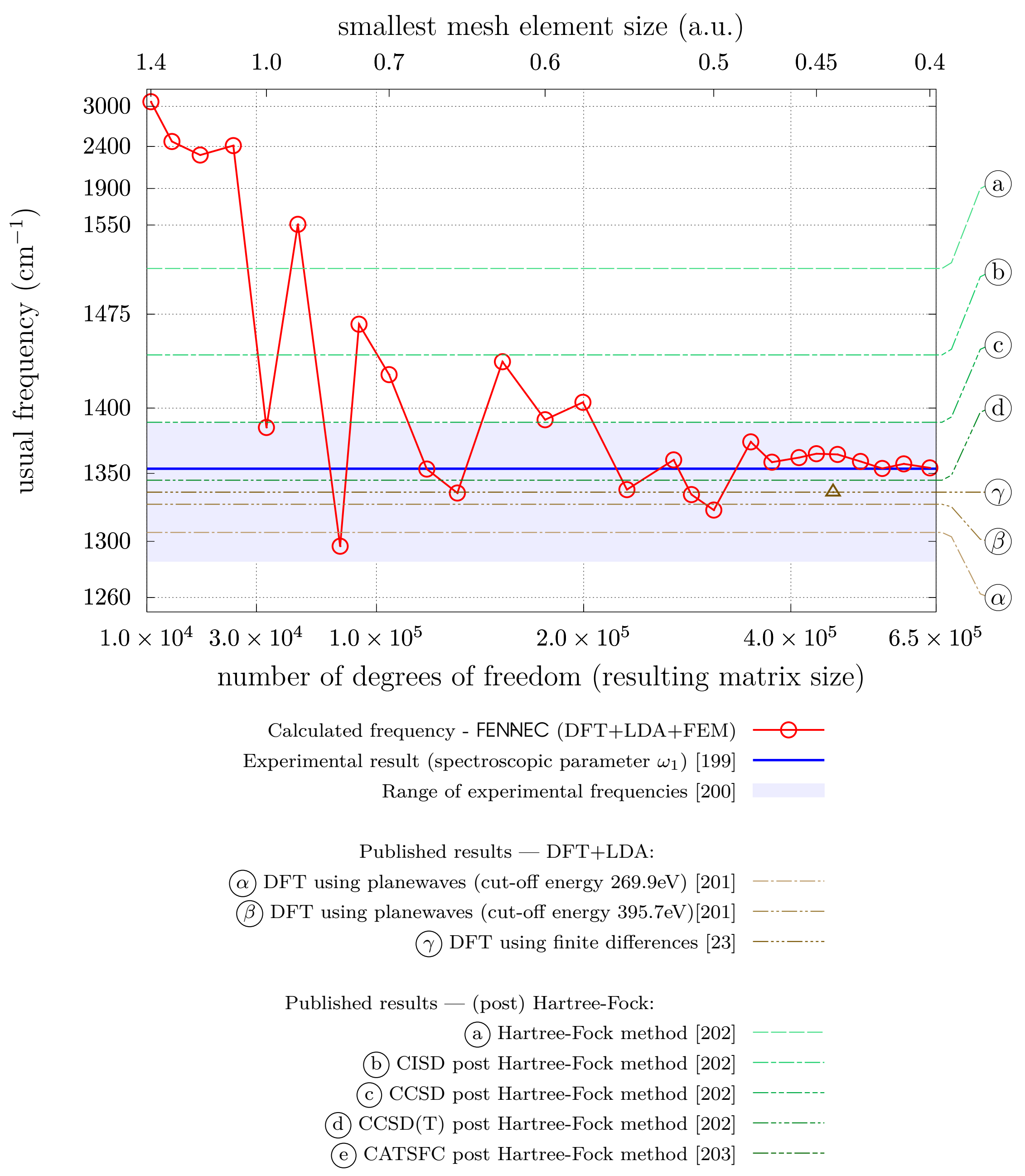


Figure 5.6: Convergence of symmetric vibrational frequencies of CO_2 molecule with the mesh element size (and with the number of degrees of freedom of the basis that is bound to the mesh element size), compared to other published experimental and calculated results. The axes are in logarithmic scale, with compressed region below 10^5 (on the x-axis) and above 1550 (on the y-axis) to cover the whole convergence curve while emphasizing the region relevant to the most accurate results. The convergence properties of the calculations done by other methods, even if the numbers of degrees of freedom are indicated in the references, can be roughly compared only in the case of finite differences (denoted by $\triangle$), because of essentially different character of bases and of resulting matrices in the other methods.

pseudopotential — softer and less transferable pseudopotentials generally converge faster.

For a stricter verification of our code and of its convergence properties, the vibrational frequencies of the symmetric mode (often called v_1, see [204]) of carbon dioxide molecule have been calculated. Since the vibrational frequency is in fact a derivative of HF force (and the second derivative of the total energy), it is usually much more sensitive to the computational errors than the equilibrium position and, therefore, it can be used as a more stringent benchmark for the precision of the method.

The vibrational frequencies have been calculated from the force acting on carbon atom displaced by 0.005 a.u., using a simple model of a linear harmonic oscillator. Our results match the data obtained from the experiment [199] quite well and document good convergence properties of newly proposed formula — see Fig. 5.6.

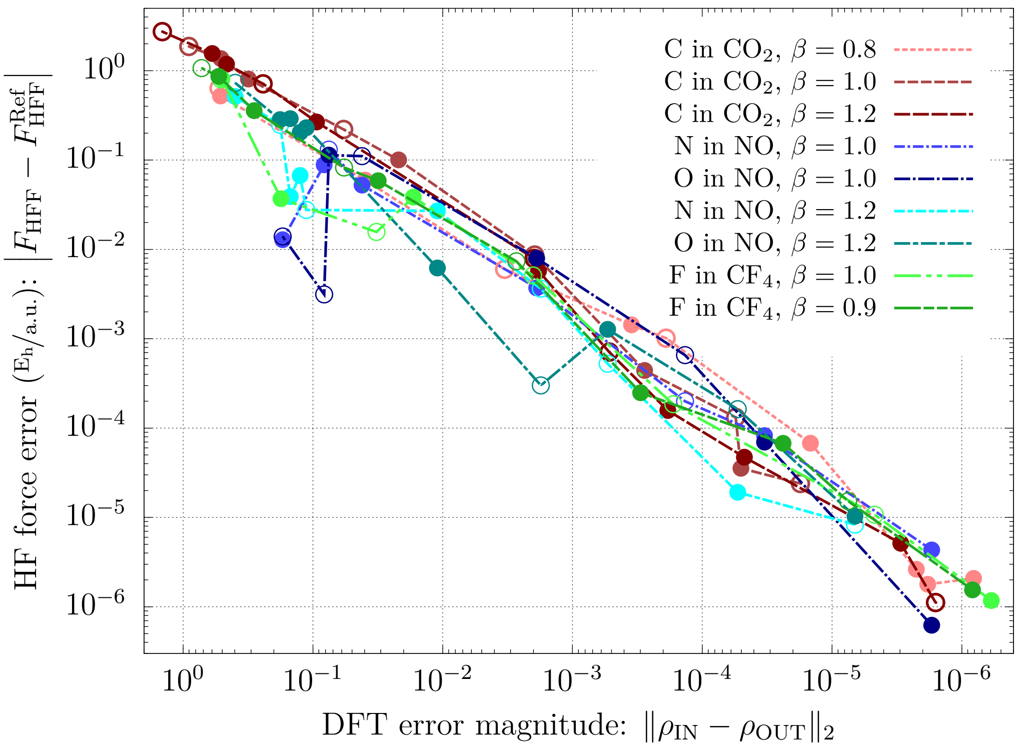


Figure 5.7: Convergence of HF forces with respect to the stopping criterion of the DFT loop. The DFT error magnitude is computed as L^2 norm of the difference between the input and output charge densities of each DFT iteration. The HF forces from the last — fully converged (DFT error magnitude $< 10^{-7}$) — step of the DFT loop serve as the reference forces $F_{\mathrm{HFF}}^{\mathrm{Ref}}$. Empty/full points denote negative/positive differences with respect to the reference values. β is the spatial scale factor (stretching the molecule). The Anderson/Pulay mixing scheme [82, 83] was used in the DFT loop to achieve the self-consistent state.

I also tested the dependence of the HFF error (compared to reference values obtained from self-consistent states) on the degree of convergence of the DFT loop (i.e. on the precision of the solution of Kohn-Sham equations). The calculations show a nearly linear dependence of the size of the HFF error on the L^2 norm of the DFT error, see Fig. 5.7, regardless of the system and/or spatial stretching of the system. This dependence can be used as a reasonable estimate of the error of the HFF calculation. It can be also used for determining the stopping criterion for the DFT loop, e.g. in geometry optimizations of more complex systems.

It should be noted that the experimental values above are used just as reference values for comparing the calculations mutually. The differences of calculated quantities (like the equilibrium distance) from the experimental values are not significant in this context, because they originate particularly in using a simple form of the exchange-correlation potential, in some fundamental limitations of applying the local density approximation for molecules and in limited transferability of the used pseudopotentials (we used generic pseudopotentials not specifically tuned for the charge distribution in the molecule).

5.4.2 Convergence and fluctuations of various methods for computing HF forces

A series of calculations to compare the errors and convergence rates of six methods for evaluating the nonlocal components of Hellmann-Feynman forces is shown in Fig. 5.8. Using the series of meshes with varying base element edge length α (described in the previous section), the results of various methods have been compared and assessed in comparison with reference values obtained for the finest mesh ($\alpha = 0.7$). We can see that all the methods converge to the same solution. This solution is smooth, stable, and matches the derivative of the total energy (as verified in the previous section).

Four of the compared methods use separable pseudopotentials (A, B, C, E in Fig. 5.8) and two of them use l-dependent pseudopotentials (D, F). The methods A to D express HF-forces directly using the HF theorem (5.39), where the methods A and D use the approach proposed in this thesis, i.e. differentiating of the wavefunctions (for separable and l-dependent pseudopotential, respectively), whereas B and C differentiate the projectors of the separable pseudopotentials — in which case the singularity of the analytical gradient of the separable pseudopotential is treated by projecting into the FEM basis (B) or using the vector spherical harmonics for $l > 0$ (C, see (5.55)).

On the other hand, methods E and F use the numerical differentiation of the total energy in combination with separable (E) and l-dependent (F) pseudopotentials. In both cases, wavefunctions — according to the HF theorem (5.39) — remain "frozen" during differentiating.

We do not consider the numerical differentiation of the total energy with non-frozen wavefunctions (i.e. without employing the HF theorem) in this section. The need for recalculating wavefunctions for various atomic positions made such methods uncompetitive as regards both efficiency and accuracy: computational demands and the inaccuracies of wavefunctions (numerical as well as arising from non-fully converged DFT-loop or incompleteness of the FEM basis, see Fig. 5.3) practically disqualify this approach.

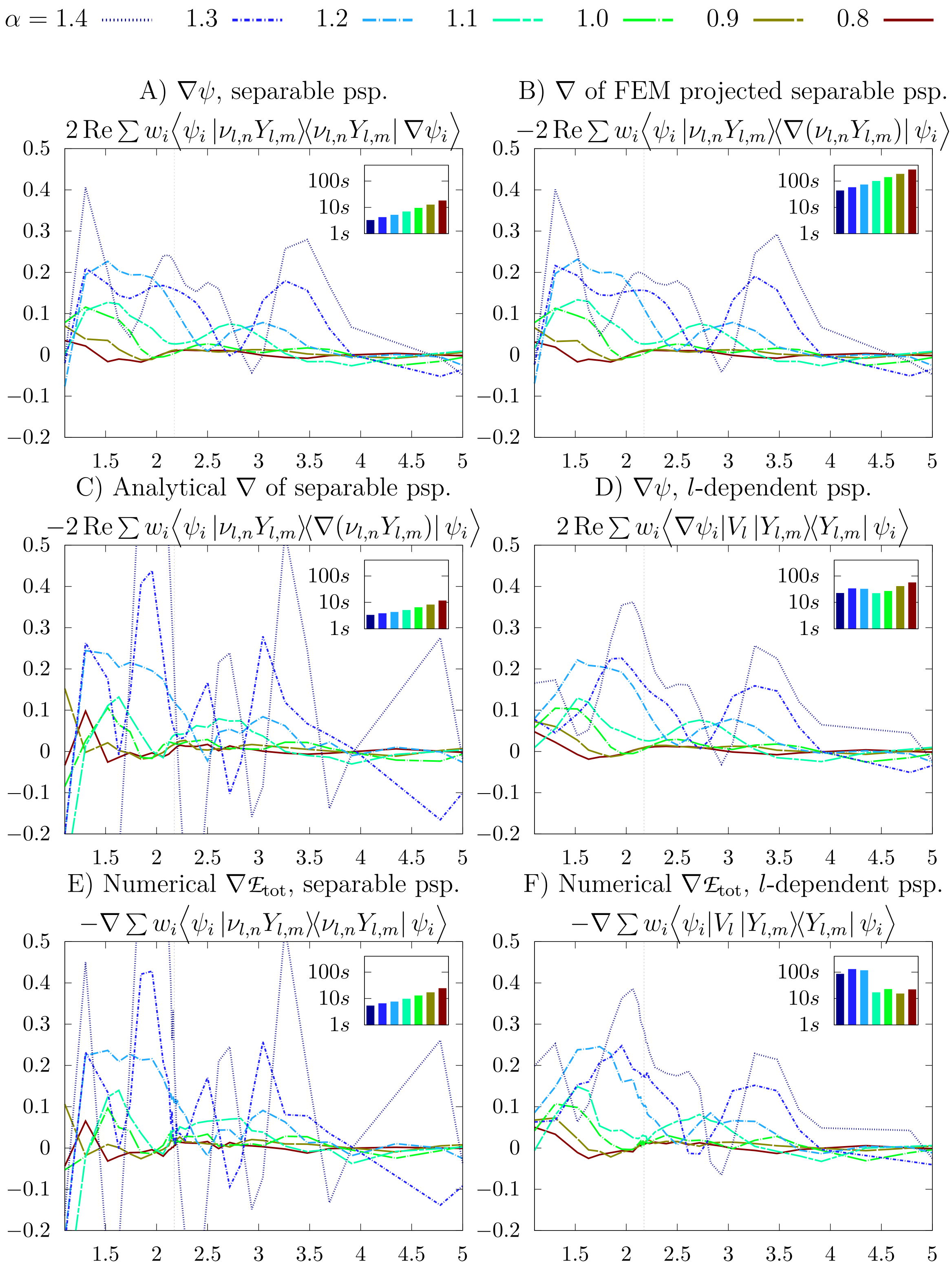


Figure 5.8: Convergence of the HF-force acting on N atom in NO molecule with the mesh element size (the parameter α is the linear size of the smallest element, in atomic units) for various approaches. The curves show the numerical error of HFF (in $E_h/\mathrm{a.u.}$) related to the reference force obtained using the finest mesh. The x axes show the interatomic distance (in a.u.), with equilibrium marked by the vertical dotted line. Small subgraphs show the computational time on a logarithmic scale.

The Fig. 5.8 is complemented by Tab. 5.1 showing the asymptotic complexity of operations of the considered methods in terms of the discretization parameters.

5.4.3 Discussion of the methods

At this moment, we can assess the accuracy of the considered methods, taking the computational demands into account. Note that the oscillations — that can create false local minima or spoil the convergence of the minimization algorithm — might interfere with the accuracy of the whole computation more than a constant error that only shifts the equilibrium position a bit. From this point of view, a worse precision has been achieved using the expressions that differentiate the separable pseudopotential, either analytically or numerically (see Fig. 5.8 C and E), compared to those using the gradient of wavefunctions (A and D).

From the anti-hermiticity of the bilinear form $(* \cdot \nabla *)$ it follows

$$\operatorname{Re}\left(\nu_{l,n} Y_{l,m} \cdot \nabla \psi_i\right) = -\operatorname{Re}\left(\nabla \nu_{l,n} Y_{l,m} \cdot \psi_i\right) \tag{5.59}$$

and so, that the calculation using the gradients of the projectors (Fig. 5.8 C) is mathematically equivalent to our present method (differentiating ψ, Fig. 5.8 A). The two cases differ just by numerical errors caused by differentiating: while using derivatives of a separable pseudopotential leads to integrating previously differentiated projectors of the separable pseudopotential, our new method allows us to do both steps — differentiating and integrating — "at once" during the finite-element basis assembling. Moreover, the basis functions have polynomial derivatives that are more suitable for the numerical quadrature [6] used in the FE assembling than the derivatives of projector functions of separable pseudopotential. Thus, it is not so surprising that our present method results in smaller numerical fluctuations, as can be seen comparing Fig. 5.8 A and Fig. 5.8 C.

The poor numerical precision of differentiating separable pseudopotentials (Fig. 5.8 C and E) can be improved by projecting the pseudopotentials into the FEM basis prior to differentiating, as in (Fig. 5.8 B). This approach eliminates both sources of errors noted in the previous paragraph. Since the pseudopotentials can be projected into the FEM basis quite precisely, the resulting error is nearly the same as in our present method (Fig. 5.8 A). However, the projections of pseudopotentials into the FEM basis are computationally expensive, whereas the projections of wavefunctions have been already prepared. Therefore, our newly proposed method provides a significant improvement in computational efficiency compared to that approach.

The results obtained using l-dependent potentials (Fig. 5.8 D and F) do not suffer from the inaccuracy of the gradient evaluation, but they have another disadvantage: evaluation of the spherical integrals (we use Lebedev quadrature for integration on spheres, see [205]) is not a natural approach for the finite element method, therefore its computational expenses do not depend on the mesh size. Or even worse: for smaller meshes with larger elements, the projection from the real space to the reference element, which is necessary for evaluating the integral, can be more demanding than for finer meshes.[2] This behavior — that may look strange at first sight — is caused by the fact that the same number of integration points for the radial integration is needed for a fine and for a coarse mesh. Solving the L^2 projection on larger elements of the coarse mesh can require more computational time to achieve the same accuracy of interpolation than solving the same problem on a finer mesh. This behavior is very disagreeable because it interferes with using a coarse mesh as a fast and cheap preprocessing for obtaining a reasonable guess at initial positions of atoms.

In some cases, the numerical derivative of the total energy using the l-dependent pseudopotentials (Fig. 5.8 F) provides better computational efficiency than the newly proposed method, which may look strange. However, none of the methods was implemented with all possible optimizations. In addition, when using a particular HFF evaluation method in combination with a particular electronic-structure code, the efficiency of the HFF evaluation depends very much on which quantities have been already precalculated in the code for other purposes. Therefore the presented computational times are just indicative and they can rather serve to illustrate how the computational time scales with the system size than to compare the methods mutually. The computational expenses for evaluating the matrix elements of separable pseudopotentials can be substantially reduced by taking the full advantage of the short-range nature of the non-local pseudopotential components, evaluating the integrals only on the mesh elements within the spatial scope of the nonlocal part of each pseudopotential; in that case, n_e in Tab. 5.1 would correspond just to the number of elements within the nonlocal pseudopotential component support. It could reduce the computational expenses for evaluating the integrals substantially (as the experience with similar techniques from other codes, e.g. [197, 17], confirms), whereas that advantage — i.e. evaluating just on the short-range radial mesh — have been already employed in the case of l-dependent pseudopotentials.

Moreover, the l-dependent pseudopotentials provide worse precision near the equilibrium position according to our test calculations, and the numerical derivatives of the total energy suffer from much larger fluctuations (due to the inaccuracies of the numerical differentiation), i.e. they are not so smooth, compared to the forces obtained by differentiating the wavefunctions, as it can be seen in Fig. 5.8 near the equilibrium position (where the forces were calculated for interatomic distances varying with a fine step).

Therefore, we conclude that the method A — i.e. differentiating the wavefunctions in combination with separable pseudopotentials — seems to be the most suitable method for applications within FEM combined with the non-local potentials. The proposed approach provides both sufficient numerical accuracy and computational efficiency and outperforms the other methods considered within the scope of this study.

[2]Our additional numerical tests show a strong dependence of the computational demands on the geometry of the used mesh.

Symbol	**Meaning**
n_ψ	number of ψ_i
n_{P}	number of projectors of separable pseudopotential of the atom
n_{lm}	number of l, m components of l-dependent pseudopotential
n_{e}	number of elements of the FEM mesh (function of the α in the graphs above)
n_{qp}	number of quadrature points in each element
n_{bf}	number of basis functions in a single finite element
n_{fd}	number of points used for numerically differentiating e_{tot}
n_{rp}	number of points in the radial mesh for the projection of the wavefunctions into the L-space
n_{ap}	number of points in the angular mesh for the projection of the wavefunctions into the L-space
Θ_{P}	L^2 asymptotic complexity of the projection into the FEM basis
Θ_{E}	asymptotic complexity of evaluating a quantity in the FEM basis in a given point

Method	**Asymptotic computational time complexity**
A	$n_\psi n_{\text{P}} n_{\text{e}} n_{\text{bf}} + n_{\text{P}} n_{\text{e}} n_{\text{qp}} n_{\text{bf}}$
B	$n_\psi n_{\text{P}} n_{\text{e}} n_{\text{bf}} + n_{\text{P}} n_{\text{e}} n_{\text{qp}} n_{\text{bf}} + n_{\text{P}} \Theta_{\text{P}}$
C	$n_\psi n_{\text{P}} n_{\text{e}} n_{\text{bf}} + n_{\text{P}} n_{\text{e}} n_{\text{qp}} n_{\text{bf}}$
D	$n_\psi n_{lm} n_{\text{e}} n_{\text{qp}} + n_\psi n_{lm} n_{\text{rp}} n_{\text{ap}} \Theta_{\text{E}}$
E	$n_{\text{fd}} n_\psi n_{\text{P}} n_{\text{e}} n_{\text{bf}}$
F	$n_{\text{fd}} n_\psi n_{lm} n_{\text{e}} n_{\text{qp}} + n_{\text{fd}} n_\psi n_{lm} n_{\text{rp}} n_{\text{ap}} \Theta_{\text{E}}$

Table 5.1: The asymptotic complexity of the methods considered in Fig. 5.8 for computing the non-local part of the HF-forces acting on one atom, in terms of the discretization parameters.
Both Θ_E and Θ_P cannot be easily expressed using the quantities above, bacause they depend in a complicated way on the mesh geometry (especially the dependence of Θ_E on n_{e} is strongly non-monotonic for various meshes covering the same spatial volume, as can be seen in Fig. 5.8 D and F). The short range character of nonlocal pseudopotential components is not used in the expressions above; by employing that feature, n_{e} would correspond just to the number of elements within the nonlocal pseudopotential component support.

5.4.4 Notes on efficiency with respect to other codes

The accuracy of the calculations increases with decreasing the parameter α (the smallest mesh edge length), while the problem size in terms of the number of degrees of freedom (DoF's) increases as well, proportionally to $1/\alpha^3$. More DoF's usually lead to higher computational demands, but, in the FEM basis, the efficiency is affected not only by the number of DoF's but also by the number of non-zero elements in the Hamiltonian matrix (see section 3.4.2), by the number of the rank-k-updates, by the computational demands of projecting quantities to the basis and by computational demands for finding the reference element coordinates of points in general positions (the last factor can be hardly expressed in terms of the DoF's number because for large elements the solutions are often not faster than for a larger number of smaller elements).

Therefore — with respect to the more general aims of the comparison — the details too specific for the particular basis types should be disregarded. In addition, seen from the viewpoint of the overall computational time, the HFF evaluation is not the bottleneck for most codes, including FENNEC (the HFF evaluation takes a few percent of the total elapsed time: below 1% for the largest of the tasks solved above, and it diminishes progressively with increasing problem size). For the comparison of HFF evaluation algorithms, their precision, numerical stability, and the absence of oscillations are much more relevant factors, and these are primarily demonstrated in this section of the thesis.

6. FENNEC software package

Contents

This chapter describes the implementation of the proposed method within the software package FENNEC, the development of which has been a primary aim of the doctoral study associated with this thesis.

The aim of this chapter is both to introduce the FENNEC software package, and to share experience with the used programming languages, libraries, solvers etc..., obtained during the development of the package. This text is far from providing a full description of FENNEC; still, I hope that the following brief introduction of the framework can demonstrate the complexity of the package and the large veriety of problems to be solved during the development. This chapter is also intended to serve as a brief introduction for potential new FENNEC users and/or developers.

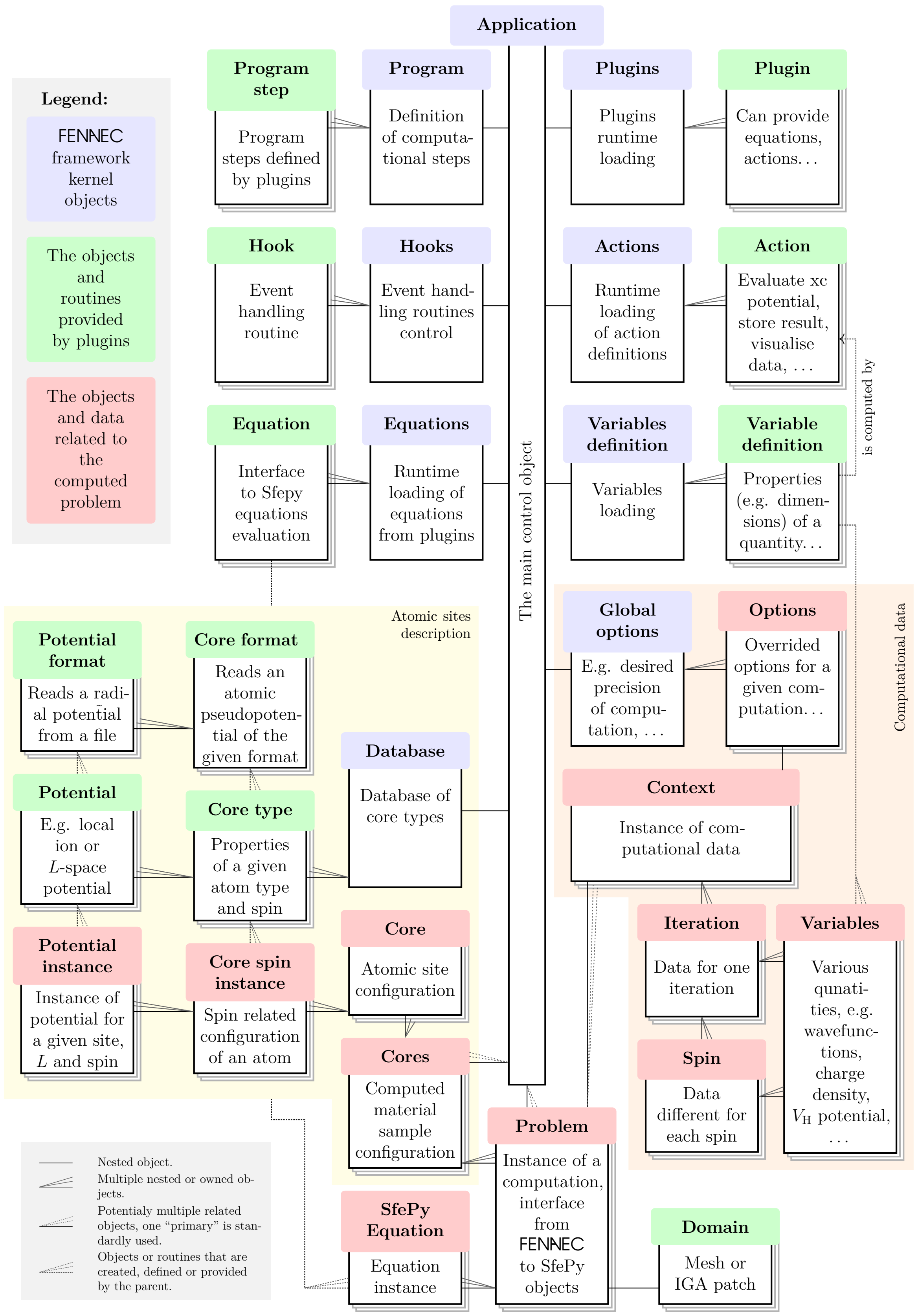
In the first section of this chapter, the architecture of FENNEC is described. The second section discusses the programming languages, libraries, and parallelization techniques used in FENNEC. The third section is focused on the linear algebra solvers that are crucial to the FENNEC performance. The short last section describes the current work in progress and the future plans for further development and extensions of FENNEC.

6.1 Description of the FENNEC software package

The FENNEC software package is implemented in Python[1] programming language, which allows fast prototyping of software. Crucial performance routines of FENNEC are written in Cython [206] or Fortran. The framework SfePy[207], developed and maintained at the University of West Bohemia, is used for finite element discretization and assembling.

[1]http://www.python.org

Figure 6.1: Architecture of FENNEC package – relations of main entities in the package.


Application
Legend:
FENNEC framework kernel objects
The objects and routines provided by plugins
The objects and data related to the computed problem
Program step
Program steps defined by plugins
Program
Definition of computational steps
Plugins
Plugins runtime loading
Plugin
Can provide equations, actions...
Hook
Event handling routine
Hooks
Event handling routines control
Actions
Runtime loading of action definitions
Action
Evaluate xc potential, store result, visualise data, ...
is computed by
Equation
Interface to Sfepy equations evaluation
Equations
Runtime loading of equations from plugins
Variables definition
Variables loading
Variable definition
Properties (e.g. dimensions) of a quantity...
The main control object
Atomic sites description
Potential format
Reads a radial potential from a file
Core format
Reads an atomic pseudopotential of the given format
Database
Database of core types
Potential
E.g. local ion or L-space potential
Core type
Properties of a given atom type and spin
Potential instance
Instance of potential for a given site, L and spin
Core spin instance
Spin related configuration of an atom
Core
Atomic site configuration
Cores
Computed material sample configuration
Global options
E.g. desired precision of computation, ...
Options
Overrided options for a given computation...
Computational data
Context
Instance of computational data
Iteration
Data for one iteration
Variables
Various quantities, e.g. wavefunctions, charge density, V_H potential, ...
Spin
Data different for each spin
Problem
Instance of a computation, interface from FENNEC to SfePy objects
SfePy Equation
Equation instance
Domain
Mesh or IGA patch
Nested object.
Multiple nested or owned objects.
Potentialy multiple related objects, one "primary" is standardly used.
Objects or routines that are created, defined or provided by the parent.

The architecture of FENNEC consists of the framework kernel and plugins. The relatively small kernel provides primary services for the plugins, their loading and communications. All the key routines for electronic structure calculations are implemented in plugins. That architecture makes possible to customize easily the program flow (e.g. to add preprocessing and postprocessing, to perform the geometry optimization instead of plain DFT loop iteration etc.). Also it makes easy to replace any code segment by an alternative implementation (e.g. to incoporate a new eigenvalue solver or a new XC potential, etc.) and it makes the package on the whole highly configurable, customizable and expandable. A larger discussion of the advantages and disadvantages of the proposed architecture will be discussed below, after the introduction to the program code.

6.1.1 Basic description of the FENNEC architeture

The basic scheme of the essential parts of the FENNEC architecture can be seen in Fig. 6.1. For better understanding of that scheme, a short description of the objects of the FENNEC framework is provided on the following pages.

As we can notice on Fig. 6.1, three are three types of objects in FENNEC. The first and the most important are the plugins and the objects that plugins provide: as e.g. *Actions* or event handlers called *Hooks*. Special types of plugins can also define the types of atomic sites, create a mesh for a computation, etc.

The second group of objects consists of all objects containing computational data, environment settings and the configuration of atomic sites.

The objects in the third group are FENNEC kernel objects — i.e. the objects that provide the basic functionality: loading the program, plugins, and libraries and setting the environment. These objects also mediate interactions among plugins.

Plugins and the entities provided by them

FENNEC plugins are small classes[2], that provide the functionality needed for DFT calculations (e.g. solvers, smearing routines), definitions of pseudopotentials and/or the geometry data of the calculated sample, and the data that control the flow of the computation. The plugins can be loaded either from the command line (to specify the atomic structure and/or other problem characteristics, or to configure the program flow) or from other plugins (if some particular functionality is needed).

To be as independent as possible, the plugins do not typically communicate directly with each other. They just provide an implementation of a particular functionality (as, e.g., the routine for smearing) or a description of an entity (e.g. a format of a pseudopotential) that can be requested by other plugins via kernel objects. This indirect communication allows us to replace easily the parts of the functionality for the desired implementation (e.g. to change the XC potential, to add an external potential etc.).

Among the entities provided by the plugins, it belongs:

Action The main tool for cooperating among plugins are actions: the routines that are callable by other plugins. Each action should implement just one

[2]In the sense of object-oriented programming, see e.g. [208]

task: all the needed subtasks should be performed by calling the proper *Actions* from other plugins.

For example (with a little simplification), a plugin that implements the DFT loop takes care only of the DFT loop, and it contains three tasks: (1) initialization of the loop (e.g. obtaining the initial charge density), (2) calculation of the output charge density from the input one, and (3) mixing the output charge density into a new input one.

All these task are provided by other plugins via action. Thus, the DFT loop plugin consists just of few lines: first it uses an *Action* `dft_mixer` to obtain the object that perform the actual mixing, then it prepares a DFT cycle using an *Action* `dft_prepare_iteration` that initializes the charge density and finally it uses the mixer on the *Action* `dft_iteration`. After each iteration, the DFT loop plugin checks the `dft_convergence` *Variable* (provided by an other plugin), whether the charge density is converged sufficiently or not.

We can notice that the DFT loop itself does not know at all, what it actually iterates — it can iterate both the potential or the charge density (see potential mixing on page 2.5.2). Or better, it is even not allowed to know, since this "knowledge" would introduce unnecessary dependency and therefore it would break the effort *to be as isolated as it can be* plugin paradigm.

Variable (definition) The concept of plugins and actions takes into account that many quantities are needed more than once during calculation: e.g. the Hartree potential is needed both for evaluating the Kohn-Sham matrix and for the total energy calculations.

The decomposition into plugins, and the fact that the flow of the computation is not fixed, require a mechanism for specifying which quantities should be stored for later use and how long they should be stored. For that purpose, the concept of variables has been introduced: each plugin can state that there is a quantity of a given name and type (e.g. scalar number, function on the volume, or assembled matrix). The definition of a variable is done by creating and registering an "example variable". The actual *Variable* that holds data for a particular calculations is created by cloning the example one.

The definition can further specify whether the quantity is spin-dependent (its value differs for each spin) or not, it can contain default value of the variable (either fixed one or lazy-evaluated by a function: e.g. the sum of the isolated-atom charge densities of the atoms in the sample is the default value of the charge density) and it is usually connected with an *Action* that computes the quantity when it is needed.

A plugin that defines the variable also prescribes, whether the variable remains unchanged during iterations (e.g. atomic core potentials), or if it has its scope limited to one iteration. If the latter case holds, the variable is automatically cleared at the start of the new iteration to retain memory efficiency. However, any plugin can ask a variable to store one or more its previous values (e.g. Kohn-Sham equation solver requests from the wave functions variable to retain its value from the previous iteration, since it uses them as a first-guess for underlying eigenvalue solver).

The concept of the modification of variable properties by another plugin is used more times in FENNEC. Another example could be the default value of the charge density again: whereas one plugin defines the charge density variable, its default value (superposition of atomic densities) is defined by a different plugin. If the user needed, he could create and load his own implementation of the default value, which overrides the default one.

Hook Another method how plugins can react to the activity performed by other plugins are *Hooks*. They can be viewed as event handlers: any plugin can register a handling routine — *Hook* — for an event with a given name. Any plugin can trigger an event anytime: then all the corresponding *Hooks* are called with the arguments provided by the triggering plugin.

This mechanism is used e.g. for postprocessing of DFT iteration: at the end of each iteration the `dft_iteration_end` *Hook* is triggered. The hook can be and is handled by many plugins: the summary of the results gained so far can be displayed, HF-forces or total energy can be evaluated, the current state of the computation can be saved, etc.

Program step The flow of the computation by FENNEC is not fixed. Instead, each *Plugin* can define tasks that should be performed.

Usually just one "program plugin" is loaded, which defines the main objective of the computation (e.g. whether to do single DFT cycle or to optimize atomic geometry), while other plugins can modify the flow of computation: e.g. they can add initialization of some data (e.g. the plugin can determine the geometry of atomic sites), some preprocessing (e.g. to compute initial atom density for better convergence of the main DFT loop) or postprocessing (e.g. visualization of results).

(Atomic) Core type FENNEC needs to know the properties of atoms in the structure under study. Each atom has its type: e.g. the carbon on the surface, the carbon in the bulk, the oxygen ion, etc. These types are described by *Core type* object, which is created by *Core format* according to the definition (e.g. given XML file). The *Core type* does not take care of spin: Spin-aware *Core format* should create one *Core type* object for each spin and link them together. This approach allows using both spin-aware and spin-less potentials in the same manner without increasing the complexity of the FENNEC class structure.

Potential Each *Core type* contains (besides other properties) a pseudopotential for each described L-subspace, and local and unscreen potential (the XC-term associated with the valence charge density subtracted for "unscreening"), shared by all L-spaces: each of them is described by a *Potential* object. Unscreen potential is required only in the case of a L-dependent pseudopotential when it is used to generate a separable form of the potential.

Potential format FENNEC can use various types of pseudopotentials. New potential formats can be defined by plugins. If they are placed in the proper directory, they are automatically loaded and used, if needed.

The pseudopotential format may emerge in two forms: The simpler one consists of the radial potentials that together form the pseudopotential (local ion potential, unscreen, and the potentials for each described L) in separate files. In such a case, it is sufficient to create a new *Potential format* with just one method for reading the potential from a file. The filenames, where the potentials should be read from, are given by a definition of properties of an actual type of atom (see later), which allows combining even potentials of different formats.

Core format If the potential format is more complex — e.g. a XML file that contains all the individual atomic L-potentials — a new *Core format* can be created. Core format should employ *Potential formats* (e.g. dedicating them a particular item in the XML file) for creating the *Potentials* objects and then combine them to one *Core type* object.

Equation Any plugin can define equations: prescriptions for SfePy framework how to assemble matrices and vectors: how to discretize a given equation using finite element (or IGA). For the details of SfePy configurating, see its well-done tutorial[3]. Equations (similarly as all other parts of SfePy configurations) can be defined in plugins in the SfePy config file. This property allows very easy use of any SfePy functionality to FENNEC developer.

Tests Each good software should be covered by tests. The tests in FENNEC are implemented as special plugins[4] They mimic most of the behavior of well-established test frameworks as PyUnit[5], making available all the functionality of FENNEC framework in the tests. This makes very simple to write the tests, especially the integration ones.

E.g. testing translation symmetry of a computation (except for border effects) can be written on a few lines — see the following example pseudocode:

```
Create_cube_mesh
Add_an_atom_to_the_centre
Run_dft_cycle
Move_the_atom_slightly
Run_dft_cycle
Compare_the_results
```

where each line is either a calling of an action, or calling a function provided by the FENNEC kernel. A test writer can also use a parallelization easily: e.g. to run the `Run_dft_cycle` in the previous example in parallel mode to make the tests faster.

[3] `http://sfepy.org/doc-devel/tutorial.html`

[4] They are not shown on the scheme Fig. 6.1, for the sake of readability of the scheme.

[5] `http://pyunit.sourceforge.net/pyunit.html`

Computation related objects

FENNEC is not restricted to one computation at a given time. E.g. for the purpose of multigrid computation, two computation can be run “simultaneously” and transfer the results there and back. This property is also useful for loading the instance of another computation during a computation, e.g. for obtaining initial charge density. To be able to work with more instances of computation at a time, FENNEC has to hold computational separately from plugins in their own data structures.

The objects associated to a particular calculation can be further subdivided into three groups: (1) objects that hold the configuration of atomic sites (geometry, potentials), (2) objects that provide actual implementations of finite element methods routines (e.g. assembling of the matrices) for a given mesh and (3) objects that store the computational data (charge density etc...).

Structures that define the atomic sites configuration As can be seen in Fig. 6.1 (yellow part), there are three main types of atomic-site-related objects: formats (that describe how a pseudopotential is stored), types (pseudopotentials themselves), and instances (declarations: “an atomic site placed here has this pseudopotential”). Formats and types have been discussed above; being defined framework-wide, only actual instances of atomic sites can vary for each calculations.

Cores The root object for storing informations about the studied structure. It holds one or more *Cores*.

Core This object contains all the data related to a particular atomic site: position, number of valence electrons and possible constraints for geometry optimizations...

The constraints can be easily specified in various coordinates, e.g., for axially symmetrical material, the radial distance of an atom in cylindrical coordinates can be optimized, letting its angular coordinate fixed, and reflect the changes of the radial coordinate of this “reference” atom to the other symmetrical atoms.

Core spin instance To hide the differences among calculations with and without spin as much as possible, each core owns separate “property object” for each spin. If it is a spin-aware core type, it owns two of such objects; otherwise, it owns one that serves for both spins. This design allows both to use spinless cores in spin-aware (SDFT) computation and vice versa.

Default properties of a *Core spin instance* are provided by a *Core type* (e.g. local potential of the core), however, one can modify them if it is needed.

Potential instance This object describe a potential of a *Core spin instance* for a given L-space. Just as *Core spin instance*, *Potential instance* inherits defaults properties from *Potential* (e.g. number of used projectors for its transformation to a separable pseudopotential), but all the properties could be redefined.

Finite element method objects In fact, there are many classes describing the discretization of the problem in the SfePy framework. Only a few of them will be mentioned here — the ones which are directly created within the FENNEC framework. For full documentation of the SfePy framework see SfePy webpages, which contains very elaborate documentation.[6]

Problem *Problem* object holds all necessary information about a particular discretized problem. This object is inherited[7] from SfePy `problem` object and extended by some useful methods.

This object serves as a container for all FEM/IGA related functionality: it holds FEM/IGA description of the volume and description of the discretized equations, and it provides an interface to the SfePy finite element framework (for interpolating, matrices assembling. . .)

Domain Each computation needs to have a *Domain* defined: a volume, where the calculation is performed, and its discretization. The domain can be determined either by a FEM mesh or, in the case of isogeometric analysis, by an IGA patch.

The meshes are created by plugins on demand: several plugins exist that can generate various meshes (see chapter 3.2.1). The mesh creating plugin can take the positions of the atomic sites into account. One configuration of a mesh generating plugin is thus suitable for a large scale of problems. Therefore, the meshing in the FENNEC package is usually reduced to the checking of automatically generated mesh, either visually using the mesh visualization plugin, and/or by a sample calculation.

The form of the FEM discretization also depends on the degree of the used basis, which can be determined in options. Therefore, a mesh generating plugin should provide the mesh only: the domain (SfePy object `FEMDomain`) is automatically created from it, according to the options.[8] IGA patches contain both the description of the volume and the definition of the corresponding basis (see section 3.3); therefore patch generating plugins have to provide directly an instance of SfePy `IGADomain`.

SfePy equation The *Equation* object mentioned above contains just a definition of an equation. The actual implementation of the equation — its discretization in a given *Domain* — is described by this SfePy object which is automatically created from the definition for a particular *Problem* object if it is needed.

Objects that hold computational data The last ones of computation-related objects are the objects that hold the quantities occuring in the calculation, e.g. charge density, potentials, etc.

[6]`http://sfepy.org`

[7]In the sense of the class inheritance in object-oriented programming.

[8]In the case of the mesh with hanging nodes, the data related to the hanging nodes must be provided in addition to the mesh definition itself, due to actual implementation of hanging nodes in the SfePy framework.

Context *Context* is the main object for a computation. It is associated with its *Problem*, however, more *contexts* can exist for one *Problem*.

Since the values of many quantities differ in each iteration, *Context* does not directly own data; it just serves as a container for *Iterations*. However, since just the data from the last iteration are usually needed, there is a syntactic sugar[9]: the data can be read directly from *Context* which returns the data related to the last iteration.

Iteration The object *Iteration* holds data valid for just one iteration of a DFT cycle. It contains *Variables* for spin-undependent data, and one or two *Spins* for spin-dependent ones.

Spin *Spin* is a container for spin-dependent data. This object is employed even in spin-restricted calculation (see section 2.4.2), to make the spin and spinless calculation as much similar as possible. Therefore the calculation with and without spin differs just in the number of *Spin* objects contained in each *Iteration*: the spinless calculation utilizes one *Spin* only, whereas a SDFT calculation two *Spins*. The actual computation of quantities that are potentially spin-dependent is performed by a cycle over the *Spins*.

Spin object also expose all (spinless) variables from its parent *Iteration*. Therefore, one can pass *Spin* to all routines that excepts *Iteration*.

Variable Both the variable containers — *Iterations* and *Spins* — hold *Variables*, i.e. the instances of *Variable definitions* prescribed by plugins. The variables are created on-demand: when a new unknown variable is requested, FENNEC looks for its prescription and creates the variable according to it. The prescriptions are contained in plugins. To be loaded automatically, the plugin should have the same name as the variable.

Options To allow to configure each calculation independently (e.g. set the DFT loop precision, solver precision, etc.) each *Context* holds its *Options* object. The *Options* objects are hierarchical: any option undefined in a child *Option* object is inherited from its parent. In most cases, the hierarchy has two levels: all the *Context Options* inherit from a *Global options* object, but there can be any level of nesting (e.g. a child *Options* object can be created to change some configuration temporarily).

FENNEC kernel objects

The last-mentioned group of FENNEC objects is kernel objects: objects that provide all necessary functions for the already mentioned functionality. Unlike other mentioned objects, they are typically instantiated just once and they live during the entire run of the program.[10].

[9]An easy way how to express something.

[10]But they are not true singletons, since theoretically they can be constructed (e.g. for the purpose of testing) more times.

Application The *Application* object represent the core of the whole package. When the program is started, the *Application* loads all other core objects, prepares the environment, initializes *Global options*, loads all plugins given by command line and/or configuration file and then runs the *Program*.

Program After the initialization of the *Application*, the program flow is no longer fixed, but vary according to the loaded plugins. The loaded plugins register the requested task by the *Program* plugin (typically during the initialization of the plugin) and the *Program* object stores the requested tasks and runs them in the proper order.

Plugins The *Plugins* object takes care of all the required tasks, looking for plugins and loading them. Generally, a plugin can be loaded by its name. FENNEC looks for it in the subdirectories of plugin directories (several plugin directories can exist, e.g. one directory provided by the FENNEC framework itself and the second one containing one-time plugin scripts written by a FENNEC user).

If there is no class with a "camelized" name of a plugin in the loaded plugin file, the plugin class is created from functions contained in the file. This property allows creating very simple plugins (commonly containing just one or two function calls) without bothering about importing proper base class and all the complex functionality behind plugins.

However, there is still an option to define a Plugin as a "true class" to utilize any of the extended functionality provided by the FENNEC.

Global options This object stores global and/or default options, which are inherited by all *Context Option* objects. E.g. the degree of the used FEM basis, the order of the numerical quadrature, the convergence threshold of a DFT cycle, etc. belong to those options. A FENNEC user can change the options via command-line argument or in a config file. Plugins can change these options and/or define their own ones.

Actions, Hooks, Equations, Variables definitions Containers for objects of given types. All of them support dynamic loading if the user (or a plugin) requests a so-far-unknown one. Exception for the *Hooks* object: unlike the others, there is no requirement for a hook to exists. Thus, calling unknown hook does not result in searching for its implementation. This behavior is used e.g. for an easy debugging of the code: calling the hook `breaks` does nothing unless the plugin `breaks` is loaded[11]: in such a case, the hook causes the debugger to be run. This allows having predefined breakpoints on important places of the code and activating them on demand easily, without any modification of the code.

Database All the *Core types* are stored in *Database*. The atomic configuration is stored in a tree structure to allow to store the values in common for more atomic types in one place. The tree structure of *Database* can be mirrored in the filesystem, where the data files containing the potentials should be placed.

[11]With the same argument as the hook is called with.

The common use is to define the properties of a given chemical element as a tree node, and its variants (e.g. the potentials that reflect different locations of the particular atom in the calculated structure) as child nodes.

6.1.2 Discussion of the FENNEC architecture

We could notice the whole FENNEC package is built around plugins — each plugin is a small independent piece of code, that solves just one thing. The kernel of the package consists just of a few, mostly general-purpose, objects. Therefore the kernel functionality could be easily adopted for solving a particular type of problem using FEM or IGA, even not related to the electronic structure calculation at all.

All other non-plugin objects serve primarily as data storage. Therefore, nearly all the key functionality of the FENNEC package is contained in plugins: e.g. DFT loop, mixing, the DFT iteration itself, evaluating of the functionals (XC, Hartree) etc.

This approach provides the following advantages:

Architecture quality The plugin architecture naturally leads to partitioning of the code to as-independent-as-possible routines. Therefore, the code is clearly arranged and straightforward, compared to common monolithic code, having a minimum amount of hidden dependencies that would hinder from understanding the code, possible reusing an old functionality or implementing a new one.

Code cleanliness Concepts of plugins simplifies the code in many cases. For example, to implement the capability to iterate charge density or potential in the DFT loop, the monolithic code would be typically interspersed with if-clauses "potential or charge". In plugin architecture, the natural way of implementing the loop is to write two simple plugins: the implementation of a DFT iteration with potential mixing and the implementation of charge density mixing: both of the implementations consist just of a few lines of code because all underlying "common" functionality is provided by the other plugins.

Code cleanliness can be, of course, achived within a non-plugin architecture as well if it is well designed. Nevertheless, even well designed monolithic code must be explicitly written with the possibility in mind to replace a particular implementation of the routine with another. The plugin architecture naturally leads the developer to separate independent tasks and provides him all the necessary tools to do such a decomposition without greater effort, which significantly contributes to code cleanliness.

Extensibility and modifiability In a plugin architecture, any piece of code can be easily (temporarily or permanently) replaced by another version. This allows easy comparison of different implementations of the same functionality (e.g. different XC functional library, different mixing algorithms, and so on), just by loading an appropriate plugin, which can be done just by one command-line argument. This property is very useful for developing, where a comparison of a newly created implementation with a reference one is often needed, as well as for an ordinary use (e.g. for switching eigenvalue solvers if the default one does not converge in a problematic case, or to include another XC functional to the program without the necessity of any code modification).

Easy test writing The plugin architecture — especially the concept of actions — provides a straightforward way to developers of mocking (replacing a function by "fake" implementation returning precomputed data) the functionality. Dependency injection, which is a common approach recommended for mocking, often leads to the necessity of passing many arguments through routines, which makes the program much worse readable, while the simpler use of singletons leads to a far less flexible code.

Configurability The partitioning of the functionality into plugins makes the program easy customizable since each plugin can receive its own set of options. The plugins thus create natural "namespaces" for program options, which makes the configuration very easy: associating the parameters with plugins allows a user to easily find all the options related to a given functionality.

Such property also allows to expose (make available) nearly all possible configurations as plugin parameters to the user, which results in a very high degree of customizability, whereas the program is usable — owing to default values — even without (nearly) any configuration options in common cases. The plugins' parameters also form the basic self-explanatory documentation of the plugins.

Powerfull but still simple The plugin architecture is used for defining the functionality as well as for defining the solved problem itself (e.g. for defining the geometry of a calculated structure). Thus, while entering a common configuration remains simple and easy, either via commandline, e.g.:

```
> dft dimer='(N,O)',bond_length='115pm'
```

or via simple plugin scripts:

```
def init(self):
self.application.set_atom_conf([
('N', -115,0,0),
('O', 0,0,0)
], units ='pm')
```

— still the full power of the Python language is available for all the necessary preprocessing (e.g. for determining a complex geometry of a deformed semiperiodical structure) without the necessity of using any external tool.

The selected architecture has, of course, also some drawbacks. The independence of plugins leads to the necessity to pass all the data between plugins somehow, since the client of a plugin cannot in principle know which data the "server plugin" needs to perform the requested calculation. While one implementation of a given routine could use a certain quantity as input, another implementation can use more of them or different ones.

This problem has been solved by the introduction of *Context* objects, which holds all the quantities in the computation and that is able to evaluate them "lazily" (on demand) via *Actions* and *Variables*. This solution violates another paradigm of good programming — *"to pass only the data actually needed to the called routines"* — but the other possible solution — to change the implementation of the caller each time a callee is changed — is substantially worse. Since the different implementations of many routines (e.g. XC functional or total energy evaluation) often require different sets of input parameters, no better way seems to be available.

Another concept used in the code that can be viewed as controversial is the concept of autoloading of plugins and automatic evaluation of variables. It can be viewed as a violation of the basic (and reasonable) Python law "*Better explicit than implicit*"[12]. It can exhibit unexpected behavior in some cases — e.g. the variable that is used with a hope to be a "fast input guess" might be automatically (and "costly") computed, if a user forgot to specify not to do it. Moreover, the implementation of autoloading relies on adding (on the fly) objects (variables, equations) to the existing instances of the SfePy problem, which has required a certain amount of work on patches to the SfePy package to enable that functionality.

However, if the concept of autoloading were not employed, the extensive granularity of the code providing most of the benefits mentioned above, would require a lot of "load a dependent plugin" commands, which would ruin the readability of the code. Moreover, autoloading is very useful for interactive computing (e.g. during manual postprocessing of computed data in Python shell).

Thus, the concept of autoloading, despite the disadvantages mentioned above, substantially simplifies the code and facilitates a developer to focus on the solved problem and not on the implementation details. It assists in keeping the code clean and clearly arranged and enables faster development of new functionalities within the framework. Awareness of the drawbacks mentioned above helps to design the architecture and the interaction among plugins so that the consequences of the drawbacks could be mitigated.

To conclude, all the problems mentioned above do not represent important issues, compared to the benefits provided by the chosen architecture: the plugin architecture very simplifies both the use of the framework and its further development.

6.2 Used programming languages and libraries

This section discusses the used programming languages and libraries. The discussion reflects my experience with software development (in several programming languages) and it is, of course, presented from my subjective point of view. The

[12] `https://www.python.org/dev/peps/pep-0020/`

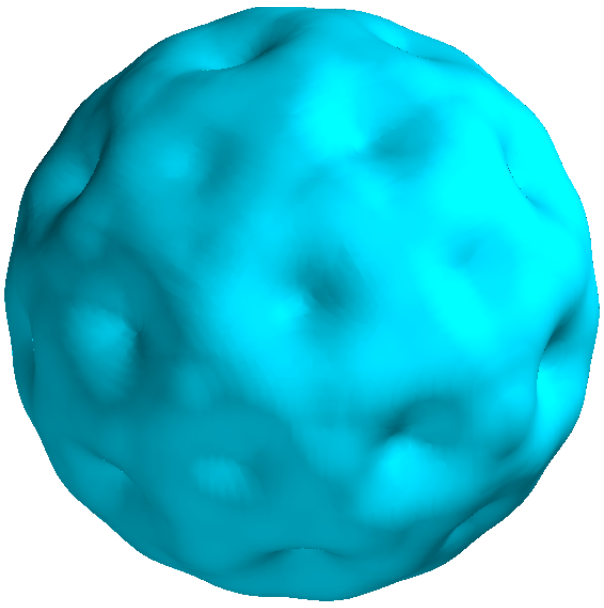


Figure 6.2: Charge density of fullerene C_{60} calculated using FENNEC and visualised using SfePy and Maya-Vi.

following contemplation should not be taken as proven facts, but just as experience obtained during the development of such a not-so-small framework. I believe that sharing experience and developer skills is useful and that the summary of the experience collected during the development can help other scientific developers to find the right tools for their work.

FENNEC is written in Python 3 [209] using its standard implementation CPython[13]. The choice of Python 3[14] was justified primarily by the need for problem-free compatibility with SfePy finite-element framework[207, 150], and for cooperation with SfePy author — Robert Cimrman from the University of West Bohemia. The language proved itself to be worthy of that choice. Even if the language has its weak points, the development in Python is substantially faster than in other languages that I'm experienced with. Not only compared to the compiled languages like C, C++, Delphi, or Fortran (where it is expectable) but also with the interpreted ones, which are aimed at rapid development, e.g. Java or PHP.

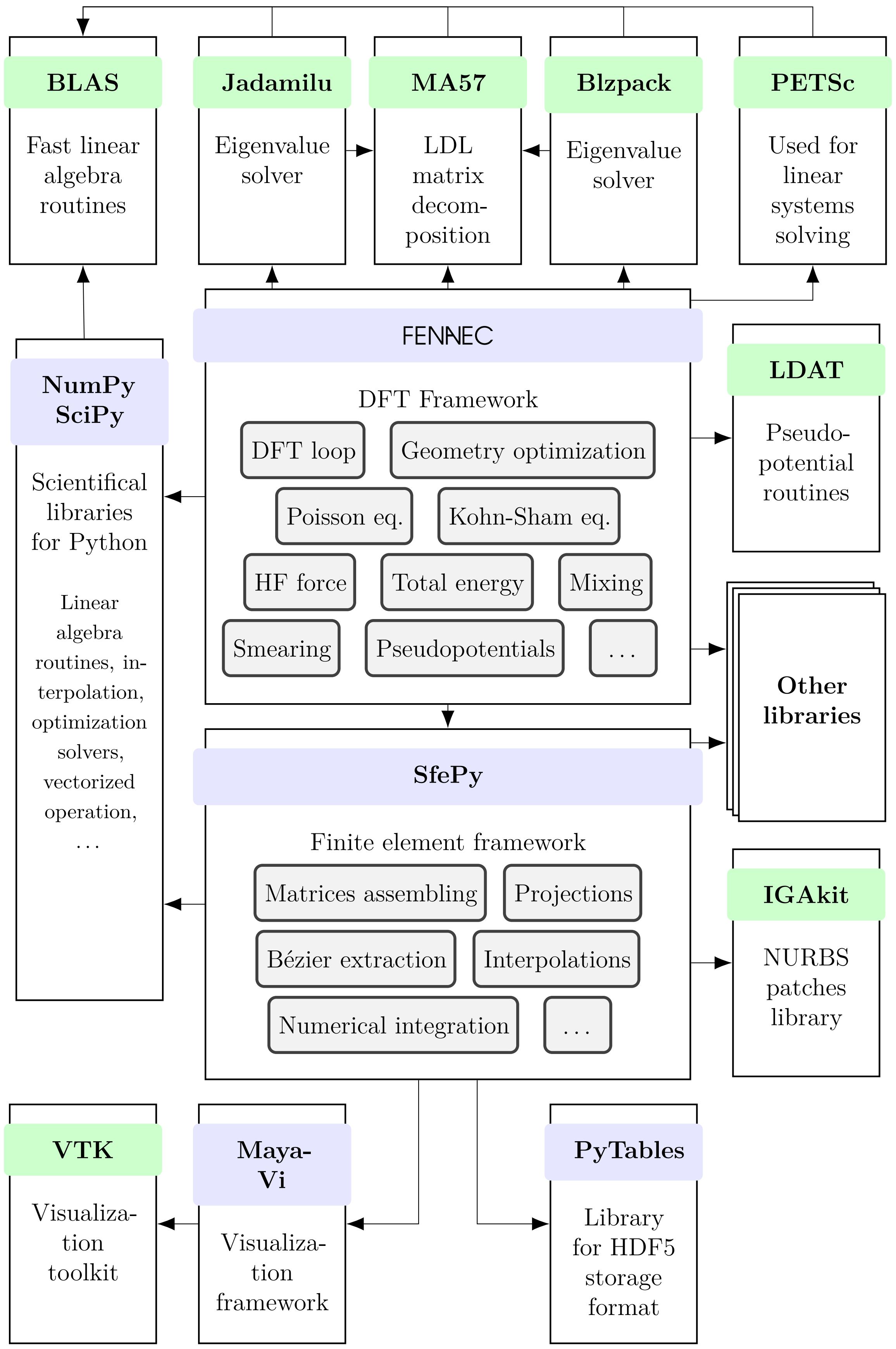
The result of comparison with PHP[15] seems weird since PHP language (nearly unused in the scientific community) is very frequently used by commercial companies developing (mostly web-based) software, where using of the most productive languages would be expected. However, this observation is affected by the local sit-

[13]`https://www.python.org/`

[14]Originally, FENNEC have been developed in Python 2, but the project was migrated to the Python 3 not to become obsolte

[15]`http://php.org`

Figure 6.3: Architecture of the FENNEC framework – the main used libraries and their main relations and dependencies. Python libraries are shown in blue, compiled (Fortran, C or C++) ones in green.



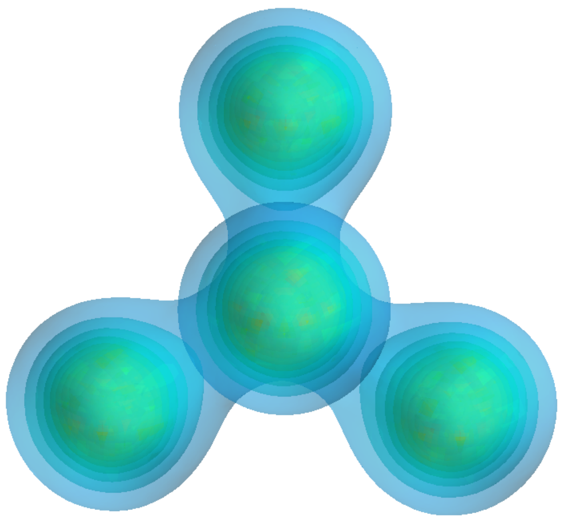


Figure 6.4: Charge density of CF_4 calculated using FENNEC and visualised using SfePy and Maya-Vi.

uation in the Czech Republic, the worldwide statistic shows the growing popularity of Python[16] already overcoming the PHP popularity [210].

These facts confirm the quality of the language: especially very clean and idiomatic design of the language should be mentioned, which facilitates keeping the code readable and speeds up the development.

One of the strong points of the Python language is the strong community around it, especially a large scientific community which develops many useful scientific libraries for Python. First of all, the following ones should be mentioned: NumPy [211], which provides reasonable fast linear algebra and large data manipulation routines, and SciPy [212], the extension of NumPy, which provides an implementation of many useful algorithms from many scientific fields. Also, the visualization framework Maya-Vi [213], based on the VTK library [214] is worth mentioning, and, of course, SfePy framework itself, upon which is FENNEC built. The basic scheme of the framework and its library dependencies can be seen in Fig. 6.3. A list of main used libraries can be found in appendix A.

Another strong property of Python language is the possibility to stop the program execution and to drop to a Python shell at any point of the code, just by calling one simple function. In the stopped program, not only one can debug the current execution (as it is common in other languages) but he can run a full-fledged Python interactive shell, too. This “trifle” in fact very simplifies a programmer life, since it allows rapid prototyping and debugging without the necessity to run the program again and again to correct all bugs in the code.

To assess also the weaknesses of the Python language, I should mention especially the one-threaded nature of the CPython implementation of Python (other Python implementations are not compatible with many needed libraries) due to using so-called “global interpret lock” (GIL) during accessing any language object. This

[16]`http://pypl.github.io/PYPL.html`

is a pretty big problem, as the performance requires the parallel implementation of the crucial parts of code. The parallelization will be discussed in the next section.

The only other more serious issue concerning Python that I encounter during the development is a bit tricky implementation of AsyncIo[17] — part of the language that allows programming in coroutines. Coroutines are tasks that can be run semi-parallel — just one coroutine can be active at a given moment, but when a one is waiting for something (e.g. for data from a subprocess), it can put itself to sleep and another coroutine can be run (or resumed). This concept very simplifies running of some parallel task, but the Python implementation has some subtleties and the code is sometimes hard to debug.

The Javascript implementation of the coroutines, from which the Python AsyncIo has been inspired, is simpler to use, better designed, and moreover multithreaded (using worker threads, see e.g. [215]). If we consider not too complicated C to Javascript binding, modern Javascript (despite its origin as web-based language) would be a surprisingly handy tool for scientific computing, if there were similar library background like in Python. However, the lack of useful scientific libraries (particularly an equivalent of NumPy) means that a large amount of work would be necessary to build a suitable environment for scientific computing.

Another language which is worth mentioning, is a relatively new language called Julia [216]. This language takes over many good properties mentioned above of both Javascript (fast just-in-time compiler, easy threading, well designed coroutines) and Python/Cython (environment for numerical computing, code block definitions via indentations, very easy binding of external functions); and its primary aiming for numerical computing makes it a very promising language for developing scientific applications.

Python and compiled languages A comparison of Python with compiled languages has obvious results: interpreted languages as Python are designed for rapid development, while the compiled ones like C, C++, or Fortran offer better performance. This general rule has of course some exceptions, when in certain cases just in time compilers are able to beat the statically compiled code[18], but it holds in common cases.

Therefore, the performance-crucial parts of the framework have to be written in a compiled language. The routines extending solvers capabilities to solve rank-n-update problems have been written in Fortran [217], since the used solvers themselves are written in Fortran, too. The Python-Fortran interface has been built using `f2py` utility, which is part of the NumPy package [211].

Secondly, the Cython language [218] has been used for that purpose, as it makes the process simpler and more error-prone and, moreover, it provides more options and tighter binding of the compiled and interpreted world. The use of Cython for other performance-crucial parts (e.g. for calculation of border conditions for the Hartree potential) has been motivated by a very easy Python-Cython "integration" and by an easy transformation of the optimized code from Python to Cython.

[17]https://docs.python.org/3/library/asyncio.html

[18]E.g. see e.g. `https://julialang.org/benchmarks/` or `https://hhvm.com/blog/2027/faster-and-cheaper-the-evolution-of-the-hhvm-jit`

Not only my experience shows that the parts of code that are necessary to be written in a compiled language are either large independent algorithms, (e.g. eigenvalue solver) — for which the best strategy is to use a library developed by specialists in the given field — or pretty small routines — that can be easily transformed to a compiled language. Especially Cython [206], Numexpr [219] or Numba [220] are very useful for this purpose.

The benefits from the rapid development in Python exceed many times the necessary work to bind the compiled and interpreted parts of the code. Python is commonly used as a rapid glue, which allows fast development of the software, providing many ways for optimizing performance-crucial parts of the developed software.

To conclude, Python is a well-accepted language in the scientific community, and (not only from my point of view, see e.g. [221]) the use of Python is very recommendable for the purpose of developing scientific software.

6.2.1 Parallelization

A modern scientific software must allow parallelization to achieve a good performance on multicore processors and multinode clusters. However, several pretty different concepts are hidden behind the word "parallelization", the use of which in FENNEC will be discussed in this section.

The first possible level of parallelization can be called *library parallelization*. It is also sometimes called "BLAS parallelization", but the concept is not specific to the BLAS numerical libraries. In this concept, the program itself is single-threaded, but it uses libraries that can perform some operations (e.g. matrix multiplication and other matrix operations in the case of BLAS) using library worker threads. Usually, the implementation of this level of parallelism consists just in linking the right library. Cython [218] routines, parallelized using Cython "`range(..., parallel=True)`" or OpenMP [222], or Numba [220] compiled routines, can be included under this term, as such pieces of code can be viewed as library functions.

The previously described concept is sometimes also called *thread-level parallelism*, although this term also denotes a substantially different concept. In a "true thread-level parallelization", more time-consuming tasks are computed simultaneously by parallel threads. Unlike the "library parallelization", where one small part of the code is parallelized and one task is computed by more threads, in thread-level parallelism, the threads do not cooperate on one task, but each thread solves a different task. As a consequence, this concept does not suffer from locking issues and work distribution overheads (at least compared with the previous case) and therefore it provides a substantially greater performance benefit generally. Such a concept would allow e.g. to compute spin-related quantities in two (independent) threads.

There are authors that do not recommend the thread-level parallelization at all (e.g. [223, 224]). Even if the criticism of thread-level parallelism comes mostly from network-related-software developers and not from the high performance computing community, some of their arguments are valid globally: e.g. that thread-level parallelism is very hard to debug and that a well-designed process-level parallelization can utilize the parallel computational facilities in the same way as threads.

The second claim should be especially emphasized, since in many cases, the implementation of thread-level parallelism just delays implementation of a process level parallelism, which is, unlike the thread-level one, easily expandable to more computational nodes. However, there is still one indisputable advantage of thread-level parallelism: a possibility to pass large data between threads easily without greater performance penalty — so the thread-level parallelism has (in my opinion) its rank in scientific computing.

The *process-level parallelism* is the highest level of parallelism (from a certain point of view) since it can, and usually does, utilize more computer nodes, and thus requires a coarse work distribution to overcome the slow interprocess communication.

It should be distinguished, whether the processes run on the same machine, or have to communicate over a network (no matter if over common Ethernet or over a high-performance network as e.g. Infiniband or Omni-Path, which generally provide higher throughput or shorter latency, e.g. [225, 226]). In practice, both the one-node and the network process-level parallelization require the same approach. Thud, it is a good strategy to skip the one-node level completely. Moreover, many libraries for this type of parallelization — from a de facto standard MPI (Message Passing Interface) [227] that should be mentioned here — totally hide the differences between one- and more-nodes parallelization.

The *accelerator parallelism* is the concept that stands a bit aside from the mentioned others. Graphics cards — i.e. the most commonly used accelerators — similarly as e.g. neural network accelerators, are suitable only for specific tasks, while the remaining parts of the code must be still computed sequentially using CPU. From this point of view, the accelerator parallelism is alike the library parallelism. On the other hand, the overhead of data transfers between an accelerator and main memory is by no means negligible, which recalls much more process-level parallelism than the library parallelism.

Parallelization in the FENNEC framework

FENNEC employs two levels of parallelism at present time. The first is the library parallelism. The main usage of this type of parallelization is a threaded implementation of the BLAS library (library for fast matrix-vector and matrix-matrix operation). We have good experience with the OpenBLAS [228], however, any BLAS implementation can be used. According to our tests, MKL achieves similar speed as OpenBLAS on our heterogeneous (both AMD- and Intel-CPU cluster), whereas ATLAS [229] and AMCL [230] were slower. The new AMD BLAS library has not been tested yet. However, BLAS implementation is so dependent on both the used compiler and on the CPU architecture, that just a much larger and more rigorous analysis could claim general validity.

The library level of parallelism is used in FENNEC with no problems in many cases — e.g. in computing boundary conditions of the Poisson equation. However, in the case of eigenvalue solver, the performance of which is crucial for the package, the effect of this level of parallelization is limited.

The thread-level parallelism is not employed in FENNEC since it brings no advantage for CPU intensive tasks for a program written in CPython due to its GIL. Therefore, the next level of parallelism implemented in the FENNEC is the process level parallelization. The parallelization is implemented using a classical client-server design pattern, where a number of workers "server" processes can be started and the tasks can be distributed to them.

The configuration of the worker processes is given by a plugin, therefore it is easy to specify the nodes of a cluster where the processes should be started, and the number of used CPU cores — either manually, or e.g. using environment variables provided by a batch system (e.g. SLURM [231], which our group has good experiences with).

The communication between client and server has been written using AsyncIo routines and employs the already described concept of *Actions* and *Variables*: client can run any *Actions* on any server; there are actions defined to read and write *Variables* for setting up the server process and for retrieving results from it. FENNEC developers have at disposal the methods to scatter *Actions* with a given set of arguments across worker processes, as well as the methods to queue different *Actions* to be successively called on the free worker processes.

This concept allows easy large scale parallelization, e.g. to evaluate HF forces for several configurations of atomic structures: it is sufficient to write a small plugin that defines an action, and scatter the set of arguments of the action across the plugins. The concept is very similar to the MPI parallelization mentioned above, but it is tighter bound to FENNEC, which makes it a bit simpler to set up and use.

Moreover, MPI has not been used intentionally for two reasons: First, having *Actions* and *Variables* available as communication language between processes, it is relatively easy to add the required functionality for parallelization. Second, the MPI is utilized by some eigenvalue solvers (see the next section). The performance of the eigenvalue solver is crucial for the performance of FENNEC. From the architecture point of view, eigenvalue solver parallelization lies on a lower level than e.g. the parallelization of computations of different configurations of a material sample. It is simpler (both for a user and a developer) to differentiate these two levels of parallelism. Thus, the standard way for utilization process-level parallelization has been reserved for the performance-crucial eigenvalue solvers, which makes both their implementation and use simpler. The use of MPI for eigenvalue solvers will be further discussed in the next section, dedicated to solvers.

And finally, since the cluster, where the FENNEC is primarily used, is not equipped with (computational) graphics cards, the FENNEC currently does not employ any form of the accelerator parallelism. Accelerators could be utilized for an acceleration of some linear algebra routines, which could be easily implemented either as a special solver plugin. With the increasing speed of GPU-CPU communication and/or existence of GPU that shares its memory with the main CPU, it may be also worth employing the GPU accelerated BLAS. However, we have not enough programming time capacity for attempts in this direction.

There are of course other parallelization libraries, undiscussed yet and unused in FENNEC, that are worth considering for a scientific developer. Among the most interesting I can mention DASK [232], a library for parallel computation (not only) with NumPy arrays.

To conclude: leaving aside the linear algebra solver that will be discussed in the next section, the parallelization of the FENNEC software in the current state of development generally fulfills all the needs for a large scale computation.

6.3 Linear algebra problems and used solvers

This section is dedicated to solving linear algebra problems and to linear algebra solvers employed in the FENNEC framework. The solvers truly deserve their own section, since their performance is crucial for the performance of the whole package.

The eigenvalue problem solving is the most time-demanding task in the whole DFT calculation: solving of the eigenvalue problem takes up more than 95% of the time in large scale tasks. Therefore, the attention is paid primarily to this problem. Solving a rank-k-update eigenvalue problem had been one of the main topics of my master thesis several years ago. Thus, I describe the used algorithms here just briefly. For a more detailed discussion of the topic, I refer the reader either to my thesis [233] or to some of many books covering this topic, e.g. [118, 234, 235]. Especially for the case of the Krylov subspace algorithms, I recommend the book *Krylov Subspace Methods — principle and analysis* [6] that provides very deep insight into the inside of the Lanczos algorithm enriched by historical context.

As it has been already explained in the second chapter, two differential equations have to be solved in each DFT iteration. The first is the Poisson equation 2.47, whose discretization yields a system of linear equations (see the equation 3.29 in chapter 3.1.2). The second and the more important ones are the Kohn-Sham equations 2.44, which results in a rank-k-eigenvalue problem (see 4.50 in chapter 4).

6.3.1 Common background: direct and iterative solvers

There are two basic kinds of both eigenvalue and linear system solvers: the *direct* solvers and the *iterative* ones (see e.g. [118, 233]) — their brief comparison can be seen on the Tab. 6.1. The direct ones are usually based on a factorization of the matrix and thus they generally require a large amount of memory for storing factorized matrix (factorization does not generally retain sparsity) and a not negligible time for performing the factorization. Therefore, they are suitable in the cases when a large number of solutions for one matrix is needed.

Iterative solvers have a pretty different nature. Iterative solvers algorithms (at least the well known ones) can be again divided into two groups: to the methods that iterates vector or block of vectors till they converge to the solution(s) (e.g. *power* or *conjugate gradient method*), or to the ones that build a subspace iteratively, into which is the problem projected to obtain the solution (for example Krylov subspace methods).

The iterative nature of such algorithms determine their properties: as they iterate only a few vectors at once (or add them to projection space, respectively), iterative solvers can offer only a limited number of solutions in each run. On the

Table 6.1: Comparison of direct and iterative solver types — common properties

	Linear problem solver type	
	direct	**iterative**
number of computed eigenvalues/right hand sides	many or all	a limited number
required operations	factorization	matrix-vector multiplication (and linear system solving for eigenvalue solver)
the control of the result precision available	no	yes
parallelization	not so good	very good
memory consumption (for sparse matrices)	high	low
time for the first solution	high	moderate
time for next sollutions	low	moderate

other hand, they offer several advantages: they are generally much more memory efficient, they can be more easily parallelized and they do not require "pre-warming" time for factorization of the matrix as the direct solvers. Moreover, they offer precision control: one can stop the iterations as soon as the requested precision is achieved. Therefore, if only a few (with regard to the dimension of the matrix) solutions of the problem are needed, the iterative solvers are generally a better choice.

From the previous discussion follows that the iterative solvers are more appropriate both for the Poisson problem same as for eigenvalue problem arising from the Kohn-Sham equations discretization since in both cases only one or few solutions are needed.

6.3.2 Eigenvalue solvers

We will see that finding a suitable linear solver for the Poisson problem is not so difficult task, as there are many libraries for solving this type of problem. The more difficult situation has been in the case of the rank-k-eigenvalue problem, as there is no existing library for such a problem available, and therefore an existing eigenvalue solver (and as we will see, solvers) have had to be extended to cover this type of problem. Moreover, the performance of the whole package stands on the performance of the eigenvalue solver.

Spectral transformations and Sherman-Morrison-Woodbury identity

Iterative eigenvalue solvers generally converge to eigenvalues with the largest magnitudes. This can be easily demonstrated on the simple *power method*: method that iteratively computes vector $x_n = A^n \mathbf{x_0}$, till it converges to an eigenvector. If we use a basis consisting from the matrix eigenvectors $[\lambda_0, \lambda_1, \ldots, \lambda_n]$ and set x_0 (in this basis) to the vector of ones, it holds that

$$x_n = [\lambda_0^n, \lambda_1^n, \ldots, \lambda_n^n] \tag{6.1}$$

Therefore, during multiplication, the largest magnitude eigenvalue components outweigh the others during consequent multiplications. (The proof is a bit more technically difficult for an unsymmetrical matrix case, see [233], but essentially the same). Since matrix multiplication is behind the scene somehow in all iterative algorithms, this behavior is more or less common for them.

However, we are interested in the ground states with such energies, which lie around zero. Therefore it is not viable to use a “naive” approach for solving generalized eigenvalue problem as the simple one:

$$Ax = \lambda Bx \qquad \longrightarrow \qquad B^{-1}Ax = \lambda x \tag{6.2}$$

Instead, we must utilize a spectrum transformation to make the desired eigenvalues the magnitude-largest one. For the purpose, a form of *shift and invert* transformation is commonly used. E.g. in Blzpack [174] the following one is used

$$B(A - \sigma B)^{-1}Bx = \lambda' Bx\,, \tag{6.3}$$

where the eigenvalues are transformed as

$$\lambda = \frac{1}{\lambda - \sigma}\,. \tag{6.4}$$

A *shift and invert* transformation is commonly built in into the package providing the eigenvalue solver. However, it requires matrix inversion, thus in the case of rank-k-update problem the inversion of rank-k-update must be treated. To do so, the Sherman-Morrison-Woodbury identity [236] is utilized.

Theorem 6.1 (Sherman-Morrison-Woodbury identity)**.** *Given matrices $A \in \mathbb{C}^{n\times n}$, $U \in \mathbb{C}^{n\times k}, V \in \mathbb{C}^{k\times n}, C \in \mathbb{C}^{k\times k}$ it holds that*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1} \tag{6.5}$$

Proof. This identity can be easily proved by multiplication of the equation with $A + UCV$, which results in the identity matrix on the both sides. □

The necessity of the inverting the matrix $(A - \sigma B)$ – since the inverted matrix is needed only for multiplication of vectors – can be (and usually it is) replaced by solving a system of linear equations. Therefore, we have just obtained a second linear system problem that has to be solved during the DFT iteration cycle. However, we let these linear systems aside and we will pay attention to the eigenvalue problem, first.

Reverse communication strategy

The requirement to solve rank-k-update problem limits the solvers that can be used in the FENNEC package: using the Sherman-Morrison-Woodbury formula we can still exploit the sparsity of the matrices, however, we need solver that let the user handle the matrix multiplication and inversion himself.

A common mechanism used for the purpose is *reverse communication strategy*: the solver interrupts its operation each time, when it needs to perform multiplication (either by matrix or by shifted and inverted matrix), and it returns program flow back to calling routine with a request for multiplication. It is the responsibility of the user of the solver to perform the requested operation and call the solver function again.

From the solvers providing such (or alike) interface, we choose (for a detailed description and the analysis of possible algorithms, see [233]) the BLZPACK [174] that implements Lanczos algorithm [235]. As the properties of the solver were unsatisfactory for the later stages of the DFT cycle (see page 186), we also implemented the JADAMILU [237] that implements Jacobi-Davidson algorithm [238, 239].

Blzpack

Blzpack [174] is an implementation of a block Lanczos algorithm [235] that projects an eigenvalue problem

$$Ax = \lambda x \tag{6.6}$$

into an iterativelly built subspace

$$K^n_{V,A} = \{V, AV, A^2V, \dots, A^{n-1}\}, \tag{6.7}$$

where V is a block of a few start vectors, typically of size from two to six. It can be easily verified that in the case of one start vector, matrix A projected into the subspace $K^n_{V,A}$ has zero lower triangle up to the diagonal next to the main diagonal of the matrix — such matrix is called *upper Hessenberg*. Due to the symmetry of the matrix A, the projected matrix has to be symmetrical, too: therefore, it is tridiagonal. Such a matrix has very interesting properties, see [6], and its tridiagonal structure has very desirable properties for solving the projected eigenvalue problem.

Moreover, since the subspace is formed by vectors $A^n\mathbf{v}$, it can be viewed as a generalization of the *power method*, where all the information obtained during calculation are preserved and used. Thus, the convergence of the method is much superior to the simple power method.[19]

However, several problems linked to the loss of the orthogonality during iterations (caused by finite precision arithmetic) spoil the theoretical estimations of the convergence. The estimation of the convergence rate in the finite precision arithmetic is still subject of research (see e.g. [240]), thus the popularity of the algorithm is based on the very good practical experiences obtained with this algorithm.

[19]The exact reason for (at least theoretical) excellent convergence of Lanczos method has in fact far deeper roots, however, their discussion is beyond the scope of the thesis, for more details see e.g. mentioned [6].

The block variant of the algorithm allows a better parallelization of the algorithm [235] and especially it improves the convergence of the method in the case of clustered eigenvalues: i.e. if there are multiple eigenvectors with the same or nearly the same corresponding eigenvalue.[241]

Blzpack was the first eigenvalue solver implemented into the FENNEC. However, since it uses only a small block, and therefore even a small block of starting vectors, it cannot fully reuse the solution from the previous DFT iteration. Moreover, it tends to miss some eigenvalues, especially in the later iterations, and that undesirable property spoils the convergence. Therefore, another eigenvalue solver has to been implemented.

Jadamilu

Jadamilu solver [237] is an implementation of the Jacobi-Davidson method [238, 242, 239]. Jacobi-Davidson method consists of building the subspace P_i iteratively, into which the problem is projected. The eigenvalues and eigenvectors are computed in this subspace, same as in the Lanczos method — they are called *Ritz vector* and *Ritz values* or together *Ritz pairs.*

The subspace is iteratively enlarged using following *correction equation*:

$$(I - B\mathbf{u_i}\mathbf{u_i}^{+})(A - \nu B)(I - \mathbf{u_i}\mathbf{u_i}^{+}B) = \mathbf{r}_i\,, \tag{6.8}$$

where I is the identity matrix, $\mathbf{u}_i$ and σ_i form an i_{th} Ritz pair, ν is an approximation of a wanted eigenvalue (user given in the first iterations, later updated by the computed Ritz values) and r_i is the residuum corresponding to u_i:

$$\mathbf{r}_i = A\mathbf{u}_i - \sigma_i\mathbf{u}_i \tag{6.9}$$

The correction equation 6.8 has a simple meaning: to enrich the space by the *correction* for the error, but only by the component orthogonal to the current estimation of the eigenvector. For

This scheme is a basic scheme of Jacobi-Davidson algorithm for a computation of one eigenvector. For computing more of them, a block version of algorithm or/and a *deflation* have to be incorporated into the algorithm, see e.g. [243]. For computing eigenvalues in the interior of the spectrum harmonic Ritz values — eigenvectors of shifted and inverted matrix $(A - \nu B)^{-1}$ — could be utilized for a better precision of the eigenpairs approximation [244].

The strong part of the Jadamilu solver should be the built-in preconditioning (using ILUPACK [245]) for the in JADAMILU built-in symmetric quasi minimal residuum solver [118] that solves the correction equation. The preconditioner is based on incomplete LU factorization [237] of the matrix. However, it cannot take the rank-k-update into account and therefore it does not perform well in our case.

Thus, we use JADAMILU with a user-supplied preconditioner based on shift and invert (see Eq. 6.3):

$$P = (A - \sigma B)\,, \tag{6.10}$$

using a fixed σ during the whole eigensolver run. σ must be chosen so that the resulting preconditioner is positive definite, therefore, it should be lower than the lowest expected eigenvalue. On the other hand, too low values of σ lead to worse convergence. The good "guesses" for σ are between -6 and -10 for the majority of the matrices resulting from the discretization described in the previous chapters for the first iteration. $\lambda - 2$ appears to be a good guess for σ in the subsequent iterations, where λ is the lowest eigenvalue from the previous iteration.

Requested precision of the eigenvalue solvers

It is not necessary to solve early iterations of the DFT cycle in full precision. To save computational time, the required precision of the eigenvalue solver is set according to the convergence criterium.

The default criterium for convergence is the L^2 norm of the error of the charge density (the difference between input and output charge density). First, the convergence criterium is transformed into a *convergence rate.* The convergence rate is expressed as a dimensionless number from interval $\langle 0, 1\rangle$, where 1 means fully converged calculation (when convergence criteria reach e_ρ) and 0 means totally unconverged calculation (the error is greater than 1). The convergence error is mapped logarithmically to the convergence rate between these two bounds

In FENNEC, there are four (user configurable) options affecting the transformation of the convergence rate to the solver precision: c_{min}, c_{max} and e_{min}, e_{max}. c_{min} (c_{min}) defines bound for the convergence rate, where the minimal (maximal) solver precision e_{min} (e_{max}) is used. Exponential mapping from the convergence rate to the solver precision is then used within the bounds $(c_{\text{min}}, c_{\text{max}})$.

$$\|\rho_{\text{out}} - \rho_{\text{in}}\| \xrightarrow[\langle 1, e_\rho\rangle]{\substack{\text{logarithmic}\\\text{mapping}}} \begin{matrix}\text{convergence rate}\\ \langle 0, 1\rangle\end{matrix} \xrightarrow[\langle c_{\text{min}}, c_{\text{max}}\rangle]{\substack{\text{exponential}\\\text{mapping}}} \begin{matrix}\text{solver precision}\\ \langle e_{\text{min}}, e_{\text{max}}\rangle\end{matrix} \tag{6.11}$$

Comparison of the used solvers

The two solvers are used since each of them offers different properties. It has been already noted, that Blzpack iterates only small block of vectors and, thus, it is not able to fully reuse eigenvectors from the previous DFT iteration as an estimation of the newly computed eigenvectors.

The Jadamilu solver constructs the projection space by iterating a large block of vectors. This block can be initialized by the results of the previous iterations, and the larger iterated block (compared to Blzpack) is less prone to the missing of an eigenvalue. However, Jadamilu algorithm turned out to be less robust: in the case when only inexact estimations of the solutions (or none at all) are available, the solver sometimes does not converge at all, and the necessity of guess the value σ makes the Jadamilu less robust in the first iterations of the DFT cycle.

We can see in Fig. 6.5 that in the first iterations, where no or poor estimations of the eigenvectors from previous iterations are available, and only low precision is requested, BLZPACK performs betters than JADAMILU. The picture rapidly

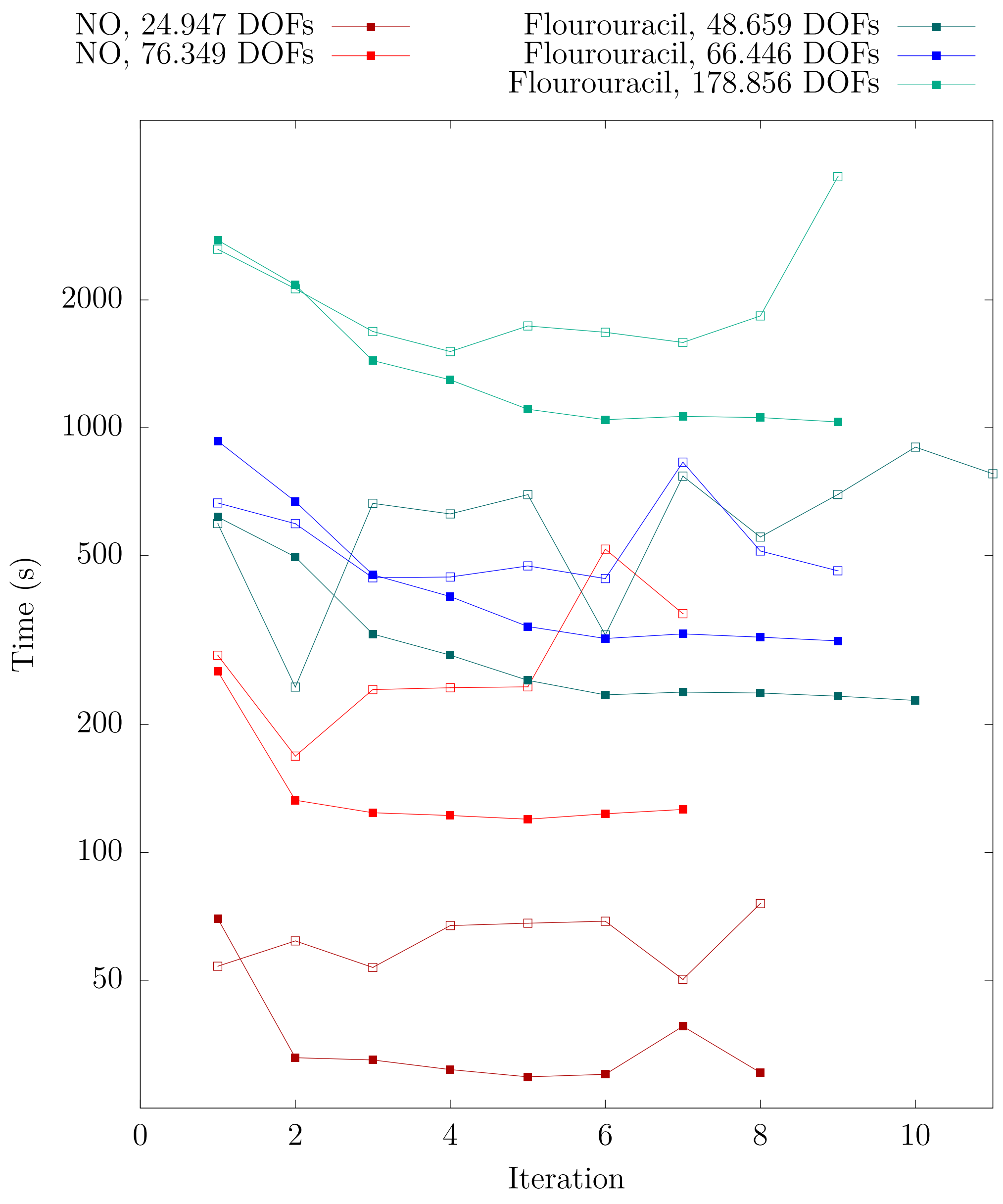


Figure 6.5: Performance comparison of BLZPACK □ and JADAMILU ■ solvers for iterations of DFT cycle for Flourouracil and Nitric Oxide for various meshes. DOFs denotes the number of degrees of freedom: the dimension of matrices of the resulting generalized eigenvalue problem. Computation was performed on 2×6-core Xeon E5645 [171] with 800Mhz DDR3 memory.

changes in the later iterations, where JADAMILU can fully utilize the eigenvectors from previous iterations.

From the previous follows that it is the best strategy to start with the BLZPACK solver and after few first iterations (typically one to two) switches to JADAMILU. The moment when FENNEC switch from BLZPACK to JADAMILU is configurable (by the iteration number or by the convergence rate).

6.3.3 Linear system solvers

There are two linear systems that have to be solved: the Poisson equation 2.47, and the linear system for shift and invert transformation (Eq. 6.3) in BLZPACK or preconditioning in JADAMILU, respectively. Although both of the systems are sparse and with the same matrix structure, the requirements for the solver are quite different: the boundary conditions change each iteration for the Poisson equation, therefore only one result for each "Poisson matrix" is needed. In the case of the shift and invert operation and preconditioner, there have to be solved many right-hand sides for the same matrix.

For one-time solving of the Poisson equation, an iterative solver is the best choice, again. We had utilized SciPy [212] conjugate gradient algorithm [118], but its performance was unsatisfactory for larger problems. PETSc [246] implementation of the conjugate gradient, preconditioned by ICC [247] incomplete Cholesky factorization, provides much better performance and makes the time needed for solving Poisson equation insignificant relative to the time needed for solving the eigenvalue problem. E.g. there were totally 305 seconds (average 34 seconds for a problem) spent in Poisson equation solver in the largest problem on the Fig. 6.5, compared to 12.974 seconds needed to solve all eigenvalue problems in the same DFT cycle (average 1441 second per one eigenvalue problem).

The required precision for the linear system solver is set to the required precision for eigenvalue solver multiplied by 1×10^-3, as this threshold seems safe according to the performed test runs: a higher multiplicator than 1×10^-2 leads to a slower convergence of the DFT cycle.

For the shift and invert/preconditioning case, where many right-hand sides of the same matrix have to be solved, a direct solver is a better choice, as long as the memory requirements could be fulfilled. First, FENNEC utilized the MA47 library for that purpose [248], which we have later replaced by newer and substantially faster MA57 library [140]. Both the libraries do a sparse factorization of a matrix that is then used for solving the linear system.

The prewarming factorization time (76 seconds in the mentioned case) of MA57, which is the largest disadvantage of the direct solver, is negligible to the total time of the eigenvalue solver and it is well amortized: there is e.g. average 3274 requests for linear problem solving per one DFT cycle iteration for BLZPACK respectively 2031 requests for preconditioning for JADAMILU (for the same problem as in the previous paragraph). Computational time for solving one right-hand side is lower than one second. The greater memory requirements of the direct solver do not matter here too, as it is still low compared to the memory requirements of eigenvalue solvers.

6.4 The current and the future work

To conclude, FENNEC is a mature software package for electronic structure calculations. Its open plugin-based architecture allows both its easy use as well as its easy further development. In a near future, we plan to incorporate new routines for relativistic pseudopotentials that are already developed by other members of our team. These pseudopotentials also allow us to put spin density functional theory calculations into practice.

Other possibilities to improve the FENNEC package are mostly outlined in the thesis: further research and improvement of a mixing algorithm and utilizing of a more sophisticated geometry relaxation solvers (see the next section), implementation of partially periodical boundary conditions, utilizing of FEM *p*-adaptivity or IGA locally refined patches.

A special field for improvements is the solvers since they determine the performance of the package. Thus, I summarize their evaluation and current and future work in a special subsection.

6.4.1 Evaluation of the solvers and the work in progress

The achieved computational times mentioned above prove that the used linear system solvers, both the PetSc implementation of the conjugate gradient and MA57 for matrix factorization, provide fully satisfactory performance.

The performance of the eigenvalue solvers is, in the current state of the FENNEC package development, the main field for enhancing the FENNEC package now, since Blzpack is worth to use only for first few iterations and both Blzpack and Jadamilu are only parallelized using multithreaded BLAS. Therefore, a more powerful or MPI parallelized solver would enlarge the set of problems, on which FENNEC is applicable.

We[20] have already done several steps in this field. The solvers have been incorporated into the system of FENNEC plugins, which allows switching among the solvers easily. Second, we created the "eigenvalue-solver-independent" interface to MA57 and to routines for Sherman-Morrison-Woodbury identity. Third, we have rewritten the interface for currently used solvers from Fortran to Cython, using the mentioned interfaces, and we have compared the performance of the old and the new implementation of the solvers, which proves that the new routines are a slightly faster than the original ones (mainly due to employing better routines for sparse matrix manipulation).

All these steps have been done to have a framework for an as-easy-as-possible incorporating new eigenvalue solvers into the FENNEC, the benchmark tests then show that the performance of our new routines is reasonable.

The first eigenvalue solver to be incorporated into the FENNEC package is FEAST solver [249]: this solver has been already used for some DFT calculations [250] and it is MPI parallelized. In the current stage of the development, the MPI version of the Feast performs multiple better than Jadamilu on the same number of processor cores (however, only on more cores). Unfortunately, it is not yet able to compute all eigenvalues in the interval, which leads to instability of the DFT cycle. Therefore, further research in this area will be required to overcome this problem and to incorporate the FEAST solver into FENNEC.

If the FEAST will show as a dead end, we want to try solvers from the SLEPc package [251] or a promising approach presented in [252].

[20]Substantial part of the work on this field has been done in cooperation with my younger college Vojtěch Fišer.

During the development of the new solvers interface, the not-fully-satisfactory performance of SciPy implementation of the sparse matrix-vector multiplication have been identified. Therefore, the implementation has been replaced by our own version, which performs better in some cases, and we plan to try and possibly incorporate even more either sophisticated sparse format whose authors claims that the formats allow faster multiplication [253, 254] or a sparse BLAS implementation provided by CPU manufacturers [255, 256].

However, although the solvers are one of the main fields that requires further development of the package, FENNEC is even in its current state capable of solving real scale problems, as it is demonstrated in the next chapter.

7. Graphene fragment deformation study

Contents

This chapter presents the results of the study of graphene fragment deformation, obtained using the FENNEC software package. The study has been performed during preparing a new grant application, in which the ab-initio computation will be linked with molecular dynamics to give more insight into the behavior of graphene and possibly other modern 2D materials.

This chapter has three sections. The first one contains theoretical motivation for the study. The theoretical background presented in this short section is based on the text of our (unpublished) grant appliance, which I am a co-author of, and which has been reused with approval of its main author Otakar Frank.

The next section contains the setup of the calculations and the results of the study are presented in the third section. Both the sections are original.

7.1 Motivation for the study

Graphene is a modern and promising material, frequently used in electronics, photonics, and optoelectronics, energy storage and conversion, or sensing [257]. Mechanical deformations can cause drastic changes of 2D materials properties — from (opto)electronic to magnetic, chemical, or mechanical [258, 259]. For example, the in-plane tension causes decreasing of a band gap energy, changing the character of the gap from direct to indirect (where direct gap means that excitation of the electron does not change its momentum, while indirect excitation leads to the change of the momentum being a source of phonons).

In addition, the deformation effects comprise also all kinds of topographical distortions like ripples, wrinkles, folds or crumples [260]. These corrugations are almost omnipresent because of the low bending rigidity of 2D materials and they form and change readily upon even minuscule stress. Nanometer-sized ripples cause a dramatic decrease in mobility. Larger wrinkles and folds can generate a plethora of effects like gauge fields, energy funneling, or quantum confinement. There is a dire need to control the appearance and dimensions of such structures. Non-homogeneous deformation gradients causing energy funneling [261] or deformation singularities causing single-photon emitters [262] can be created either statically using a properly designed substrate or dynamically using AFM nanoindentation. However, the theoretical description of such phenomena suffers from various levels of approximation.

The effects of strain on the crystal lattice of 2D materials can be monitored directly by Raman spectroscopy [263]: e.g. for in-plane compression and wrinkling [264], a coupling of phonons to the electronic structure [265], manipulating the interaction in twisted bilayers [266] or quantifying Fermi level shift together with shear and hydrostatic stress from the Raman spectra [267].

Naturally, the obtained information is limited by diffraction, collected from an area of several hundreds of nanometers in diameter. Even though it is possible to derive spatially better-resolved data by fine-step mapping, as have been shown for graphene suspended over nanopillars with 250nm separation [268], the limit still stays within hundreds of nanometers. To break the limit, Atomic Force Microscopy (AFM) or Scanning Tunneling Microscopy (STM) with plasmonically active tips (gold, silver) can be coupled to Raman spectroscope for Tip-Enhanced Raman Spectroscopy (TERS) or Photoluminescence (TEPL). The spatial resolution of these near-field methods is driven by the diameter of the tip.

The experimental approach can be viewed as global: only properties of a relatively big fragment of the material can be observed and the resolution of the measurement is limited. This limitation of the experimental approach for a better understanding of the properties of 2D materials can be overcome by the computational approach, which however has the opposite limitation: the computational demands rise sharply with the number of electrons and atomic sites in the sample under study. Thus, the computational approach provides an insight into “local” properties, limited by the computationally manageable size of the sample. The experimental data are available for the samples with millions of atoms, while the ab-initio calculations can provide answers up to hundreds or thousands of atoms. Therefore, another method is needed to fill the gap between experiment and ab-initio calculations.

The atomistic simulations by MD technique, e.g. [269, 270], can treat up to billions of atoms ($\approx$ volume of a few cubic microns). Advanced interaction potentials can have their parameters adjusted according to results of ab-initio (first principles) quantum mechanical calculations [271], as well as empirically according to important experimental data like elastic constants, vacancy formation energy, cohesion energy, etc., see [272]. Also those data can be, however, calculated from the first principles [273].

The classical atomistic MD methods are very successful in modeling various macroscopic mechanical properties [274], including the propagation of the acoustic waves and providing correct phonon frequencies, but — lacking electronic structure information — *in principle* cannot provide material properties linked with electronic states (i.e. electric and optical properties). Moreover, since this approach relies on semi-empirical parameterization, even the mechanical properties may not be reflected correctly in non-standard situations for which the interaction potentials have not been adjusted.

Therefore neither ab-inito approach nor molecular dynamics are sufficient to calculate all desired properties of such a large material structure. Our approach is to bi-directionally link the ab-initio calculations using FENNEC and molecular dynamics together: first, the local properties of the material will be computed using ab-initio calculations. The results will serve as input for molecular dynamics

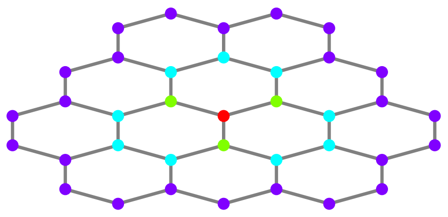


Figure 7.1: Deformed graphene fragment: spatial structure. The central red atom have been displaced, the positions of the green and cyan atoms (in the twelve optimized atoms case) have been optimized.

(for fitting potentials). Using molecular dynamics, the geometry of the material structure will be obtained — and this geometry will serve as input for next ab-initio calculations being able to determine various (e.g. the electric and optical) properties of the material.

The first steps of the mentioned approach are described in this chapter: it contains results of relaxation of deformed graphene fragment using software package FENNEC. The following calculations are the proof of concept: example geometry, where the forces acting on the central atom of the fragment are determined. The forces will be then (by our specialist on the molecular dynamics) fitted into the molecular dynamics potentials. The result will be self-consistent multiscale computation, where the data obtained using ab-initio calculations should serve as input for molecular dynamics, and conversely, the results of molecular dynamics should determine more realistic geometry structures for ab-initio calculations to update and refine the parameters used for generating molecular dynamics potentials.

7.2 Graphene layer deformation

To determine the force needed to tear off the graphene layer using a point-acting force, a geometry of graphene layer fragments has been optimized for various displacements of the central atom. The computed graphene layer fragment consists of 37 carbon atoms: there are included all the fourth neighbors of the central atoms so that each atom belongs to at least one complete hexagon (see Fig. 7.1).

The following results have been computed using hexahedral mesh (see chapter 3.2.1 and Fig. 7.2) and environment-reflecting pseudopotentials [20] with r_c radius 1.5 a.u. (see section 4.2.1). The force acting on the central atom has been computed as follows: the central atom has been displaced in the z-axis and his position has been fixed. Then a geometry optimization of the positions of his three or twelve nearest neighbors has been performed.

The setup of the study, of course, does not have the ambition to describe a real deformation of the graphene — of course, any real deformation would take a much larger area of the graphene than the one atom and computing of such a large sample exceeds the computational capabilities of any current ab-inito computational method.

The aim of this study is to prove that the FENNEC applied to the graphene is able to obtain physically reasonable results and thus they can be used as inputs for constructing molecular dynamic potentials. For the purpose, the case with three optimized atoms has been computed using well established ab-initio plane-wave DFT code ABINIT [16]. ABINIT setup differs a bit from FENNEC setup: since it is software for computing periodical structures, the computed sample has been infinite graphene layers in a cell of a size 5x5 atoms. The layers have been positioned 20 atomic units apart. For a better comparison of the results, the forces acting on the central atoms have been computed using FENNEC also in a fixed geometry setup, with the positions of atoms were obtained from the ABINIT calculation.

7.2.1 Results of the computation and their discussion

As we can see in Fig. 7.3, the result forces on the central atom obtained using FENNEC and ABINIT for the same geometry are similar. Since ABINIT is periodic code, whereas FENNEC compute finite graphene fragment, the boundary conditions are necessarily different in these two cases and a small difference of the force is expectable. Such a difference has been really observed — especially for a greater deformation, which disturbs the charge density to greater distances — but its magnitude seems to be reasonable.

A maximal observed force for ABINIT has been slightly greater (-0.206 $^{\mathrm{E_h}}/_{\mathrm{a.u.}}$) than the one for FENNEC (-0.183 $^{\mathrm{E_h}}/_{\mathrm{a.u.}}$). In the case of the fixed position, FENNEC, the maximum of the force is slightly shifted to a greater displacement of the central atom compared to the ABINIT results. When the positions of the atoms were relaxed using FENNEC, the three optimized atoms have been attracted back to the layer earlier (see Fig. 7.4), which results in a shift of the maxima of the force, its magnitude, however, remains nearly the same (in fact it grows a little). The differences of the forces on the three atoms in this region were very small, which allows the boundary conditions to affect more the results.

The differences of the forces obtained by optimizing the positions of three and twelve atoms are expectable: more “layers” of optimized atoms causes the force to

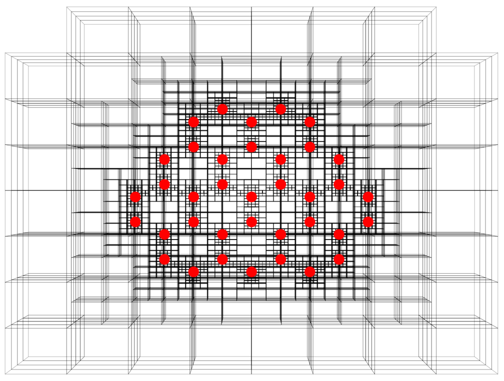
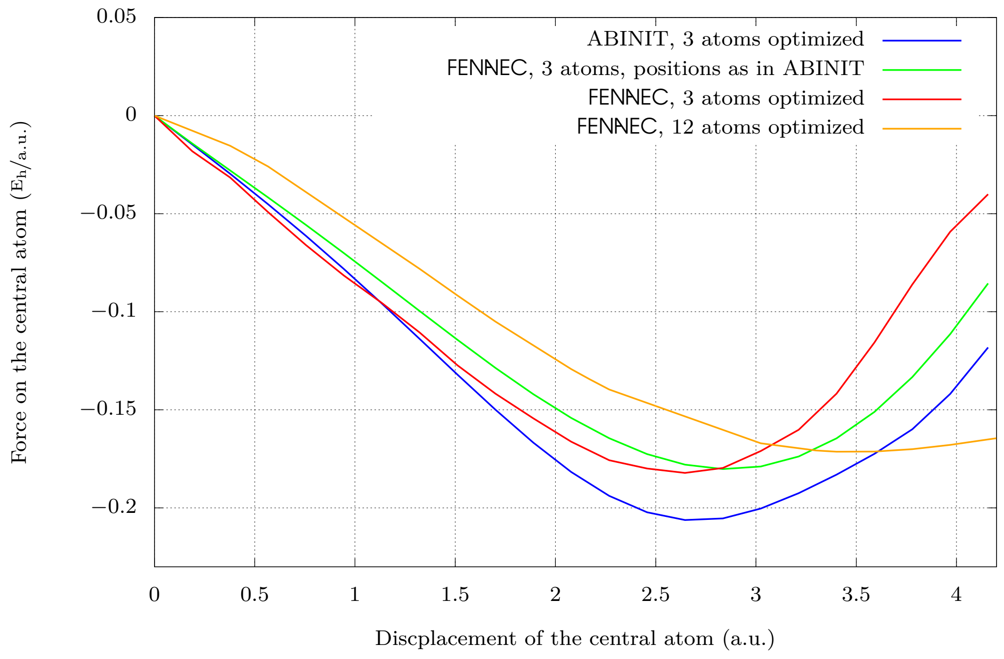


Figure 7.2: A mesh for the graphene fragment computation.

Figure 7.3: Computed forces on the central atom of a deformed graphene fragment.



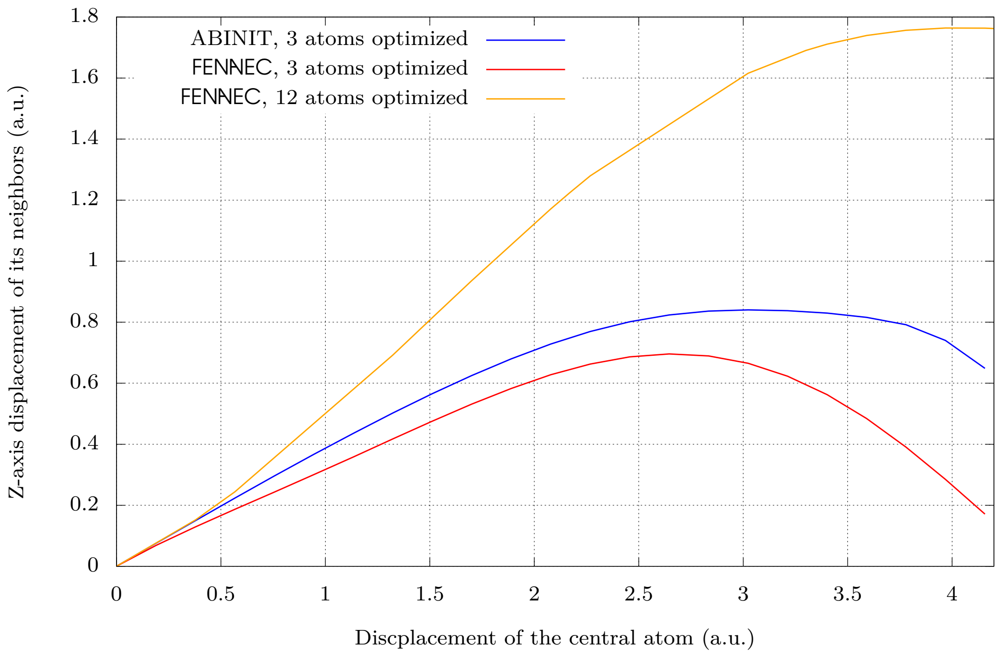
grow slower, allows the greater displacement of the central atom, and the maximal force acting on the central atom is a lower, as the charge density is more "diluted" in the maximum of the force due to greater spatial displacement.

The computations serve as proof of concept and demonstration of the capability of the FENNEC code to obtain reasonable forces from graphene fragment and to obtain experiences with the computation. The comparison with the ABINIT for the same geometry proves, that the results obtained with FENNEC are reasonable. Moreover, for further calculations and research, the following experiences and recommendations could be stated:

▷ The best mesh seems to be the one with a base cube size corresponding to the diameter of the hexagon: this choice allows the automatic adaptation algorithm (see algorithm 3.3 on page 70) to generate a mesh with as the same level of refinement around all atoms as possible (see Fig. 7.2).

▷ In the three atoms optimized case, the optimization has been performed using Anderson mixing (see section 2.5 on page 36). For the twelve atom case, this algorithm failed to optimize the positions properly. The Verlet algorithm (molecular dynamic with dampening) [275] performs better in this case, however, further improvement in the performance could be probably achieved by using a more sophisticated algorithm, e.g. some form [276]. A good trade-off between speed and stability [277] seems to offer a modification of Verlet scheme called FIRE [278, 279], often used for this purpose (e.g. [280]) is well-known BFGS method [79].

Figure 7.4: The displacement of the nearest neighbors of the central atom in the z-axis during the deformation of graphene fragment.



- ▷ Use of pseudopotentials tuned for the specific positions in the structure (we use a different pseudopotential for the tip and the three nearest atoms) leads to better results, same as a more-than-one-projector separable pseudopotential substantially lowers the differences between results obtained by ABINIT and FENNEC, which further confirms the results from section 4.3.

Conclusion of the study

To conclude FENNEC proves, that is able not only to compute reasonable forces for samples of a size that is necessary for constructing molecular dynamics potentials, but it is also able to perform geometry optimizations, which are more computationally demanding task than just force computing. The resulting forces differ a bit from the reference calculation, but the differences of the results are fully acceptable, taking into consideration the differences between the two methods and from them following the differences of the setups of calculations.

The performance of the package could be further improved by utilizing more sophisticated geometry optimization algorithms (and/or eigenvalue solvers, see the previous chapter), but even in the current state of development FENNEC is capable of solving real-scaled problems.

Figure 7.5: Deformation of graphene fragment: wire model of its relaxed structure and corresponding charge densities for the various displacement of the central (red) atom. Positions of all the other atoms except the border ones (blue) have been optimized to have zero HF-forces.

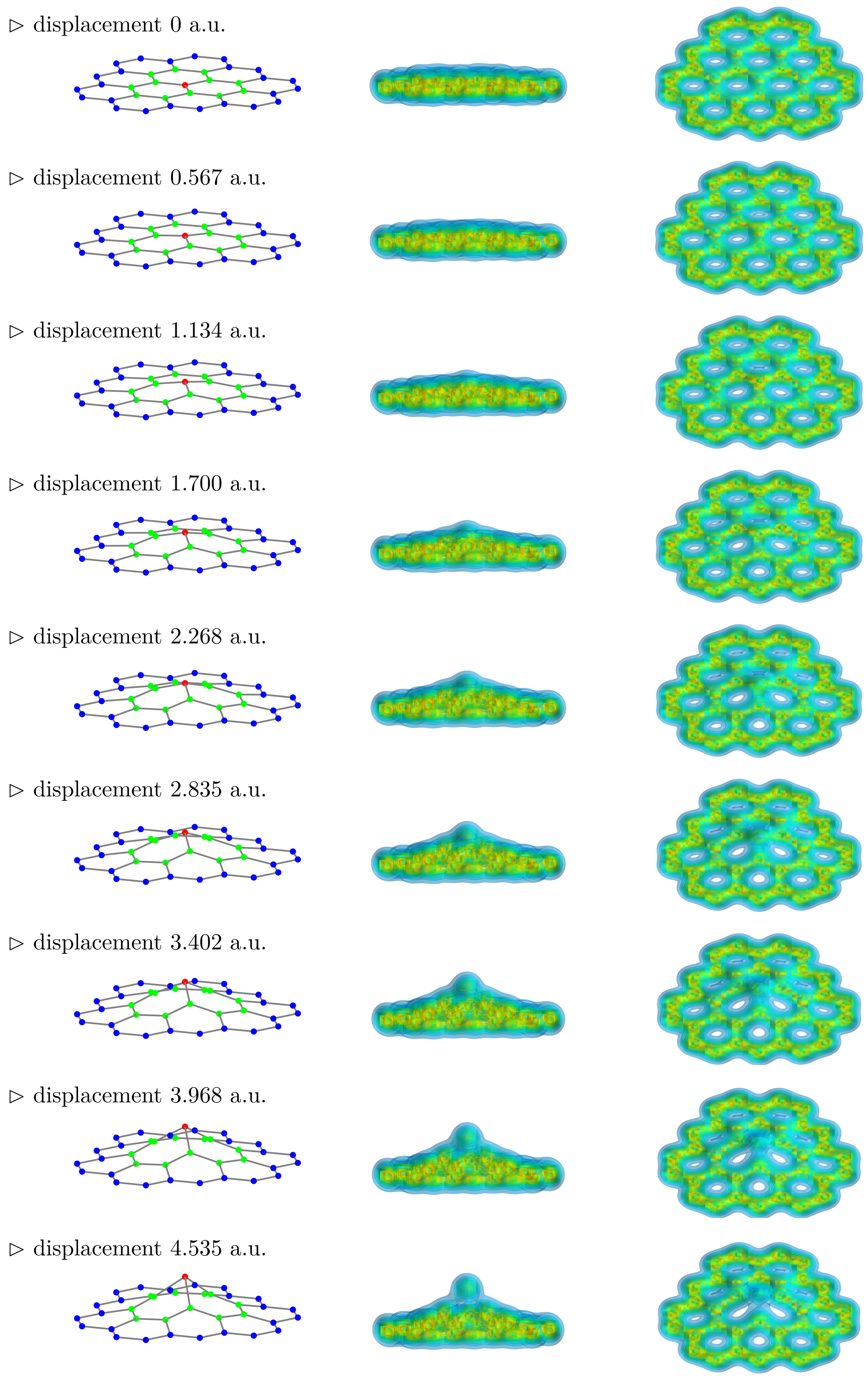
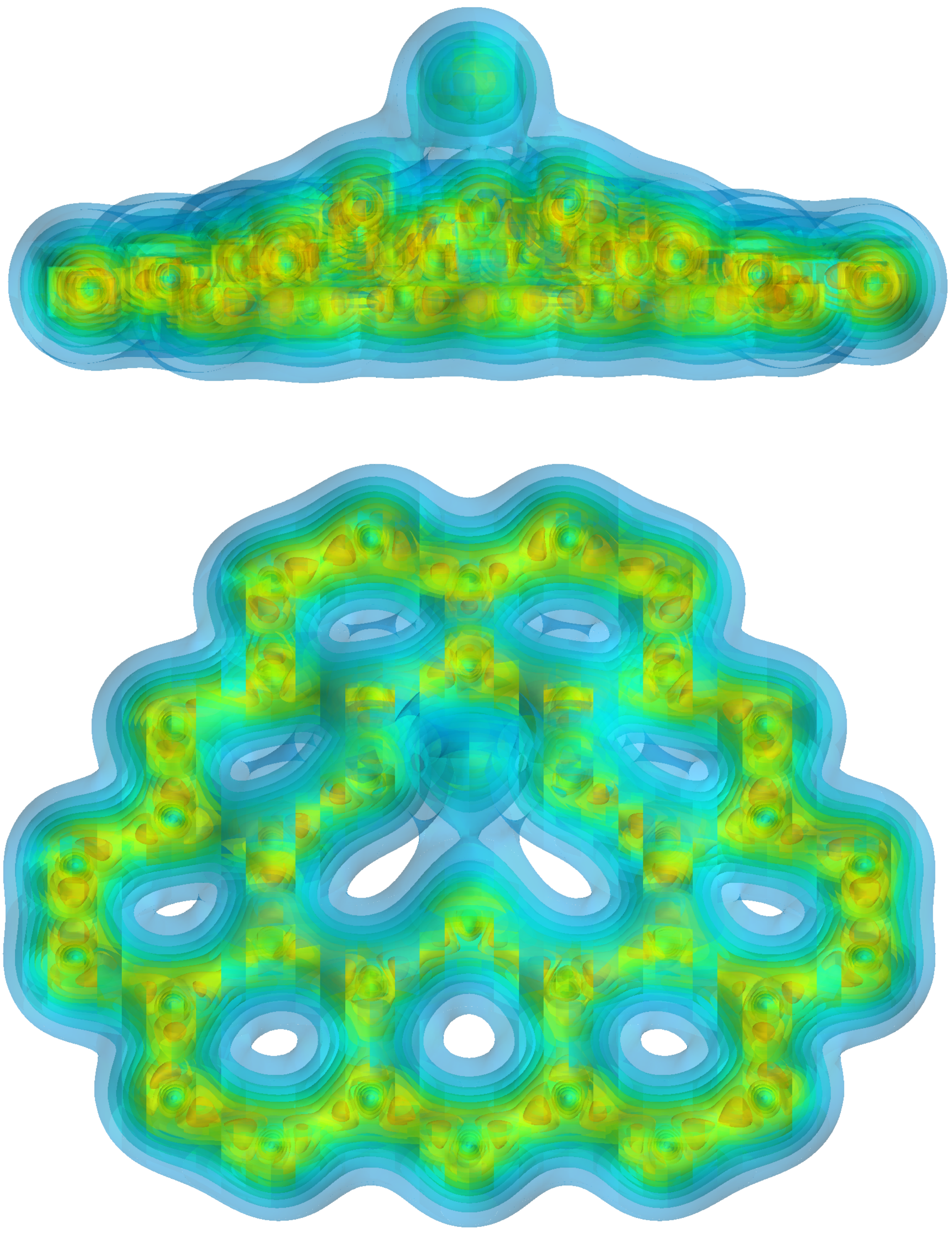

▷ displacement 0 a.u.
▷ displacement 0.567 a.u.
▷ displacement 1.134 a.u.
▷ displacement 1.700 a.u.
▷ displacement 2.268 a.u.
▷ displacement 2.835 a.u.
▷ displacement 3.402 a.u.
▷ displacement 3.968 a.u.
▷ displacement 4.535 a.u.

Figure 7.6: A deformed graphene fragment for the displacement of the central atom equal to 4.535 a.u. and twelve optimized atomic positions. Color layers denotes the levels of charge density. Notice the low density of charge between the central atom and its neighbors: it signals, that the bonds are nearly torned off and the attractive force have started to decline.



Conclusion

This thesis describes the development of a new method for electronic structure calculation, its implementation into the software package FENNEC and the application of the package to the problem of graphene fragment deformation.

The new method is designed for calculating aperiodic materials (even with charge neutrality violation) using a general basis and, by that, to fill the gap between the existing well-established methods.

FENNEC represents a new approach, but it is based on well-established elements: (i) the density functional theory (which replaces the multi-electron Schrödinger equation with a set of Kohn-Sham single-electron equation), (ii) discretization by the finite element method (FEM) providing a general basis and excellent convergence control, and (iii) environment-reflecting pseudopotentials.

Another possibility for the discretization that the newly proposed method offers is an isogeometric analysis (IGA) that provides continuous derivations up to the order of the basis minus one.

The thesis includes a suitability analysis of FEM and IGA bases for electronic structure calculations using FENNEC. The results show, that the best bases are tri-cubic FEM bases on hexahedra elements, especially with the use of refined meshes. Tri-quadratic bases can — in some refined cases — outperform tri-cubic ones, but their accuracy should be verified. IGA offers both smooth convergence and continuous derivatives at the costs of higher computational demands.

To obtain a computationally efficient expression of the discretized equations, a separable form of l-dependent pseudopotentials is used. The implementation of separable pseudopotentials into the discretized Kohn-Sham equations results in a rank-k-update generalized eigenvalue problem. This problem is solved by iterative solvers using the Sherman-Morrison-Woodbury formula. Since the early and later stages of the computation require different properties of the used solver, the new method utilizes two solvers, based on implementing different algorithms: the Lanczos algorithm and the Jacobi-Davidson algorithm. For a better parallelization and the overall performance of the package, the FEAST solver will be incorporated as a third possibility in the near future.

During the implementation of the method, many problems had to be solved and new techniques had to be developed for usage within the new framework: Formulas for the total energy of the system, and for gradients of the total energy called Hellmann-Feynman forces, were derived and implemented for efficient geometry optimizations. Whereas the obtained formula for the total energy does not differ very much from the formulas used by other authors, the derived formula for the Hellmann-Feynman forces is original. This formula presents very good properties as regards convergence, performance, and accuracy.

For efficient mixing in a DFT self-consistent cycle, a new mixing algorithm was developed. This adaptive-Anderson mixing provides an automatic adaptation of the mixing coefficient which would have to be guessed otherwise. This algorithm was tested not only in FENNEC but also in another DFT code based on KKR approach.

For transforming an l-dependent pseudopotentials into its separable form, a new algebraic description based on B-orthogonality has been developed, offering insight into the suitability of bases used for the transformation. The recommendations for the choice of the number of projectors were obtained and the option of including energy derivative into the basis was implemented and analyzed.

The entire method was implemented in Python using finite element framework SfePy [207] and many other numerical libraries into the FENNEC package, which is still being expanded. The package architecture is characterized by an open and extensible architecture based on plugins, which provides many benefits for developers and users. A short discussion of the package architecture and design principles is included in the thesis, and the crucial properties and parts of the package determining its performance are presented: the used linear algebra solvers and parallelism of the package.

FENNEC is a highly useful tool for computing electronic structure. This fact has been demonstrated on a sample computation of a deformed fragment of a graphene layer. The geometry of the fragment deformed by a point force acting on the central atom was determined and the force required for tearing the fragment was calculated. These computations will be used in further research for modeling molecular dynamics potentials. The ability of FENNEC to do a geometry optimization of real material samples proved that it is already a mature software, prepared for real use.

The main goal and the most important result of my doctoral studies is the FENNEC framework itself: a still developed but already mature software package for electronic structure calculations of aperiodic materials.

During its development, various results were published and presented, see the appendix J. Some of them can be highlighted, e.g.: the newly proposed formula for Hellmann-Feynman forces and its analysis [281] published in Computer Physics Communications [1], the convergence study of isogeometric analysis and finite element method applied on the Kohn-Sham equations [106] (in Applied Mathematics and Computation[2]) and the not yet published a new adaptive mixing scheme for a self-consistent problem, presented in the second chapter of this thesis (part of this work have been published in [282]).

There are several fields, where FENNEC could be further developed. First, the efficiency of FENNEC could be further improved by employing a more parallelized eigenvalue solver, as proposed in the sixth chapter. We also plan to incorporate routines for creating relativistic and spin-aware pseudopotentials which are already developed by our college Robert Cimrman. The implementation of (partial) periodic boundary conditions (e.g. for calculations on infinite strings) could be handy, too. We also consider enriching the SfePy package and FENNEC by p-adaptivity and/or by a refined version of isogeometric analysis.

[1]The journal is in Q1 quartile for the year 2019 both in General Physics and Astronomy and Hardware and Architecture according to Scopus (`https://www.scopus.com/sourceid/13184`, visited 20/6/2020).

[2]The journal is in Q1 quartile for the year 2019 both in Applied Mathematics and Computational Mathematics, according to Scopus (`https://www.scopus.com/sourceid/25170`, visited 20/6/2020).

Beside FENNEC improvements, the work on the implementation of the algorithm opens some interesting questions, which will be the subject of our future research. E.g. there is a possibility, that the smoothness of computed Hellmann-Feynman forces with a FEM basis could be further improved by incorporating the so-called IBS term (which is discussed in the fifth chapter), in spite of the fact that our basis is not dependent on atomic positions. The new adaptive-Anderson mixing algorithm, proposed in the third chapter, deserves a deeper theoretical background, which could lead to an even more efficient and robust adaptation scheme. The options for constructing the basis for transforming a potential into its separable form should be examined more thoroughly.

Research in all of the suggested fields (whose list is far from complete) is far beyond available capacity of our research group. Thus, we will need to incorporate new colleagues who would help us mainly in the applications of the FENNEC package to “real world” problems.

In the near future, we plan (in cooperation with other research groups) to use FENNEC for forces calculations in deformed graphene fragments and other 2D materials. These forces will be used for tuning up the Finis-Sinclair-like interaction potentials (see e.g. [283]) in molecular dynamics for a better understanding of the behavior of deformed layered material. Therefore, our effort will probably be aimed at improving the FENNEC efficiency (solvers and mixing algorithms) and at the routines for creating fully relativistic pseudopotentials, which will enable us to study a wider range of materials.

References

[1] Ira N Levine. *Quantum chemistry.* Prentice-Hall Of India Pvt. Limited, 2009.

[2] J Ihm, Alex Zunger, and Marvin L Cohen. "Momentum-space formalism for the total energy of solids". In: *Journal of Physics C: Solid State Physics* 12.21 (1979), p. 4409.

[3] JM Haile et al. "Molecular dynamics simulation: elementary methods". In: *Computers in Physics* 7.6 (1993), pp. 625–625.

[4] Rudolph Pariser and Robert G Parr. "A Semi-Empirical Theory of the Electronic Spectra and Electronic Structure of Complex Unsaturated Molecules. I." In: *The Journal of Chemical Physics* 21.3 (1953), pp. 466–471.

[5] Paul Adrien Maurice Dirac. *The principles of quantum mechanics.* Oxford university press, 1981.

[6] Jörg Liesen and Zdeněk Strakoš. *Krylov subspace methods: principles and analysis.* Oxford University Press, 2013.

[7] C. Pickard. "Hyperspatial optimization of structures". In: *Physical Review B* 99.5 (Feb. 2019), p. 054102. DOI: 10.1103/PhysRevB.99.054102.

[8] M. H. Kalos. "Monte Carlo Calculations of the Ground State of Three- and Four-Body Nuclei". In: *Phys. Rev.* 128 (4 Nov. 1962), pp. 1791–1795. DOI: 10.1103/PhysRev.128.1791. URL: https://link.aps.org/doi/10.1103/PhysRev.128.1791.

[9] Walter Kohn and Lu Jeu Sham. "Self-consistent equations including exchange and correlation effects". In: *Physical review* 140.4A (1965), A1133.

[10] C. Kittel. *Introduction to Solid State Physics.* Wiley, 2004. ISBN: 9780471415268. URL: https://books.google.cz/books?id=kym4QgAACAAJ.

[11] James P Lewis et al. "Advances and applications in the FIREBALL ab initio tight-binding molecular-dynamics formalism". In: *physica status solidi (b)* 248.9 (2011), pp. 1989–2007.

[12] Thom H Dunning Jr. "Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen". In: *The Journal of chemical physics* 90.2 (1989), pp. 1007–1023.

[13] Rici Yu, D Singh, and H Krakauer. "All-electron and pseudopotential force calculations using the linearized-augmented-plane-wave method". In: *Physical Review B* 43.8 (1991), p. 6411.

[14] Hans L Skriver. *The LMTO method: muffin-tin orbitals and electronic structure.* Vol. 41. Springer Science & Business Media, 2012.

[15] W.E. Pickett. *Pseudopotential Methods in Condensed Matter Applications.* Computer physics reports. North-Holland, 1989.

[16] X. Gonze et al. “Recent developments in the ABINIT software package”. In: *Comput. Phys. Commun.* 205 (Aug. 2016), pp. 106–131. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2016.04.003. URL: https://doi.org/10.1016/j.cpc.2016.04.003.

[17] Stewart J Clark et al. “First principles methods using CASTEP”. In: *Zeitschrift für Kristallographie-Crystalline Materials* 220.5/6 (2005), pp. 567–570.

[18] Claes Johnson. *Numerical solution of partial differential equations by the finite element method.* Courier Corporation, 2012.

[19] D. Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Elasticity Theory.* Cambridge University Press, 2007. ISBN: 9780521705189. URL: http://books.google.cz/books?id=PizECgOWoGgC.

[20] Jiří Vackář, Antonín Šimůnek, and Raimund Podloucky. “Ab initio pseudopotentials for interacting atoms”. In: *Phys. Rev. B* 53 (12 Mar. 1996), pp. 7727–7730. DOI: 10.1103/PhysRevB.53.7727. URL: https://link.aps.org/doi/10.1103/PhysRevB.53.7727.

[21] Steven R White, John W Wilkins, and Michael P Teter. “Finite-element method for electronic structure”. In: *Physical Review B* 39.9 (1989), p. 5819.

[22] James R Chelikowsky et al. “Higher-order finite-difference pseudopotential method: An application to diatomic molecules”. In: *Physical Review B* 50.16 (1994), p. 11355.

[23] Francois Gygi and Giulia Galli. “Real-space adaptive-coordinate electronic-structure calculations”. In: *Physical Review B* 52.4 (1995), R2229.

[24] Eiji Tsuchida and Masaru Tsukada. “Adaptive finite-element method for electronic-structure calculations”. In: *Physical Review B* 54.11 (1996), p. 7602.

[25] NA Modine, Gil Zumbach, and Efthimios Kaxiras. “Adaptive-coordinate real-space electronic-structure calculations for atoms, molecules, and solids”. In: *Physical Review B* 55.16 (1997), p. 10289.

[26] JE Pask et al. “Real-space local polynomial basis for solid-state electronic-structure calculations: A finite-element approach”. In: *Physical Review B* 59.19 (1999), p. 12352.

[27] J-L Fattebert and J Bernholc. “Towards grid-based $O(N)$ density-functional theory methods: Optimized nonorthogonal orbitals and multigrid acceleration”. In: *Physical Review B* 62.3 (2000), p. 1713.

[28] Yong-Hoon Kima, In-Ho Lee, and Richard M Martina. “Object-oriented construction of a multigrid electronic-structure code with Fortran 90”. In: *Computer Physics Communications* 131 (2000), pp. 10–25.

[29] In-Ho Lee, Yong-Hoon Kim, and Richard M Martin. “One-way multigrid method in electronic-structure calculations”. In: *Physical Review B* 61.7 (2000), p. 4397.

[30] JE Pask et al. “Finite-element methods in electronic-structure theory”. In: *Computer Physics Communications* 135.1 (2001), pp. 1–34.

[31] JE Pask and PA Sterne. “Finite element methods in ab initio electronic structure calculations”. In: *Modelling and Simulation in Materials Science and Engineering* 13.3 (2005), R71.

[32] Eric J Bylaska, Michael Holst, and John H Weare. “Adaptive Finite Element Method for Solving the Exact Kohn-Sham Equation of Density Functional Theory”. In: *Journal of chemical theory and computation* 5.4 (2009), pp. 937–948.

[33] Phanish Suryanarayana et al. “Non-periodic finite-element formulation of Kohn-Sham density functional theory”. In: *Journal of the Mechanics and Physics of Solids* 58.2 (2010), pp. 256–280.

[34] Phanish Suryanarayana et al. “Non-periodic finite-element formulation of Kohn–Sham density functional theory”. In: *J. Mech. Phys. Solids* 58.2 (Feb. 2010), pp. 256–280. ISSN: 0022-5096. DOI: 10.1016/j.jmps.2009.10.002. URL: https://doi.org/10.1016/j.jmps.2009.10.002.

[35] Jun Fang, Xingyu Gao, and Aihui Zhou. “A Kohn-Sham equation solver based on hexahedral finite elements”. In: *Journal of Computational Physics* 231.8 (2012), pp. 3166–3180.

[36] Gang Bao, Guanghui Hu, and Di Liu. “An h-adaptive finite element solver for the calculations of the electronic structures”. In: *Journal of Computational Physics* 231.14 (2012), pp. 4967–4979.

[37] Arif Masud and Raguraman Kannan. “B-splines and NURBS based finite element methods for Kohn-Sham equations”. In: *Computer Methods in Applied Mechanics and Engineering* 241 (2012), pp. 112–127.

[38] Volker Schauer and Christian Linder. “All-electron Kohn–Sham density functional theory on hierarchic finite element spaces”. In: *J. Comput. Phys.* 250 (Oct. 2013), pp. 644–664. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2013.04.020. URL: https://doi.org/10.1016/j.jcp.2013.04.020.

[39] Gang Bao, Guanghui Hu, and Di Liu. “Numerical Solution of the Kohn-Sham Equation by Finite Element Methods with an Adaptive Mesh Redistribution Technique”. In: *J Sci Comput* 55.2 (Sept. 2012), pp. 372–391. ISSN: 0885-7474, 1573-7691. DOI: 10.1007/s10915-012-9636-1. URL: https://doi.org/10.1007/s10915-012-9636-1.

[40] Phani Motamarri et al. “Higher-order adaptive finite-element methods for Kohn-Sham density functional theory”. In: *Journal of Computational Physics* 253 (2013), pp. 308–343.

[41] Gang Bao, Guanghui Hu, and Di Liu. “Numerical solution of the Kohn-Sham equation by finite element methods with an adaptive mesh redistribution technique”. In: *Journal of Scientific Computing* 55.2 (2013), pp. 372–391.

[42] Phani Motamarri and Vikram Gavini. “Subquadratic-scaling subspace projection method for large-scale Kohn-Sham density functional theory calculations using spectral finite-element discretization”. In: *Physical Review B* 90.11 (2014), p. 115127.

[43] Huajie Chen et al. “Adaptive Finite Element Approximations for Kohn–Sham Models”. In: *Multiscale Model. Simul.* 12.4 (Jan. 2014), pp. 1828–1869. ISSN: 1540-3459, 1540-3467. DOI: 10.1137/130916096. URL: https://doi.org/10.1137/130916096.

[44] Yvon Maday. “h — P Finite element approximation for full-potential electronic structure calculations”. In: *Chin. Ann. Math. Ser. B* 35.1 (Jan. 2014), pp. 1–24. ISSN: 0252-9599, 1860-6261. DOI: 10.1007/s11401-013-0819-3. URL: https://doi.org/10.1007/s11401-013-0819-3.

[45] Yvon Maday. “h-P Finite Element Approximation for Full-Potential Electronic Structure Calculations”. In: *Partial Differential Equations: Theory, Control and Approximation*. Springer, 2014, pp. 349–377.

[46] Huajie Chen et al. “Adaptive Finite Element Approximations for Kohn-Sham Models”. In: *Multiscale Modeling & Simulation* 12.4 (2014), pp. 1828–1869.

[47] Volker Schauer and Christian Linder. “The reduced basis method in all-electron calculations with finite elements”. In: *Advances in Computational Mathematics* 41.5 (2015), pp. 1035–1047.

[48] Eiji Tsuchida, Yoong-Kee Choe, and Takahiro Ohkubo. “An adaptive finite-element method for large-scale ab initio molecular dynamics simulations”. In: *Physical Chemistry Chemical Physics* 17.47 (2015), pp. 31444–31452.

[49] Denis Davydov, Toby D Young, and Paul Steinmann. “On the adaptive finite element analysis of the Kohn-Sham equations: methods, algorithms, and implementation”. In: *International Journal for Numerical Methods in Engineering* 11 (2016), pp. 863–888.

[50] John E Pask and N Sukumar. “Partition of unity finite element method for quantum mechanical materials calculations”. In: *Extreme Mechanics Letters* 11 (2017), pp. 8–17.

[51] Gaigong Zhang et al. “Adaptive local basis set for Kohn–Sham density functional theory in a discontinuous Galerkin framework II: Force, vibration, and molecular dynamics calculations”. In: *Journal of Computational Physics* 335 (Apr. 2017), pp. 426–443. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.12.052.

[52] Denis Davydov et al. “Convergence study of the h-adaptive PUM and the hp-adaptive FEM applied to eigenvalue problems in quantum mechanics”. In: *Adv. Model. and Simul. in Eng. Sci.* 4.1 (Dec. 2017), p. 7. ISSN: 2213-7467. DOI: 10.1186/s40323-017-0093-0. URL: https://doi.org/10.1186/s40323-017-0093-0.

[53] Susi Lehtola. “Fully numerical Hartree-Fock and density functional calculations. I. Atoms”. In: *arXiv:1810.11651 [physics]* (Oct. 2018). arXiv: 1810.11651. URL: http://arxiv.org/abs/1810.11651.

[54] Phani Motamarri and Vikram Gavini. “Configurational forces in electronic structure calculations using Kohn-Sham density functional theory”. In: *Phys. Rev. B* 97.16 (Apr. 2018), p. 165132. ISSN: 2469-9950, 2469-9969. DOI: 10.1103/physrevb.97.165132. URL: https://doi.org/10.1103/physrevb.97.165132.

[55] Phani Motamarri et al. “DFT-FE–A massively parallel adaptive finite-element code for large-scale density functional theory calculations”. In: *Computer Physics Communications* 246 (2020), p. 106853.

[56] Pierre Hohenberg and Walter Kohn. “Inhomogeneous electron gas”. In: *Physical review* 136.3B (1964), B864.

[57] Robert G Parr and Robert G Yang Weitao. *Density-functional theory of atoms and molecules*. Oxford University Press, 1995.

[58] Richard M Martin. *Electronic structure: basic theory and practical methods*. Cambridge university press, 2004.

[59] Kieron Burke. *The ABC of DFT*. 2007. URL: http://dft.uci.edu/book/gamma/g1.pdf.

[60] RO Jones. “Introduction to density functional theory and exchange-correlation energy functionals”. In: *Computational Nanoscience: Do It Yourself* 31 (2006), pp. 45–70.

[61] Claude Cohen-Tannoudji et al. *Quantum Mechanics (2 vol. set)*. 2006.

[62] Jiří Formánek. *Úvod do kvantové teorie I*. Akademia, Praha, 2004. ISBN: ISBN 80-200-1176-5.

[63] Richard P Feynman. “Feynman lectures on physics. Volume 3: Quantum mechancis”. In: *Reading, Ma.: Addison-Wesley, 1965, edited by Feynman, Richard P.; Leighton, Robert B.; Sands, Matthew* (1965).

[64] Yakir Aharonov and David Bohm. “Significance of electromagnetic potentials in the quantum theory”. In: *Physical Review* 115.3 (1959), p. 485.

[65] Lev Vaidman. “Role of potentials in the Aharonov-Bohm effect”. In: *Phys. Rev. A* 86 (4 Oct. 2012), p. 040101. DOI: 10.1103/PhysRevA.86.040101. URL: https://link.aps.org/doi/10.1103/PhysRevA.86.040101.

[66] Erwin Schrödinger. “Quantisierung als eigenwertproblem”. In: *Annalen der physik* 385.13 (1926), pp. 437–490.

[67] Michela Massimi. *Pauli’s exclusion principle: The origin and validation of a scientific principle*. Cambridge University Press, 2005.

[68] Werner Kutzelnigg. “Density functional theory in terms of a Legendre transformation for beginners”. In: *Journal of Molecular Structure: THEOCHEM* 768.1 (2006), pp. 163–173.

[69] Helmut Eschrig. *The fundamentals of density functional theory*. Vol. 32. Springer.

[70] P Haynes. “Linear-scaling methods in ab initio quantum-mechanical calculations”. PhD thesis. University of Cambridge, 1998.

[71] M Städele et al. “Exact exchange Kohn-Sham formalism applied to semiconductors”. In: *Physical Review B* 59.15 (1999), p. 10031.

[72] John P. Perdew et al. “Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation”. In: *Phys. Rev. B* 46 (11 Sept. 1992), pp. 6671–6687. DOI: `10.1103/PhysRevB.46.6671`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.46.6671`.

[73] Kieron Burke, John P Perdew, and Matthias Ernzerhof. “Why the generalized gradient approximation works and how to go beyond it”. In: *International journal of quantum chemistry* 61.2 (1997), pp. 287–293.

[74] E. Engel, D. Ködderitzsch, and H. Ebert. “Exact exchange in relativistic spin-density-functional theory: Exchange splitting versus spin-orbit coupling”. In: *Phys. Rev. B* 78 (23 Dec. 2008), p. 235123. DOI: `10.1103/PhysRevB.78.235123`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.78.235123`.

[75] Susi Lehtola et al. “Recent developments in libxc—A comprehensive library of functionals for density functional theory”. In: *SoftwareX* 7 (2018), pp. 1–5.

[76] Ulf von Barth and Lars Hedin. “A local exchange-correlation potential for the spin polarized case. i”. In: *Journal of Physics C: Solid State Physics* 5.13 (1972), p. 1629.

[77] MPAT Methfessel and AT Paxton. “High-precision sampling for Brillouin-zone integration in metals”. In: *Physical Review B* 40.6 (1989), p. 3616.

[78] June Gunn Lee. *Computational materials science: an introduction.* Crc Press, 2016.

[79] Roger Fletcher. *Practical methods of optimization.* John Wiley & Sons, 2013.

[80] Péter Pulay. “Convergence acceleration of iterative sequences. The case of SCF iteration”. In: *Chemical Physics Letters* 73.2 (1980), pp. 393–398.

[81] Duane D. Johnson. “Modified Broyden’s method for accelerating convergence in self-consistent calculations”. In: *Physical review. B, Condensed matter* 38 (18 Jan. 1989), pp. 12807–12813. DOI: `10.1103/PhysRevB.38.12807`.

[82] Donald G Anderson. “Iterative procedures for nonlinear integral equations”. In: *Journal of the ACM (JACM)* 12.4 (1965), pp. 547–560.

[83] V Eyert. “A comparative study on methods for convergence acceleration of iterative vector sequences”. In: *Journal of Computational Physics* 124.2 (1996), pp. 271–285.

[84] Haw-ren Fang and Yousef Saad. “Two classes of multisecant methods for nonlinear acceleration”. In: *Numerical Linear Algebra with Applications* 16.3 (2009), pp. 197–221.

[85] DePrince research group. *Programming Projects: DIIS convergence acceleration in SCF.* URL: `https://www.chem.fsu.edu/~deprince/programming_projects/diis/` (visited on 12/09/2019).

[86] M Mešina. “Convergence acceleration for the iterative solution of the equations X= AX+ f”. In: *Computer Methods in Applied Mechanics and Engineering* 10.2 (1977), pp. 165–173.

[87] Charles G Broyden. “A class of methods for solving nonlinear simultaneous equations”. In: *Math. Comput.* 19.92 (1965), pp. 577–593. DOI: 10.1090/S0025-5718-1965-0198670-6.

[88] Amartya S Banerjee, Phanish Suryanarayana, and John E Pask. “Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations”. In: *Chemical Physics Letters* 647 (2016), pp. 31–35.

[89] DR Bowler and MJ Gillan. “An efficient and robust technique for achieving self consistency in electronic structure calculations”. In: *Chemical Physics Letters* 325.4 (2000), pp. 473–476.

[90] Lin Lin and Chao Yang. “Elliptic preconditioner for accelerating the self-consistent field iteration in Kohn–Sham density functional theory”. In: *SIAM Journal on Scientific Computing* 35.5 (2013), S277–S298.

[91] Marcus Heide and Tomoya Ono. “Convergence of the Broyden Density Mixing Method in Noncollinear Magnetic Systems”. In: *Journal of the Physical Society of Japan* 82.11 (2013), p. 114706.

[92] G. P. Kerker. “Efficient iteration scheme for self-consistent pseudopotential calculations”. In: *Physical Review B* 23 (6 Mar. 1981). DOI: 10.1103/PhysRevB.23.3082. URL: http://gen.lib.rus.ec/scimag/index.php?s=10.1103/PhysRevB.23.3082.

[93] Yuzhi Zhou et al. “Applicability of Kerker preconditioning scheme to the self-consistent density functional theory calculations of inhomogeneous systems”. In: *Physical Review E* 97.3 (2018), p. 033305.

[94] Raffaele Resta. “Thomas-Fermi dielectric screening in semiconductors”. In: *Phys. Rev. B* 16 (6 Sept. 1977), pp. 2717–2722. DOI: 10.1103/PhysRevB.16.2717. URL: https://link.aps.org/doi/10.1103/PhysRevB.16.2717.

[95] H Ebert, D Ködderitzsch, and J Minár. “Calculating condensed matter properties using the KKR-Green’s function method—recent developments and applications”. In: *Reports on Progress in Physics* 74.9 (Aug. 2011), p. 096501. DOI: 10.1088/0034-4885/74/9/096501. URL: https://doi.org/10.1088%2F0034-4885%2F74%2F9%2F096501.

[96] Pauli Virtanen et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature methods* 17.3 (2020), pp. 261–272.

[97] Wenqing Ouyang et al. “Nonmonotone Globalization for Anderson Acceleration Using Adaptive Regularization”. In: *arXiv preprint arXiv:2006.02559* (2020).

[98] Donald GM Anderson. “Comments on “Anderson Acceleration, Mixing and Extrapolation””. In: *Numerical Algorithms* 80.1 (2019), pp. 135–234.

[99] Michael J Borden et al. “Isogeometric finite element data structures based on Bézier extraction of NURBS”. In: *International Journal for Numerical Methods in Engineering* 87.1-5 (2011), pp. 15–47.

[100] Robert Cimrman. “Enhancing SfePy with isogeometric analysis”. In: *arXiv preprint arXiv:1412.6407* (2014).

[101] Robert Cimrman et al. “Isogeometric analysis in electronic structure calculations”. In: *Mathematics and Computers in Simulation* 145 (2018), pp. 125–135.

[102] G. Dhatt and G. Touzot. *The finite element method displayed.* (A Wiley-Interscience publication). Wiley, 1984. ISBN: 9780471901105. URL: `http://books.google.cz/books?id=PZBRAAAAMAAJ`.

[103] P.G. Ciarlet and J.L. Lions. *Finite element methods.* Handbook of numerical analysis / general eds.: P. G. Ciarlet Vol. North-Holland, 1991. ISBN: 9780444703651. URL: `http://books.google.cz/books?id=nyknAQAAIAAJ`.

[104] Alexander Ženíšek. *Metoda konečných prvků.* 1999.

[105] Klaus-Jürgen Bathe. *Finite element procedures.* Klaus-Jurgen Bathe, 2006.

[106] Robert Cimrman et al. “Convergence study of isogeometric analysis based on Bézier extraction in electronic structure calculations”. In: *Applied Mathematics and Computation* 319 (2018), pp. 138–152.

[107] Jean Gallier and Jean H Gallier. *Curves and surfaces in geometric modeling: theory and algorithms.* Morgan Kaufmann, 2000.

[108] Gerald Farin. *Curves and surfaces for computer-aided geometric design: a practical guide.* Elsevier, 2014.

[109] Robert A Adams and John JF Fournier. *Sobolev spaces.* Elsevier, 2003.

[110] Robert Cimrman. *Notes on solving PDEs by the Finite Element Method.* URL: `http://sfepy.org/doc-devel/solving_pdes_by_fem.html` (visited on 09/01/2019).

[111] Jiří Křen and Josef Rosenberg. *Mechanika kontinua.* Západočeská univerzita, 2002.

[112] Jonathan Shewchuk. “What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint)”. In: *University of California at Berkeley* 73 (2002), p. 137.

[113] J Papež, J Liesen, and Z Strakoš. “Distribution of the discretization and algebraic error in numerical solution of partial differential equations”. In: *Linear Algebra and its Applications* 449 (2014), pp. 89–114.

[114] Cécile Dobrzynski and Pascal Frey. “Anisotropic Delaunay mesh adaptation for unsteady simulations”. In: *Proceedings of the 17th international Meshing Roundtable.* Springer, 2008, pp. 177–194.

[115] Joseph E. Flaherty. *Finite element analysis.* 2000. URL: `http://www.cs.rpi.edu/~flaherje/` (visited on 09/05/2019).

[116] Paolo Di Stolfo et al. "An easy treatment of hanging nodes in hp-finite elements". In: *Finite Elements in Analysis and Design* 121 (2016), pp. 101–117.

[117] Andy J Wathen. "Preconditioning". In: *Acta Numerica* 24 (2015), pp. 329–376.

[118] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

[119] Michele Benzi. "Preconditioning techniques for large linear systems: a survey". In: *Journal of computational Physics* 182.2 (2002), pp. 418–477.

[120] Maria Elizabeth G Ong. "Uniform refinement of a tetrahedron". In: *SIAM Journal on Scientific Computing* 15.5 (1994), pp. 1134–1144.

[121] Der-Tsai Lee and Bruce J Schachter. "Two algorithms for constructing a Delaunay triangulation". In: *International Journal of Computer & Information Sciences* 9.3 (1980), pp. 219–242.

[122] Oleg Golubitsky, Vadim Mazalov, and Stephen M Watt. "An algorithm to compute the distance from a point to a simplex". In: *ACM Commun. Comput. Algebra* 46 (2012), pp. 57–57.

[123] Kenneth H Huebner et al. *The finite element method for engineers.* John Wiley & Sons, 2001.

[124] Philippe G Ciarlet. *Handbook of Numerical Analysis: Solution of Equations in Rn (Prt 4), Techniques of Scientific Computer (Part 4), Numerical Methods for Fluids.* Vol. 8. Gulf Professional Publishing, 2001.

[125] H. S. M. Coxeter. *Introduction to Geometry, Second Edition.* John Wiley & Sons, 1969.

[126] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching". In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[127] Eugene L. Wachspress. *A rational finite element basis.* University of Southern California, 1975.

[128] Michael S Floater. "Generalized barycentric coordinates and applications". In: *Acta Numerica* 24 (2015), pp. 161–214.

[129] D Yoder. *B4wind user's Guide, NASA.* 2016.

[130] Francis Dominic Murnaghan. *Finite deformation of an elastic solid.* Wiley, 1951.

[131] OC Zienkiewicz and Robert Leroy Taylor. *The finite element method: the basis.* World Books Publishing Corporation, 2005.

[132] Gui-Rong Liu and Siu Sin Quek. *The finite element method: a practical course.* Butterworth-Heinemann, 2013.

[133] I Fried. "Accuracy and condition of curved (isoparametric) finite elements". In: *JSV* 31.3 (1973), pp. 345–355.

[134] Zhaoliang Meng, Zhongxuan Luo, and Xinchen Zhou. "A stable nonconforming finite element on hexahedra". In: *International Journal for Numerical Methods in Engineering* 109.5 (2017), pp. 611–630.

[135] Pavel Šolín, Jakub Červený, and Ivo Doležel. “Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM”. In: *Mathematics and Computers in Simulation* 77.1 (2008), pp. 117–132.

[136] Philippe G Ciarlet. *The finite element method for elliptic problems.* SIAM, 2002.

[137] Michel Crouzeix and P-A Raviart. “Conforming and nonconforming finite element methods for solving the stationary Stokes equations I”. In: *Revue française d’automatique informatique recherche opérationnelle. Mathématique* 7.R3 (1973), pp. 33–75.

[138] Zhong-Ci Shi. “Nonconforming finite element methods”. In: *Journal of computational and applied mathematics* 149.1 (2002), pp. 221–225.

[139] Youcef Saad. “SPARSKIT: A basic tool kit for sparse matrix computations”. In: (1990).

[140] Iain S Duff. “MA57—a code for the solution of sparse symmetric definite and indefinite systems”. In: *ACM Transactions on Mathematical Software (TOMS)* 30.2 (2004), pp. 118–144.

[141] Shangyou Zhang. “On the full C1-Qk finite element spaces on rectangles and cuboids”. In: *Adv. Appl. Math. Mech* 2.6 (2010), pp. 701–721.

[142] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, 2009.

[143] Dominik Schillinger, Praneeth K Ruthala, and Lam H Nguyen. “Lagrange extraction and projection for NURBS basis functions: A direct link between isogeometric and standard nodal finite element formulations”. In: *International Journal for Numerical Methods in Engineering* 108.6 (2016), pp. 515–534.

[144] Eugene V Shikin and Alexander I Plis. *Handbook on Splines for the User.* CRC Press, 1995.

[145] Isaac J Schoenberg. “Spline functions and the problem of graduation”. In: *Proceedings of the National Academy of Sciences* 52.4 (1964), pp. 947–950.

[146] Les Piegl and Wayne Tiller. *The NURBS book.* Springer Science & Business Media, 2012.

[147] Dalibor Martišek and Jana Procházková. “Relation between algebraic and geometric view on nurbs tensor product surfaces”. In: *Applications of mathematics* 55.5 (2010), pp. 419–430.

[148] Robin Bouclier, Jean-Charles Passieux, and Michel Salaün. “Local enrichment of NURBS patches using a non-intrusive coupling strategy: Geometric details, local refinement, inclusion, fracture”. In: *Computer Methods in Applied Mechanics and Engineering* 300 (2016), pp. 1–26.

[149] Thanh Ngan Nguyen. “Isogeometric Finite Element Analysis based on Bezier Extraction of NURBS and T-Splines”. MA thesis. Norges teknisk-naturvitenskapelige universitet, Fakultet for ingeniørvitenskap og teknologi, Institutt for konstruksjonsteknikk, 2011.

[150] Robert Cimrman, Vladimír Lukeš, and Eduard Rohan. “Multiscale finite element calculations in Python using SfePy”. In: *Advances in Computational Mathematics* (2019). ISSN: 1572-9044. DOI: `10.1007/s10444-019-09666-0`. URL: `https://doi.org/10.1007/s10444-019-09666-0`.

[151] Małgorzata Stojek. “Least-squares Trefftz-type elements for the Helmholtz equation”. In: *International journal for numerical methods in engineering* 41.5 (1998), pp. 831–849.

[152] Vít Dolejší et al. *Finite element methods: theory, applications and implementation.* 2011.

[153] *MoFEM, Basics of hierarchical approximation.* 2019. URL: `http://mofem.eng.gla.ac.uk/mofem/html/hierarchical_approximation_1.html#hierarchical_approximation_introduction` (visited on 12/03/2019).

[154] OC Zienkiewicz et al. *Hierarchical Finite Element Approaches Error Estimates and Adaptive Refinement.* Tech. rep. MARYLAND UNIV COLLEGE PARK INST FOR PHYSICAL SCIENCE and TECHNOLOGY, 1981.

[155] GF Carey and E Barragy. “Basis function selection and preconditioning high degree finite element and spectral methods”. In: *BIT Numerical Mathematics* 29.4 (1989), pp. 794–804.

[156] Ondřej Čertík. “Calculation of Electron Structure in the Framework of DFT in Real Space”. MA thesis. Karlova Univerzita, 2008.

[157] Adam Kiejna et al. “Comparison of the full-potential and frozen-core approximation approaches to density-functional calculations of surfaces”. In: *Physical Review B* 73.3 (2006), p. 035404.

[158] Enrico Fermi. “Sopra lo spostamento per pressione delle righe elevate delle serie spettrali”. In: *Il Nuovo Cimento (1924-1942)* 11.3 (1934), pp. 157–166. URL: `http://fisica.unipv.it/percorsi/pdf/NCFermi1934_11_157.pdf`.

[159] Hans Hellmann. “A new approximation method in the problem of many electrons”. In: *The Journal of Chemical Physics* 3.1 (1935), pp. 61–61.

[160] Leonard Kleinman and D. M. Bylander. “Efficacious Form for Model Pseudopotentials”. In: *Phys. Rev. Lett.* 48 (20 May 1982), pp. 1425–1428. DOI: `10.1103/PhysRevLett.48.1425`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.48.1425`.

[161] Jiří Vackář, Marek Hyťha, and Antonín Šimůnek. “All-electron pseudopotentials”. In: *Physical Review B* 58.19 (1998), p. 12712.

[162] Bo J Austin, V Heine, and LJ Sham. “General theory of pseudopotentials”. In: *Physical Review* 127.1 (1962), p. 276.

[163] Antonín Šimůnek and Jiří Vackář. “Correlation between core-level shift and bulk modulus in transition-metal carbides and nitrides”. In: *Physical Review B* 64.23 (2001), p. 235115.

[164] Jiří Vackář and Antonín Šimůnek. “Adaptability and accuracy of all-electron pseudopotentials”. In: *Physical Review B* 67.12 (2003), p. 125113.

[165] DR Hamann, M Schlüter, and C Chiang. “Norm-conserving pseudopotentials”. In: *Physical Review Letters* 43.20 (1979), p. 1494.

[166] Richard Courant and David Hilbert. *Methods of mathematical physics*. Vol. 1. CUP Archive, 1965.

[167] Jakub Višňák. *Matematický úvod a identity používané pro studium funkcionálu střední kvadratické fluktuace energie v an initio kvantově mechanických výpočtech*. 2009.

[168] Isaiah Lankham, Bruno Nachtergaele, and Anne Schilling. *Linear algebra as an introduction to abstract mathematics*. World Scientific, 2016.

[169] Peter E Blöchl. “Generalized separable potentials for electronic-structure calculations”. In: *Physical Review B* 41.8 (1990), p. 5414.

[170] MY Chou. “Reformulation of generalized separable pseudopotentials”. In: *Physical Review B* 45.20 (1992), p. 11465.

[171] Intel. *Intel® Xeon® Processor E5645*. 2010. URL: `https://ark.intel.com/content/www/us/en/ark/products/48768/intel-xeon-processor-e5645-12m-cache-2-40-ghz-5-86-gt-s-intel-qpi.html` (visited on 05/20/2019).

[172] Lin-Wang Wang and Alex Zunger. “Large scale electronic structure calculations using the Lanczos method”. In: *Computational Materials Science* 2.2 (1994), pp. 326–340.

[173] Erricos John Kontoghiorghes. *Handbook of parallel computing and statistics*. CRC Press, 2005.

[174] Osni A Marques. “BLZPACK: Description and user’s guide”. In: *CERFACS, Toulouse, France* (1995).

[175] J. E. Pask and P. A. Sterne. “Real-space formulation of the electrostatic potential and total energy of solids”. In: *Phys. Rev. B* 71 (11 Mar. 2005), p. 113101. DOI: `10.1103/PhysRevB.71.113101`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.71.113101`.

[176] José M Soler et al. “The SIESTA method for ab initio order-N materials simulation”. In: *Journal of Physics: Condensed Matter* 14.11 (2002), p. 2745.

[177] Richard Phillips Feynman. “Forces in molecules”. In: *Physical Review* 56.4 (1939), p. 340.

[178] David Carfì. “The pointwise Hellmann-Feynman theorem”. In: *Atti della Accademia Peloritana dei Pericolanti-Classe di Scienze Fisiche, Matematiche e Naturali* 88.1 (2010).

[179] Peter Pulay. “Ab initio calculation of force constants and equilibrium geometries in polyatomic molecules: I. Theory”. In: *Molecular Physics* 17.2 (1969), pp. 197–204.

[180] Bernd Kohler et al. “Force calculation and atomic-structure optimization for the full-potential linearized augmented plane-wave code WIEN”. In: *Computer physics communications* 94.1 (1996), pp. 31–48.

[181] Fabien Tran et al. “Force calculation for orbital-dependent potentials with FP-(L) APW+ lo basis sets”. In: *Computer Physics Communications* 179.11 (2008), pp. 784–790.

[182] Georg Kresse. “VASP the Guide”. In: *http://cms. mpi. univie. ac. at/vasp/* (2007).

[183] Mike Gillan. “Forces and Stresses”. In: *Nuts and Bolts of first-principles simulations, Castep Workshop, Durham University.* 2001.

[184] Mike Gillan. “Density functional theory”. In: *Nuts and Bolts of first-principles simulations, Castep Workshop, Durham University.* 2001.

[185] Paul Bendt and Alex Zunger. “Simultaneous relaxation of nuclear geometries and electric charge densities in electronic structure theories”. In: *Physical Review Letters* 50.21 (1983), p. 1684.

[186] Dominik Marx and Jürg Hutter. *Ab initio molecular dynamics: basic theory and advanced methods.* Cambridge University Press, 2009.

[187] A Badinski and RJ Needs. “Accurate forces in quantum Monte Carlo calculations with nonlocal pseudopotentials”. In: *Physical Review E* 76.3 (2007), p. 036707.

[188] NN Greenwood and A Earnshaw. *Chemistry of the Elements 2nd Edition.* Butterworth-Heinemann, 1997.

[189] Xiao-Gang Wang, Edwin L Sibert III, and Jan ML Martin. “Anharmonic force field and vibrational frequencies of tetrafluoromethane (CF 4) and tetrafluorosilane (SiF 4)”. In: *The Journal of Chemical Physics* 112.3 (2000), pp. 1353–1366.

[190] Burak Himmetoglu et al. “Hubbard-corrected DFT energy functionals: The LDA+ U description of correlated systems”. In: *International Journal of Quantum Chemistry* 114.1 (2014), pp. 14–49.

[191] Steven P Lewis et al. “Efficient scaling of calculations involving separable nonlocal potentials”. In: *Physical Review B* 58.7 (1998), p. 3482.

[192] K-M Ho, C-L Fu, and BN Harmon. “Microscopic analysis of interatomic forces in transition metals with lattice distortions”. In: *Physical Review B* 28.12 (1983), p. 6687.

[193] Xiaodun Jing et al. “Ab initio molecular-dynamics simulations of Si clusters using the higher-order finite-difference-pseudopotential method”. In: *Physical Review B* 50.16 (1994), p. 12234.

[194] MMG Alemany et al. “Real-space pseudopotential method for computing the electronic properties of periodic systems”. In: *Physical Review B* 69.7 (2004), p. 075101.

[195] T Miyazaki et al. “Atomic force algorithms in density functional theory electronic-structure techniques based on local orbitals”. In: *The Journal of chemical physics* 121.13 (2004), pp. 6186–6194.

[196] Dmitry Novoselov, Dm M Korotin, and VI Anisimov. “Hellmann–Feynman forces within the DFT+U in Wannier functions basis”. In: *Journal of Physics: Condensed Matter* 27.32 (2015), p. 325602.

[197] Jerzy Bernholc, Miroslav Hodak, and Wenchang Lu. “Recent developments and applications of the real-space multigrid method”. In: *Journal of Physics: Condensed Matter* 20.29 (2008), p. 294205.

[198] Miroslav Hodak et al. “Implementation of ultrasoft pseudopotentials in large-scale grid-based electronic structure calculations”. In: *Physical Review B* 76.8 (2007), p. 085108.

[199] J-L Teffo, ON Sulakshina, and VI Perevalov. “Effective Hamiltonian for rovibrational energies and line intensities of carbon dioxide”. In: *Journal of Molecular Spectroscopy* 156.1 (1992), pp. 48–64.

[200] T Shimanouchi. “Tables of Molecular Vibrational Frequencies Consolidated Volume I; National Bureau of Standards: Washington, DC, 1972”. In: *Google Scholar* (2005), pp. 1–160.

[201] Wai-Leung Yim et al. “Vibrational behavior of adsorbed CO_2 on single-walled carbon nanotubes”. In: *The Journal of chemical physics* 120.11 (2004), pp. 5377–5386.

[202] J Russell Thomas et al. “The balance between theoretical method and basis set quality: A systematic study of equilibrium geometries, dipole moments, harmonic vibrational frequencies, and infrared intensities”. In: *The Journal of chemical physics* 99.1 (1993), pp. 403–416.

[203] A Spielfiedel et al. “Bent valence excited states of CO2”. In: *The Journal of chemical physics* 97.11 (1992), pp. 8382–8388.

[204] GT Aldoshin and SP Yakovlev. “Analytic model of vibrations of a carbon dioxide molecule. Fermi resonance”. In: *Mechanics of Solids* 50.1 (2015), pp. 33–43.

[205] VI Lebedev and DN Laikov. “A quadrature formula for the sphere of the 131st algebraic order of accuracy”. In: *Doklady. Mathematics.* Vol. 59. 3. MAIK Nauka/Interperiodica. 1999, pp. 477–481.

[206] S. Behnel et al. “Cython: The Best of Both Worlds”. In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 31–39. ISSN: 1521-9615. DOI: 10.1109/MCSE.2010.118.

[207] Robert Cimrman. “SfePy - Write Your Own FE Application”. In: *Proceedings of the 6th European Conference on Python in Science (EuroSciPy 2013).* Ed. by Pierre de Buyl and Nelle Varoquaux. http://arxiv.org/abs/1404.6391. 2014, pp. 65–70.

[208] Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. “Designing object-oriented software”. In: (1990).

[209] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* Paramount, CA: CreateSpace, 2009. ISBN: 1441412697, 9781441412690.

[210] Ben Putano. “Most popular and influential programming languages of 2018”. In: *URL: https://stackify.com/popular-programming-languages-2018/(accessed Oct 12 2018)* (2018).

[211] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. “The NumPy array: a structure for efficient numerical computation”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.

[212] Eric Jones, Travis Oliphant, and Pearu Peterson. “{SciPy}: open source scientific tools for {Python}”. In: (2014).

[213] Prabhu Ramachandran and Gaël Varoquaux. “Mayavi: 3D visualization of scientific data”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 40–51.

[214] Will J Schroeder, Bill Lorensen, and Ken Martin. *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.

[215] Mike Cantelon et al. *Node.js in Action*. Manning Greenwich, 2014.

[216] Jeff Bezanson et al. “Julia: A fresh approach to numerical computing”. In: *SIAM review* 59.1 (2017), pp. 65–98. URL: `https://doi.org/10.1137/141000671`.

[217] Michael Metcalf, John Reid, and Malcolm Cohen. *Modern Fortran Explained*. 4th. New York, NY, USA: Oxford University Press, Inc., 2011. ISBN: 0199601410, 9780199601417.

[218] Stefan Behnel et al. “Cython: The best of both worlds”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 31–39.

[219] *NumExpr 2.0 User Guide*. 2017. URL: `https://numexpr.readthedocs.io/en/latest/user_guide.html` (visited on 05/10/2019).

[220] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. “Numba: A LLVM-based Python JIT Compiler”. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM ’15. Austin, Texas: ACM, 2015, 7:1–7:6. ISBN: 978-1-4503-4005-2. DOI: `10.1145/2833157.2833162`. URL: `http://doi.acm.org/10.1145/2833157.2833162`.

[221] *Here’s why you should use Python for scientific research*. IBM. Aug. 4, 2018. URL: `https://developer.ibm.com/dwblog/2018/use-python-for-scientific-research/` (visited on 05/30/2019).

[222] OpenMP Architecture Review Board. *OpenMP 5.0 Complete Specifications*. Nov. 2018. URL: `https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf` (visited on 05/09/2019).

[223] Brendan Eich. *Threads suck*. Feb. 12, 2007. URL: `https://brendaneich.com/2007/02/threads-suck/` (visited on 05/07/2019).

[224] John Ousterhout. “Why threads are a bad idea (for most purposes)”. In: *Presentation given at the 1996 Usenix Annual Technical Conference*. Vol. 5. San Diego, CA, USA. 1996.

[225] Mohammad J Rashti and Ahmad Afsahi. “10-Gigabit iWARP Ethernet: comparative performance analysis with InfiniBand and Myrinet-10G”. In: *2007 IEEE International Parallel and Distributed Processing Symposium*. IEEE. 2007, pp. 1–8.

[226] Timothy Prickett Morgan. *Battle Of The InfiniBands*. Nov. 29, 2017. URL: `https://www.nextplatform.com/2017/11/29/the-battle-of-the-infinibands/` (visited on 05/08/2019).

[227] The MPI Forum. *MPI: A Message Passing Interface*. 1993.

[228] Zhang Xianyi, Wang Qian, and Zaheer Chothia. *OpenBLAS*. 2014. URL: `http://xianyi.github.io/OpenBLAS` (visited on 05/07/2019).

[229] R. Clint Whaley and Jack J. Dongarra. “Automatically Tuned Linear Algebra Software”. In: *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*. SC ’98. San Jose, CA: IEEE Computer Society, 1998, pp. 1–27. ISBN: 0-89791-984-X. URL: `http://dl.acm.org/citation.cfm?id=509058.509096`.

[230] *AMD Core Math Library (ACML)*. Apr. 24, 2019. URL: `https://developer.amd.com/wordpress/media/2012/10/acml_userguide.pdf.pdf` (visited on 05/07/2019).

[231] Andy B Yoo, Morris A Jette, and Mark Grondona. “Slurm: Simple linux utility for resource management”. In: *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer. 2003, pp. 44–60.

[232] Matthew Rocklin. “Dask: Parallel computation with blocked algorithms and task scheduling”. In: *Proceedings of the 14th Python in Science Conference*. 130-136. Citeseer. 2015.

[233] M Novák. “Problém vlastních čísel symetrických řídkých matic v souvislosti s výpočty elektronových stavů”. MA thesis. Charles University, Prague, 2012.

[234] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU press, 2012.

[235] Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. Vol. 66. Siam, 2011.

[236] Mehmet A Akgün, John H Garcelon, and Raphael T Haftka. “Fast exact linear and non-linear structural reanalysis and the Sherman–Morrison–Woodbury formulas”. In: *International Journal for Numerical Methods in Engineering* 50.7 (2001), pp. 1587–1606.

[237] Matthias Bollhöfer and Yvan Notay. “JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices”. In: *Computer Physics Communications* 177.12 (2007), pp. 951–964.

[238] Gerard LG Sleijpen and Henk A Van der Vorst. “A Jacobi–Davidson iteration method for linear eigenvalue problems”. In: *SIAM review* 42.2 (2000), pp. 267–293.

[239] G. L. G. Sleijpen and H. A. Van der Vorst. *A Jacobi-Davidson iteration method for linear eigenvalue problems*. 1996. URL: `citeseer.ifi.unizh.ch/article/sleijpen96jacobidavidson.html`.

[240] Gérard Meurant and Zdeněk Strakoš. “The Lanczos and conjugate gradient algorithms in finite precision arithmetic”. In: *Acta Numerica* 15 (2006), pp. 471–542.

[241] Ren-Cang Li and Lei-Hong Zhang. “Convergence of the block Lanczos method for eigenvalue clusters”. In: *Numerische Mathematik* 131.1 (2015), pp. 83–113.

[242] A Booten et al. “Jacobi-Davidson methods for generalized MHD-eigenvalue problems”. In: *Zeitschrift fur Angewandte Mathematik und Mechanik* 76 (1996), pp. 131–134.

[243] Michiel E Hochstenbach and Yvan Notay. “The Jacobi–Davidson method”. In: *GAMM-Mitteilungen* 29.2 (2006), pp. 368–382.

[244] Zhaojun Bai et al. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. SIAM, 2000.

[245] Matthias Bollhöfer et al. “ILUPACK”. In: *Encyclopedia of Parallel Computing* (2011), pp. 917–926.

[246] Satish Balay et al. *PETSc Web page*. `https://www.mcs.anl.gov/petsc`. 2019. URL: `https://www.mcs.anl.gov/petsc`.

[247] Tony F Chan and Henk A Van Der Vorst. “Approximate and incomplete factorizations”. In: *Parallel numerical algorithms*. Springer, 1997, pp. 167–202.

[248] Iain S Duff and John K Reid. *MA47, a Fortran code for direct solution of indefinite sparse symmetric linear systems*. Tech. rep. SCAN-9503196, 1995.

[249] Eric Polizzi and James Kestyn. “FEAST Eigenvalue Solver v3. 0 User Guide”. In: *arXiv preprint arXiv:1203.4031* (2012).

[250] James Kestyn et al. “PFEAST: a high performance sparse eigenvalue solver using distributed-memory linear solvers”. In: *SC’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2016, pp. 178–189.

[251] Vicente Hernandez, Jose E Roman, and Vicente Vidal. “SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems”. In: *ACM Transactions on Mathematical Software (TOMS)* 31.3 (2005), pp. 351–362.

[252] Yunkai Zhou et al. “Self-consistent-field calculations using Chebyshev-filtered subspace iteration”. In: *Journal of Computational Physics* 219.1 (2006), pp. 172–184.

[253] Aydin Buluç et al. “Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks”. In: *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*. 2009, pp. 233–244.

[254] Athena Elafrou et al. “SparseX: A library for high-performance sparse matrix-vector multiplication on multicore platforms”. In: *ACM Transactions on Mathematical Software (TOMS)* 44.3 (2018), p. 26.

[255] *AMD Optimizing CPU Libraries User Guide, version 2.2.* 2020. URL: `https://developer.amd.com/wp-content/resources/AOCL_User%20Guide_2.2.pdf` (visited on 08/01/2019).

[256] *Intel® Math Kernel Library, developer reference.* 2020. URL: `https://software.intel.com/content/dam/develop/public/us/en/documents/mkl-2020-developer-reference-c-0.pdf` (visited on 08/01/2019).

[257] A. C. Ferrari et al. “Science and technology roadmap for graphene, related two-dimensional crystals, and hybrid systems”. In: *Nanoscale* 7.11 (Mar. 2015), pp. 4598–810. ISSN: 2040-3372 (Electronic) 2040-3364 (Linking). DOI: `10.1039/c4nr01600a`.

[258] Zhaohe Dai, Luqi Liu, and Zhong Zhang. “Strain Engineering of 2D Materials: Issues and Opportunities at the Interface”. In: *Advanced Materials* (2019), p. 1805417. DOI: `10.1002/adma.201805417`.

[259] Yufei Sun and Kai Liu. “Strain engineering in functional 2-dimensional materials”. In: *Journal of Applied Physics* 125.8 (2019), p. 082402. DOI: `10.1063/1.5053795`.

[260] Shikai Deng and Vikas Berry. “Wrinkled, rippled and crumpled graphene: an overview of formation mechanism, electronic properties, and applications”. In: *Materials Today* 19.4 (2016), pp. 197–212. ISSN: 1369-7021. DOI: `10.1016/j.mattod.2015.10.002`.

[261] Ji Feng et al. “Strain-engineered artificial atom as a broad-spectrum solar energy funnel”. In: *Nature Photonics* 6.12 (2012), pp. 866–872. ISSN: 1749-4885. DOI: `10.1038/nphoton.2012.285`.

[262] Artur Branny et al. “Deterministic strain-induced arrays of quantum emitters in a two-dimensional semiconductor”. In: *Nature Communications* 8 (2017), p. 15053. DOI: `10.1038/ncomms15053`.

[263] T. M. G. Mohiuddin et al. “Uniaxial strain in graphene by Raman spectroscopy: G peak splitting, Grueneisen parameters, and sample orientation”. In: *Physical Review B* 79.20 (2009), pp. 205433–8. DOI: `10.1103/PhysRevB.79.205433`.

[264] Otakar Frank et al. “Compression Behavior of Single-Layer Graphenes”. In: *ACS Nano* 4.6 (May 2010), pp. 3131–3138. ISSN: 1936-0851, 1936-086X. DOI: `10.1021/nn100454w`. URL: `https://doi.org/10.1021/nn100454w`.

[265] O. Frank et al. “Raman 2D-Band Splitting in Graphene: Theory and Experiment”. In: *Acs Nano* 5.3 (Mar. 2011), pp. 2231–2239. ISSN: 1936-0851. DOI: `10.1021/nn103493g`.

[266] Elena del Corro et al. “Fine tuning of optical transition energy of twisted bilayer graphene via interlayer distance modulation”. In: *Phys. Rev. B* 95.8 (Feb. 2017), p. 085138. ISSN: 2469-9950, 2469-9969. DOI: `10.1103/physrevb.95.085138`. URL: `https://doi.org/10.1103/physrevb.95.085138`.

[267] Niclas S Mueller et al. “Evaluating arbitrary strain configurations and doping in graphene with Raman spectroscopy”. In: *2D Mater.* 5.1 (Nov. 2017), p. 015016. ISSN: 2053-1583. DOI: 10.1088/2053-1583/aa90b3. URL: https://doi.org/10.1088/2053-1583/aa90b3.

[268] Barbara Pacakova et al. “Mastering the Wrinkling of Self-supported Graphene”. In: *Scientific Reports* 7.1 (Aug. 2017), p. 10003. ISSN: 2045-2322. DOI: 10.1038/s41598-017-10153-z.

[269] Farid F. Abraham et al. “A molecular dynamics investigation of rapid fracture mechanics”. In: *Journal of the Mechanics and Physics of Solids* 45.9 (Sept. 1997), pp. 1595–1619. ISSN: 0022-5096. DOI: 10.1016/S0022-5096(96)00103-2.

[270] S. J. Zhou et al. “Large-Scale Molecular Dynamics Simulations of Three-Dimensional Ductile Failure”. In: *Phys. Rev. Lett.* 78.3 (Jan. 1997), pp. 479–482. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/physrevlett.78.479. URL: https://doi.org/10.1103/physrevlett.78.479.

[271] Andrew Leach. *Molecular Modelling: Principles and Applications.* 2 edition. Pearson, Apr. 2001. ISBN: 978-0-582-38210-7.

[272] M. W. Finnis and J. E. Sinclair. “A simple empirical N-body potential for transition metals”. In: *Philosophical Magazine A* 50.1 (July 1984), pp. 45–55. ISSN: 0141-8610. DOI: 10.1080/01418618408244210.

[273] Jacob Fish. *Multiscale Methods: Bridging the Scales in Science and Engineering.* Oxford University Press, Oct. 2009. ISBN: 978-0-19-171553-2. DOI: 10.1093/acprof:oso/9780199233854.001.0001.

[274] R. Ansari, S. Ajori, and B. Motevalli. “Mechanical properties of defective single-layered graphene sheets via molecular dynamics simulation”. In: *Superlattices Microstruct.* 51.2 (Feb. 2012), pp. 274–289. ISSN: 0749-6036. DOI: 10.1016/j.spmi.2011.11.019. URL: https://doi.org/10.1016/j.spmi.2011.11.019.

[275] Loup Verlet. “Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules”. In: *Phys. Rev.* 159 (1 July 1967), pp. 98–103. DOI: 10.1103/PhysRev.159.98. URL: https://link.aps.org/doi/10.1103/PhysRev.159.98.

[276] Daniel Sheppard, Rye Terrell, and Graeme Henkelman. “Optimization methods for finding minimum energy paths”. In: *The Journal of chemical physics* 128.13 (2008), p. 134106.

[277] Julian Arndt Hirschfeld. *Ab initio investigation of ground-states and ionic motion in particular in zirconia-based solid-oxide electrolytes.* Vol. 187. Forschungszentrum Jülich, 2012.

[278] Erik Bitzek et al. “Structural Relaxation Made Simple”. In: *Phys. Rev. Lett.* 97 (17 Oct. 2006), p. 170201. DOI: 10.1103/PhysRevLett.97.170201. URL: https://link.aps.org/doi/10.1103/PhysRevLett.97.170201.

[279] Julien Guénolé et al. “Assessment and optimization of the fast inertial relaxation engine (fire) for energy minimization in atomistic simulations and its implementation in lammps”. In: *Computational Materials Science* 175 (2020), p. 109584.

[280] Taisuke Ozaki et al. *User’s manual of OpenMX Ver. 3.8.* 2016.

[281] Matyáš Novák, Jiří Vackář, and Robert Cimrman. “Evaluating Hellmann–Feynman forces within non-local pseudopotentials”. In: *Computer Physics Communications* 250 (2020), p. 107034.

[282] Matyáš Novák et al. “Mixing algorithms for fixed-point iterations in self-consistent electronic structure calculations”. In: *Proceedings Of Engineering Mechanics 2018* (2018).

[283] Olivier Hardouin Duparc. “On the Origins of the Finnis–Sinclair potentials”. In: *Philosophical Magazine* 89.34-36 (2009), pp. 3117–3131.

[284] PyTables Developers Team. *PyTables: Hierarchical Datasets in Python.* 2002–. URL: `http://www.pytables.org/`.

[285] Aaron Meurer et al. “SymPy: symbolic computing in Python”. In: *PeerJ Computer Science* 3 (2017), e103.

[286] George Karypis and Vipin Kumar. “METIS–unstructured graph partitioning and sparse matrix ordering system, version 2.0”. In: (1995).

[287] Lisandro Dalcin Nathaniel Collier. *igakit: Toolkit for IsoGeometric Analysis (IGA).* URL: `https://bitbucket.org/dalcinl/igakit/src`.

[288] Matyáš Novák et al. “A mixing scheme for fixed-point iterations in self-consistent electronic structure calculations — a comparative convergence study”. In: *Journal of Physics: Condensed Matter* (under review) (2020).

[289] Jiří Vackář et al. “Finite Element Method in Density Functional Theory Electronic Structure Calculations”. In: *Advances in the Theory of Quantum Systems in Chemistry and Physics.* Springer, 2012, pp. 199–217.

[290] Cimrman Robert, Vackář Jiří, and Matyáš Novák. “A Software for Generating and Optimizing Pseudopotentials”. In: *Proceedings of Engineering Mechanics 2019* (2019).

[291] Matyáš Novák et al. “Using a Hybrid Mixing in Fixed-Point Iteration of Density-Functional Theory Loop to Accelerate Electronic Structure Calculations”. In: *Proceedings of European Seminar on Computing 2018* (2018).

[292] Robert Cimrman et al. “Python-based finite element code used as a universal and modular tool for electronic structure calculation”. In: *AIP Conference Proceedings* 1558 (2013), pp. 1532–1535.

[293] Robert Cimrman et al. “Finite element code in Python as a universal and modular tool applied to Kohn-Sham equations”. In: *Proceedings of ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering* (2012).

[294] Robert Cimrman, Jiří Vackář, and Matyáš Novák. “Using SfePy for solving Kohn-Sham equations”. In: *Computational Mechanics 2012*. Nov. 2012, pp. 201–202.

Appendicies

Contents

A List of some libraries used by FENNEC

Sfepy	[207] the finite element framework in Python
Numpy	[211] for efficient handling of large vectors in Python
Scipy	[212] the advanced numeric and other scientifical routines based on Numpy
Jadamilu	[237] the eigenvalue solver
Blzpack	[174] the eigenvalue solver
MA57	[140] the code for LDL matrix decomposition for efficient solving of system of linear equations
PetSc	[246] Toolkit for scientific computation, the solver of linear equations system is used from the package
PyTables	[284] Python library for reading and writing HDF5 format for storing large datasets.
Sympy	[285] for symbolic manipulation of mathematical expressions
METIS	[286] for ordering of sparse matrices (used by solvers)
Cython	[218] the compiled version of Python to speed up crucial parts code, easy integration of C and Fortran routines and for parallelization.
IgaKit	[287] framework for generating NURBS patches
MayaVi	[213] visualisation framework based on VTK
VTK	[214] VTK — Visualisation Toolkit
OpenBLAS	[228] For efficient operations with vectors and matrices
Lebedev quadrature	[205] the library for integration over spheres

B Notation

As the thesis covers different fields of science with different conventions of mathematical notation, it is useful to summarize the notation used through the thesis.

Symbol	**Meaning**
$\mathbb{N},\mathbb{R},\mathbb{C}$	natural, real and complex numbers
$\mathbb{U},\mathbb{V}$	vector spaces $\mathbf{U}=\text{span}\{\mathbf{v}_1,\mathbf{v}_2\dots\mathbf{v}_i\}$
$\mathcal{L},\mathcal{H}$	spaces of functions
$a,b,c,\dots$	scalars $a\in\mathbb{R}$ or $a\in\mathbb{C}$
$\vec{x},\vec{y},\vec{z},\dots$	spatial vectors $\mathbf{v}\in\mathbb{R}^3$
$\mathbf{x},\mathbf{y},\mathbf{z},\dots$	vectors $\mathbf{v}\in\mathbb{R}^n$ or $\mathbf{v}\in\mathbb{C}^n$
f,g,ψ,θ	functions and wavefunctions $\psi\in\mathcal{L}$
$\mathcal{E}$	energy (scalar value)
U,V	operators, matrices (in finite space), or potential functions $(V(\psi))(x)=V(\psi(x))$
r,θ,φ	spherical coordinates $r\in\mathbb{R}_0^+,\theta\in\langle-\pi/2,\pi/2\rangle,\varphi\in\langle 0,2\pi)$
ψ^+,V^+	a complex conjugate of a function ψ or conjugate transpose of a matrix V $V^+_{i,j}=\overline{V_{j,i}}$
$(\mathbf{u}\cdot\mathbf{v})$	a scalar product $(\mathbf{u}\cdot\mathbf{v})\in\mathbb{R}$
$\langle\mathbf{u}\vert\mathbf{v}\rangle$	a scalar product in bra-ket notation
$\langle\mathbf{u}\vert A\vert\mathbf{v}\rangle$	a scalar product with an operator in bra-ket notation $\langle\mathbf{u}\vert A\vert\mathbf{v}\rangle=\int\mathbf{u}^+A\mathbf{v}$
$\mathbb{U}\times\mathbb{V}$	a direct product of vector spaces $\dim(\mathbb{U}\times\mathbb{V})=\dim(\mathbb{U})+\dim(\mathbb{V})$
$\psi\times\phi$	direct product of functions $\psi(\mathbf{x})\times\phi(\mathbf{y})=\psi(\mathbf{x})\phi(\mathbf{y})$ $(\psi:\mathbb{R}^n\to\mathbb{R})\times(\phi:\mathbb{R}^m\to\mathbb{R})=\sigma:\mathbb{R}^{n+m}\to\mathbb{R}$
$\Vert\mathbf{x}\Vert,\Vert\psi\Vert$	L_2 norm $\Vert\mathbf{x}\Vert=(\mathbf{x}\cdot\mathbf{x})^{\frac{1}{2}},\Vert\psi\Vert=(\int_{\mathbf{x}}\psi\psi^+)^{\frac{1}{2}}$
$P_n,P_l^m,$ Y_{lm}	the Legendre and the associated Legendre polynomials and the spherical harmonic functions[3]
$\Omega,\Omega_h,\partial\Omega_h$	the volume, discretized volume, and its boundary
φ,ϑ	elements of basis of space of solution $\varphi\in\Omega$
∇	the gradient operator $\nabla\psi=\sum_i {}^{\delta\psi}/_{\delta\mathbf{x}_i}\mathbf{e}_i$
∇^2	the Laplace operator $\nabla^2\psi=\triangle\psi=\sum_i {}^{\delta^2\psi}/_{\delta\mathbf{x}_i^2}$
$\int_\Omega,\int$	integral over volume, $d\mathbf{x}$, and Ω are omitted for the sake of simplicity, if no confusion can arise

[3] P_l denotes also projector to given l-space.

C List of used shortcuts

a.u.	(Hartree) atomic unit (of length)
CSR	Compressed sparse row
DoF('s)	Degree(s) of freedom
DFT	Density functional theory
e.g.	exempli gratia (for example)
Eq.	Equation
E_h	Hartree unit of energy
FEM	Finite element method
Fig.	Figure
GIL	(Python) global interpret lock
HF(F)	Hellmann-Feynman (forces)
i.e.	id est (in other words)
IGA	Isogeometric analisis
NNzs	Number of nonzeros
NURBS	Non-uniform rational B-spline
PDE	Partial differential equation
psp.	Pseudopotential
SDFT	Spin density functional theory
Tab.	Table
w.r.t.	with regard to
XC	Exchange-correlation

D List of algorithms

E List of tables

F List of figures

G List of theorems

H List of definitions

I Index of used definitions and terms

J List of published papers

Papers in journals with impact factor

Matyáš Novák et al. “A mixing scheme for fixed-point iterations in self-consistent electronic structure calculations — a comparative convergence study”. In: *Journal of Physics: Condensed Matter* (under review) (2020)

Matyáš Novák, Jiří Vackář, and Robert Cimrman. “Evaluating Hellmann–Feynman forces within non-local pseudopotentials”. In: *Computer Physics Communications* 250 (2020), p. 107034

Robert Cimrman et al. “Isogeometric analysis in electronic structure calculations”. In: *Mathematics and Computers in Simulation* 145 (2018), pp. 125–135

Robert Cimrman et al. “Convergence study of isogeometric analysis based on Bézier extraction in electronic structure calculations”. In: *Applied Mathematics and Computation* 319 (2018), pp. 138–152

Jiří Vackář et al. “Finite Element Method in Density Functional Theory Electronic Structure Calculations”. In: *Advances in the Theory of Quantum Systems in Chemistry and Physics.* Springer, 2012, pp. 199–217

Other papers

Cimrman Robert, Vackář Jiří, and Matyáš Novák. “A Software for Generating and Optimizing Pseudopotentials”. In: *Proceedings of Engineering Mechanics 2019* (2019)

Matyáš Novák et al. “Using a Hybrid Mixing in Fixed-Point Iteration of Density-Functional Theory Loop to Accelerate Electronic Structure Calculations”. In: *Proceedings of European Seminar on Computing 2018* (2018)

Matyáš Novák et al. “Mixing algorithms for fixed-point iterations in self-consistent electronic structure calculations”. In: *Proceedings Of Engineering Mechanics 2018* (2018)

Robert Cimrman et al. “Python-based finite element code used as a universal and modular tool for electronic structure calculation”. In: *AIP Conference Proceedings* 1558 (2013), pp. 1532–1535

Robert Cimrman et al. “Finite element code in Python as a universal and modular tool applied to Kohn-Sham equations”. In: *Proceedings of ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering* (2012)

Robert Cimrman, Jiří Vackář, and Matyáš Novák. “Using SfePy for solving Kohn-Sham equations”. In: *Computational Mechanics 2012.* Nov. 2012, pp. 201–202

Summary

A new ab-inito method and the related computer code has been developed within this work — in accordance with its aims — for calculating electronic states and structural properties of (primarily) nonperiodic, possibly electrically charged, material structures. This method has been implemented into the software package FENNEC.

The new method fills a gap among the current well-established methods and represents a counterpart of what the plane-wave method is within the class of the methods employing the Bloch's theorem (for materials with translational symmetry). The newly developed method is based on the density functional theory (DFT), the finite element method, or isogeometric analysis as an option, and environment-reflecting separable pseudopotentials, and it provides an excellent convergence control within a general, in principle arbitrarily precise basis.

During the development of the method, a new precise and computationally efficient formula for evaluating Hellmann-Feynman forces has been derived, implemented, and used for geometry optimizations. Its advantages and very good convergence properties have been demonstrated in sample calculations.

Among other originals results of this work, a purely algebraic description of separable pseudopotentials should be mentioned. This description allows constructing a projection basis to achieve any required accuracy.

Besides that, an adaptive version of the Anderson iteration scheme, which solves the problem of choosing the right mixing parameter (an approximation of the Jacobian) in the DFT loop, has been proposed, implemented, and analyzed.

Discretization of the Kohn-Sham equations using the finite element methods leads to a rank-k-update generalized eigenvalue problem, for solving of which an efficient method had to be developed: to find the appropriate eigenvalue solvers, integrate them into the FENNEC software package, and make them capable to solve the rank-k-update problem.

The convergence properties of that method in dependence on various technical parameters — i.e. the finite element shape, basis type, and the order and number of projectors for separable pseudopotentials — have been examined and analyzed to determine the best options for the new method.

The newly created method has been implemented using the Python finite element framework SfePy (and many other libraries) into the software package FENNEC. This software package is characterized by a modular plugin architecture which significantly simplifies its further development and makes its usage easy. FENNEC has been used, among other tasks, e.g. for geometry optimization of a deformed graphene fragment and calculations of forces in it as a preparation for the ab-initio construction of force-interaction molecular dynamics potentials.

Shrnutí práce

V této práci — v souladu s jejím cílem — byla vyvinuta nová ab-inito metoda a odpovídající software pro výpočet elektronových stavů a strukturních vlastností (primárně) neperiodických, popřípadně elektricky nabitých, materiálových struktur. Tato nová metoda zaplňuje mezeru mezi současnými běžně užívanými metodami a představuje protějšek toho, čím je metoda rovinných vln mezi metodami užívajícími Blochův teorém (sloužícími pro výpočet materiálů s translační symetrií). Nová metoda je založena na teorii funkcionálu hustoty, metodě konečných prvků, popř. volitelně na isogeometrické analýze, a prostředí-reflektujících pseudopotenciálů, a nabízí výtečnou kontrolu konvergence díky obecné, v principu libovolně přesné bazi.

Během vývoje metody byla vyvinuta, implementována a pro geometrické optimalizace použita nová formulace výpočtu Hellmann-Feynmanových sil. Její přednosti, především výtečná konvergence, byly demonstrovány na ukázkových výpočtech.

Mezi další originální výsledky této práce patří čistě algebraické vyjádření separabilní formy pseudopotenciálů. Tento popis umožňuje konstrukci projekční báze tak, že jde dosáhnout libovolné požadované přesnosti.

Mimo to byla navržena, implementována a analyzována adaptivní verze Andersonova iterativního algoritmu, která řeší problém výběru mixovacího parametru (počátečního odhadu jakobiánu) v DFT selfkonsistentním cyklu.

Diskretizace Kohn-Shamových rovnic za pomoci metody konečných prvků vede k rank-k-update problému vlastních čísel, pro jehož řešení bylo třeba vyvinout efektivní metodu: najít vhodné řešiče, integrovat je do softwarového balíku FENNEC a rozšířit možnosti těchto řešičů o řešení rank-k-update problému.

Byly zkoumány a analyzovány konvergenční vlastnosti této metody v závislosti na různých technických parametrech — např. tvaru konečných elementů, typu báze a jejím řádu, počtu projektorů separabilních pseudopotenciálů — a touto cestou byly získány optimální hodnoty přinášející nejlepší výpočetní výkon a přesnost.

Tato metoda byla implementována v jazyce Python s využitím konečněprvkového rámce SfePy do softwarového balíku FENNEC. Tento software se vyznačuje modulární architekturou založenou na pluginech, která podstatně usnadňuje jeho další vývoj i přispívá k jeho velmi snadnému používání. Jako demonstrace schopností tohoto software je v práci prezentovaná optimalizace geometrie fragmentu grafenové vrstvy a výpočet sil potřebných na odtržení atomu z této vrstvy. Tyto výpočty budou sloužit k ab-initio konstrukci interakčních potenciálů pro simulace grafenu pomocí molekulární dynamiky.