A Computational Environment for Multiscale Modelling

by

Morten Ledum

Thesis

for the degree of

Master of Science

Faculty of Mathematics and Natural Sciences
University of Oslo

December 2017

This thesis was typeset with the LaTeX typesetting system.
Effort has been made to adhere to the ISO 80000-2:2009 standard on mathematics typesetting [38].

Abstract

We implement two different *ab initio* electronic structure methods: Hartree-Fock (HF), and quantum variational Monte Carlo (VMC). Gaussian type orbitals are used for the HF method, while the VMC framework allows more general orbital bases (including the possibility of using the optmized HF orbitals). A thorough introduction to the underlying theory of both methods is presented, and the codes are tested on select first row atoms and simple molecules. Ground state energies are found to be in good agreement with the litterature.

Secondly, a general function approximation scheme is implemented using artificial neural networks (ANN). The ANN implementation is based on the TensorFlow library developed by the Google Brain team. It is thoroughly tested on single and multivariable functions and subsequently shown to be able to approximate potential energy surfaces (PES) using data from the aforementioned *ab initio* calculations.

The ANN may then be used as a force field in molecular dynamics (MD) simulations—in place of ordinary parametrized effective MD potentials—thereby successfully bridging the quantum mechanical and the microscopic regimes. Whereas traditional MD potentials require hand crafting and tuning of a parametrized functional form, the present work implements a *multiscale modelling* approach in which essentially no human intervention is needed. Such "parameter-free" multiscale modelling is preferable for obvious reasons: the results should be fundamentally independent of the human experimenter's ability to *guess* an appropriate functional form.

Lastly: Showcasing the full usage of the computational framework developed, we present a simple—proof of concept—MD simulation using an ANN trained to approximate a PES.

Acknowledgements

I would like to thank my supervisors (in no particular order) Anders Malthe-Sørensen, Morten Hjorth-Jensen, and Anders Hafreager for their support during the years as a master student here at the physics department. Along with the rest of the community at the computational physics group, you have created a wonderful environment for learning that I will sorely miss. To Morten: thank you for taking me under your wing here at the computational physics group. It has been a pleasure to be your student and later your TA. I am proud to have taken some small part in teaching the introductory computational physics course with you these past several years.

To my tireless *room-mate* here in office V306, Håkon Kristiansen. It feels like we have been sitting here for decades, and I very much appreciate the company. I promise to stop asking you difficult questions you can't answer (at some point). Maybe one day the dream will come true and someone will pay us to just sit around and *figure out* whatever is interesting to us that day.

I want to acknowledge the two people who have probably influenced me as a student the most. Anders Hafreager and Håvard Tveit Ihle, you both are an inspiration to me, showing what you can achieve through diligent hard work. Anders, thank you for teaching me how to write (less awful) code, and for helping me not fail QFT. I truly appreaciate the endless enthusiasm and the willingness to go out of your way to help regardless of the nature of the problem. Håvard, I am looking forward to floundering my way through your cosmology lectures this spring!

I also wish to thank the unlikely Johan, the (true[ly]) interesting Icelandic bat-camel for all the stimulating distractions throughout the time working on this thesis.

Lastly, to Vilde: Thank you for the endless encouragement (and all the sushi!). Thank you for putting up with the largely absent and massively stressed out version of myself for long months.

Collaboration Details

All code developed for this thesis have been written from scratch by the author. The ANN code has been developed in partial collaboration with Stende [1] and Treider [2], the HF program was written with a lot of inspiration taken from Dragly [3], and some ideas for the VMC code have been borrowed from an earlier program written by the author and Håvard Tveit Ihle.

Morten Ledum
Oslo, December 2017

Contents

Chapter 1

Introduction

After being deveolped and pioneered around the middle of the last century, computer modelling and numerical experiments have become ubiquitous in the natural sciences. The list of areas in which simulations have been used to produce significant new results encompass now pretty much all of them. Today computer simulations play as natural a part of the hard sciences as laboratory experiments and theory. In most cases all three are used in order to glean new scientific insight. Without massive scale computational efforts, the existence of e.g. the Higgs boson and gravitational waves would still be undetermined.

The advent of computer simulations during the last several decades have in particular made it possible to study moderately sized quantum mechanical systems from first principles. Our ability to solve—in closed form—the governing equations of quantum mechanics (QM) vanishes extremely quickly as the number of constituent particles exceed just a few. Because of this, numerics are used to augment the proverbial *pen and paper*. It is nevertheless striking that the underlying theory for all of chemistry and most of physics have been known for almost a century but the problem preventing us from essentially *solving* chemistry is almost purely computational: the equations resulting from the exact application of this theoretical framework are way too difficult to solve.

Any approximative scheme which aims to solve the many-body Schrödinger equation from scratch subject to some (more or less) well-defined simplifications is called an *ab initio* method. Working from first principles the aim of such algorithms is to extract information from a theoretical QM system in a reasonable amount of time. In order to accomplish this, a number of complicating intricacies need to be disregarded. The magnitude of the simplifications—essentially the number and importance of complicating factors dropped—determine both the efficacy and the efficiency of the method: More simplifications made allow solutions to be found for larger systems (albeit less precise solutions), whereas extremely precise solutions can be found for small systems if very few simplifications are employed.

Despite tremendous increases in available numerical computational power in the latter half of the previous-, and the early parts of the current century, any such ap-

proximate scheme used is still heavily limited w.r.t. the system size. In practice, most methods are limited to systems of containing on the order of between 10^2 (for high-precision methods such as CI, CC, DMC, etc.) and 10^5 electrons (for faster HF and DFT methods) [4–6]. Extracting information from larger systems neccessitate the use of semi-classical or classical algorithms, such as molecular dynamics (MD). Using MD, the time evolution of up to around 10^7 particles can feasibly be simulated over the order of nano seconds [7, 8]. Using computationally inexpensive two-body inter-atomic potentials and simulating for only a very short time, around 10^{12} particles can be modelled [9]. Corresponding large-scale cosmological N-body simulations have been run for as many as 10^{11} particles [10, 11]. For ensembles of particles larger than around 10^{12}, simulating the individual constituents directly becomes too computationally expensive and we have to use continuum models.

At the boundary between each domain, we are essentially forced into fundamentally differing simulations. As our models move from fine to more coarse-grained with increasing system size, the internal degrees of freedom of the constituent parts are *frozen out*. MD freezes the electronic motion, treating the atomic structure as rigid and solid. Continuum models do away with the atomic motion all together—e.g. treating only a density field defined on some lattice—thereby freezing all direct inter-atomic interactions.

Being able to preserve—in some way—the properties of the fine-grained model in the larger domain is of tremendous scientific value. Lessons learnt from the *ab initio* QM simulations should ideally provide the fundamental basis for developing MD schemes. In the same way, the field equations of the continuum model should incorporate as much of the physics of the molecular simulations as possible. This is the heart of **multiscale modelling**. By simultanously considering a system at different scales (modelling domains) we may hope to arrive at a scheme which preserves *most* of the crucial QM properties, while simultaneously retaining *most* of the efficiency of the coarse-grained models [12].

1.1 Quantum and classical dynamics

As previously noted, solving the Schrödinger equation (SE) exactly by hand is impossible in the overwhelming majority of interesting cases. However, methods which can get close to the exact solution exists. Full Configuration Interaction (FCI) or direct diagonalization of the Hamiltonian is exact in the limit of an infinite orbital basis set but suffers from an exponential complexity scaling (in system *and* basis size) [13]. The related Configuration Interaction (CI) and Coupled Cluster (CC) approaches both truncate the FCI expansion of Slater determinants, thus gaining speed but loosing some accuracy [14, 15]. Diffusion Monte Carlo (DMC) techniques can in principle provide the exact solution to the SE by imaginary-time evolution of an initial wave function guess [16, 17]. In practice, DMC methods are highly dependent on this ansatz and thus require as input the results of less accurate method but faster methods. One

example may be the Variational Monte Carlo (VMC) method: conceptually simpler and faster than DMC, but not as accurate [17–19].

The Hartree-Fock (HF) framework—which provides an efficient but not enormously accurate result—has seen extensive use since its inception in 1930 [20–22]. However, by far the most popular approximation is Density Functional Theory (DFT), developed by W. Kohn and L. J. Sham in 1965 [23, 24]. Between 1980 and 2010, DFT was the most active field in physics with eight out of the top ten most cited papers being on the subject [25].

Computational scaling of *ab initio* QM models range from $\mathcal{O}(N!)$ in the extreme (FCI), via $\mathcal{O}(N^6)$ (CC with singles, doubles, and estimated connected triples) and $\mathcal{O}(N^4)$ (formal HF), to $\mathcal{O}(N^3)$ for Hartree-Fock with integral pre-screening and density fitting [26].

In the intermediate region between these methods and molecular dynamics, we find a range of hybrid models. Examples include Car-Parrinello MD—in which the valence electrons are included in the dynamics and treated in a semi-classical way—and Born-Oppenheimer MD, where the time independent Schrödinger equation is solved approximately at each time step [27–29]. The computational complexity of such hybrids fall somewhere in the middle of the true *ab initio* methods and plain MD. The latter has a naive scaling of $\mathcal{O}(N^2)$, but can be made linear by ingenious partitioning schemes.

1.2 Machine learning and artificial neural networks

The term artificial intelligence (AI) describes machines which humans percieve as *smart*. Narrow AI, machines focused sorely on a singular task and often achieving super-human performance, are found everywhere in the 21st century. It is remarkable that for example your phone can soundly beat you at the game of chess without breaking a sweat[1]. A computer's ability to beat the best chess players in the world was once a huge breakthrough in the field of AI, but it is easy to just think of this as a computer applying an algorithm to find the correct moves. As the father of the term artificial intelligence—American computer scientist John McCarthy—put it [30]

> “As soon as it works, no one calls it AI any more.”
>
> *J. McCarthy*

While a computer beating a human at chess is impressive, it traditionally does so by sheer brute force calculational power. A human programmed search algorithm traverses vast trees of possible moves, evaluating each position using a human programmed evaluation function to try and find the optimal one. A different approach is

[1] If you happen to be a world-class chess player, substitute phone⇋laptop

taken in *machine learning* (ML). As a subdiscipline of AI, the field of machine learning is focused on creating computers which automatically learn and improve their behaviour through experience [31]. Very recently, researchers at Google Deep Mind were able to create a program which learns by playing itself [32]. With the only input being the rules of the game and the conditions for victory, the AlphaZero algorithm was able to beat one of the world's top traditional (*brute force*) chess engines—Stockfish—with less than 24 hours of reinforcement training [33].

The most popular form of machine learning today is performed by a class of algorithms called artificial neural networks (ANN). Inspired by networks of biological neurons forming brains, the ANNs model a learning process by adjusting the weights connecting individual artificial neurons in the network structure.

The chess example showcases beautifully the major problem inherent in the brute force algorithmic approach: the efficacy of the algorithm is fundamentally limited by the human programmers ability to evaluate a given board position. The machine learning approach—however—is not so limited, and essentially figures out the objectively best way to play by trial and error. We will see shortly that the analogue to the way MD potentials are ordinarily created is clear: finding the parameters of a predefined functional form limits the possible forms we can describe wheras a machine learning approach based on ANNs is in principle not so limited.

1.3 Machine learning in molecular dynamics

Traditionally, MD simulations are performed with classical effective potentials which are parametrized to hopefully capture some of the underlying quantum physics. Such empirical model potentials are computationally cheap, and may perform adequately in isolated cases. A lot of hand-tuning of empirical, chemically motivated, parameters is however required in order to reproduce mesoscopic quantities within the MD framework (pair correlation functions, crystalline structure, etc.). As such, constructing effective potentials is a highly non-trivial and time-consuming task—that ultimately applies only to a small subset of atomic systems at best. Trying to employ a given potential for a system it was not designed for may very well yield qualitatively wrong results.

In the spirit of multiscale modelling, developing "parameter free" MD schemes is very desirable. Foregoing the complicating, human labour intensive, and intrinsicly difficult step of emipirical potential fitting, the method of Behler and Parrinello use instead machine learning [34]. This and related models recently developed exploit the unbiased predictive power of *ab initio* QM simulations to parameterize the effective potential automatically by use of artificial neural networks [35]. In this way human intervention is shunned and the accuracy of the MD simulation is no longer fundamentally limited by the imagination and physical intuition of the experimenter. Despite being typically computationally more expensive than the traditionally parameterized effective potentials, ANNs still offer *orders of magnitude* better performance

than *ab initio* or even hybrid models [26, 36].

1.4 Goals

The main goal of this thesis is to implement a fully functional (albeit simple) multiscale modelling framework for connecting quantum mechanics to microscopic molecular dynamics simulations. Quantum *ab initio* training data will be generated at different levels of theory and used as training data for a feed-forward ANN. The fully trained ANN-potential-energy surface (PES) is then used in molcular dynamics simulations.

This goal is naturally split into a few intermediate objectives:

(a) Develop *ab initio* quantum simulation software

(1) Develop a Hartree-Fock code
The first level of *ab initio* theory—and the starting point for the others—is the Hartree-Fock scheme. We wish to write a completely general HF implementation for atoms and molecular systems using Gaussian type orbitals as bases. This will be done completely from scratch in C++.

(2) Develop a Variational Monte Carlo code
Having solved the Hartree-Fock equations, the next level of theory we want to employ is the variational quantum Monte Carlo approach. The VMC framework should be general enough to both work with the results from HF simulations, and as a fully stand-alone code. This will also be done completely from scratch in C++.

(3) Develop a *local density approximation*—density functional theory code
Having developed a functioning HF program, we wish to extend it to also be able to handle density functional calculations at the LDA level. This primarily entails developing a framework for efficient numerical integration of the electronic density and derived quantities. This will be done from scratch in C++, with the dftlibs/numgrid library specifically handling the integration grid setup.

(b) Develop a ANN model for PES fitting
Having developed various *ab initio* frameworks for computing molecular energy, we wish to develop a method of fitting said data to PES for use in molecular dynamics simulations. This will be done using the TensorFlow framework, which we will interface from Python scripts.

(c) Use the ANN-PES in simple MD simulations
As a proof of concept and validation of the entire multiscale modelling scheme,

we wish to implement the code neccessary to run molecular dynamics simulations using the ANN potential surfaces. This will be done using the MD package LAMMPS.

1.5 Our contributions

Dozens of highly optimized, well-tested, and professional *ab initio* QM code bases already exist. Developing code which is able to compete with such packages is unfortunately well outside the scope of the present work. The goal of developing *ab initio* QM methods from scratch is to glean useful insight about the various methods, their inner workings, their strenghts, and potential pit-falls associated with their use.

For the ANN and the training process, we employ the TensorFlow (TF) library. We manually set up and feed the NN data for the training, but the optimization, backpropagation, and automatic differentiation is all handled by TF.

For the molecular dynamics modelling, the LAMMPS library is used. In order to evaluate the ANN and it's derivatives, the custom LAMMPS extension of Stende and Treider is used [1, 2].

All in all about 16 000 significant lines of code[2] was developed for the present work of which about 85% is written in C++, about 10% in Python, with the remaining few percentages written in a mix of MATLAB, Mathematica, and Julia.

Bridging quantum mechanics and molecular dynamics with machine learning techniques is not a new concept: researchers have been using ANNs for potential-energy surface fitting for at least two decades. We do not offer any fundamental contributions to this field. We do however note that—to the best of our knowledge—the coupling of quantum Monte Carlo, ANNs, and MD simulations is a relatively novel approach.

1.6 Developed source code

All the code developed in conjunction with the present work is freely available on the author's github site, github.com/mortele, under a *"do whatever you want with it"* public licence. (Re)Using any parts of it is highly encouraged.

1.7 Structure of the thesis

This thesis is split into four parts. The first part, *foundational theory* presents an overview of classical mechanics (very briefly) and quantum mechanics as relevant for molecular dynamics and electronic structure calculations. Part one lays the ground-

[2] As counted by the cloc program which counts *significant* lines of code, leaving out blank lines, comment lines, etc. [37]

work and establishes most of the notation used later in part two: *Advanced theory*. Here, the different approximative schemes for solving the quantum equations of motion are presented. Alongside the theory of artificial neural networks.

The penultimate part contains information w.r.t. the concrete implementation of frameworks described in part two. Key parts of each code base are outlined in detail. Also contained in part three are validation tests of each implementation. Lastly, results for the full work-flow utilizing all the deveolped programs are presented.

The thesis ends with part four, containing conclusions and prospect for future work.

Part I

Foundational theory

Chapter 2

Quantum Mechanics

2.1 A (very brief) review of classical mechanics

Before venturing into the land of quantum mechanics (QM), it is useful to first review the Hamiltonian formulation of classical mechanics (CM). Classical mechanics deals with the dynamics of macroscopic objects.

Hamilton's classical mechanics formalism revolves centrally around the Hamiltonian function (hereafter just refered to as *the Hamiltonian*), $\mathcal{H}$. In order to define this function, it is necessary to first choose a set of canonical coordinates, $\mathbf{q}_i$ and $\mathbf{p}_i$. Generalized coordinate and momenta pairs are said to be canonical if they satisfy the Poisson bracket[1] $\{q_i, p_j\} = \delta_{ij}$. Examples of such canonical pairs include e.g. the cartesian position and the respective linear momentum, or the polar position and the angular momentum.

The *phase space* of a system is the space of all possible states of the system. A system of n degrees of freedom will have a cooresponding $2n$ dimensional phase space.[2] A point in phase space, $\xi = (\mathbf{q}, \mathbf{p})$, specify the generalized coordinates and their respective conjugate momenta and is sufficient to uniqely determine the state of the system as a whole.

Together with a choice of cannonical coordinates and the resulting phase space, the Hamiltonian encodes all information about a classical dynamical system. From

[1] The Poisson bracket of two functions, f and g, w.r.t. the canonical coordinate pair p and q, is defined as [39]

$$\{f, g\} = \frac{\partial f}{\partial q}\frac{\partial g}{\partial p} - \frac{\partial f}{\partial p}\frac{\partial g}{\partial q}.$$

[2] Although we are free to choose a set of generalized coordinates larger than the number of degrees of freedom, such a set will always be reducible to a smaller set of *strictly independent* coordinates with size exactly equal to the number of degrees of freedom. Consider e.g. a point particle constrained to move along the suface of a unit sphere. We may use the three cartesian coordinates to describe the motion, however the system only has two degrees of freedom due to the constraint $x^2 + y^2 + z^2 = r^2$, so the three are not *independent*. It will thus inevitably be more convenient to use e.g. the polar and azimuthal angles, θ and ϕ.

the Hamiltonian, we can find the equations of motion in terms of the chosen coordinates by applying the Hamilton equations,

$$\frac{\mathrm{d}\mathbf{p}_i}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}_i} \quad \text{and} \quad \frac{\mathrm{d}\mathbf{q}_i}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}. \tag{2.1}$$

We may also state Hamilton's equations more concisely in terms of the Poisson brackets as

$$\frac{\mathrm{d}\mathbf{p}_i}{\mathrm{d}t} = \{\mathbf{p}_i, \mathcal{H}\} \quad \text{and} \quad \frac{\mathrm{d}\mathbf{q}_i}{\mathrm{d}t} = \{\mathbf{q}_i, \mathcal{H}\}\,. \tag{2.2}$$

It is useful to note that we may interpret the Hamiltonian as *the total energy of the system*, if and only if the generalized coordinates have no explicit time dependence and the forces acting on it are derivable from a conservative potential (i.e. the work done by the force(s) are independent of the path taken) [39].

An important (to us at least) special case of classical Hamiltonian dymanics is the system consisting of N identical particles of equal mass m, subject to inter-particle forces stemming from a *central* potential $w(q_{ij})$ with $q_{ij} = |\mathbf{q}_i - \mathbf{q}_j|$, and moving in an external potetial $v(\mathbf{r})$. We may choose generalized coordinates with no explicit time dependence, and this allows us to identify the Hamiltonian as the total energy of the system. Following [14], we find that $\mathcal{H} = T + V + W$, with T, V, and W, denoting the kinetic, the potential, and the interaction energy respectively. We may write this out as

$$\mathcal{H}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{N} \frac{|\mathbf{p}_i|^2}{2m} + \sum_{i=1}^{N} V(\mathbf{q}_i) + \sum_{i=1}^{N} \sum_{j=i+1}^{N} w(q_{ij}). \tag{2.3}$$

2.2 Canonical (first) quantization

In order to go from a classical description to a quantum mechanical (QM) one, the procedure known as canonical quantization is employed. This is sometimes refered to as *first quantization*, to distinguish it from *second quantization*. Second quantization is a QM framework used to study many-particle systems (which we will largely ignore in the present work).

2.2.1 Short mathematical interlude

Under canonical quantization, the state of a system is no longer described by a *point in phase space*, but rather by *a state vector*. The enclosing vector space is a Hilbert space over $\mathbb{C}$, and almost always chosen as the space of square integrable functions[3],

[3]For any $f \in \mathcal{L}^2$, the integral $\int_{-\infty}^{\infty} |f(x)|^2 \,\mathrm{d}x$ must be finite.

$\mathcal{L}^2$. More precisely, the vector space in question is the Sobolev space H^1 over $\mathbb{C}$, essentially the subspace of $\mathcal{L}^2$ for which the first derivatives are also square integrable. For a mathematically rigorous definition of Sobolev spaces, see e.g. [40].

Abstract vectors in our Hilbert spaces are denoted using Dirac's bra-ket notation. The state vector $|\psi\rangle$ is a member of $\mathcal{L}^2$, while the corresponding Hermitian conjugate, $|\psi\rangle^\dagger = \langle\psi|$ is a member of the $\mathcal{L}^2$ dual space. In general, the dual of the Hilbert space $\mathcal{H}$ is a Banach space and the bra corresponding to ket $|\psi\rangle$ is a linear functional, $\langle\cdot| : \mathcal{H} \to \mathbb{C}$. Luckily, $\mathcal{L}^2$ is it's own dual space, and this is usually stated as the combination $\langle\psi|\phi\rangle$ meaning the inner product between the two abstract vectors $|\psi\rangle$ and $|\phi\rangle$. Since the $\mathcal{L}^2$ inner product is the familiar integral over space, we have $\langle\psi|\phi\rangle = \int_{\mathbb{R}} \psi(x)^\dagger \phi(x)\,\mathrm{d}x$.

In a finite dimensional space, we may employ an orthonormal basis and express the general vectors in terms of this basis. We can always represent the basis vectors as ordinary $\mathbb{R}^n$ column vectors[4], $|e_1\rangle = (1, 0, \ldots, 0)^T$, $|e_2\rangle = (0, 1, 0, \ldots, 0)^T$, ... In this case, bra-vectors are simply row vectors with the bra-ket composition understood to be matrix multiplication. It is important to note that *any* vector in such a space can be represented in terms of this basis, $|\psi\rangle = \sum_{i=1}^n c_i |e_1\rangle$ and employing this we may compute any inner product $\langle\psi|\phi\rangle$ as a series of matrix multiplications of the unit colum/row vectors.

When we are not fortunate enought to be able to work in a finite dimensional subspace of $\mathcal{L}^2$, we will assume that the infinite space is *separable*. This means there exists a countably infinite set $D = \{|e_i\rangle\}_{i=1}^\infty$ of orthonormal functions which form a basis for $\mathcal{H}$ [41]. In more mathematical terms, we say that D is a countable *dense* subset of $\mathcal{H}$, the closure of which span $\mathcal{H}$. The existence of such a set is (perhaps anti-intuitively) not at all obvious. Although we are guaranteed that *any* Hilbert space (not neccessarily finite dimensional) contains at least one orthonormal sequence, so we can write for any $|\psi\rangle \in \mathcal{H}$: $|\psi\rangle = \sum_{i=1}^\infty \langle\psi|e_i\rangle|e_i\rangle$. However, we are in no way guaranteed that this converges in $\mathcal{H}$ and if it does, we are in no way guaranteed that it converges to $|\psi\rangle \in \mathcal{H}$ [42]. As it turns out, the assumption that $\mathcal{H}$ be separable is exactly the neccessary and sufficient condition for this sum to behave like we are used to in the finite dimensional case. Perhaps the most important consequence of separability is that we can *realize unity* in terms of this basis, that is the following equation holds [43]

$$\sum_i |e_i\rangle\langle e_i| = \mathbb{1}.$$

This is sometimes called the *Parseval relation* [42].

The orthonormal basis is sometimes called a *complete orthonormal sequence* [42] which causes confusion: when physicists talk about completeness they are talking about the existence of such a basis set and the resulting validity of $\sum_i^\infty |e_i\rangle\langle e_i| = \mathbb{1}$

[4]Since the mapping $f : \mathcal{H} \to \mathbb{R}^n$ defined by $f(|\phi\rangle) = f(c_1|v_1\rangle + c_2|v_2\rangle + \cdots + c_n|v_n\rangle) = (c_1, c_2, \ldots, c_n)^T$ (with the set $\{|v_i\rangle\}_{i=1}^n$ being a basis for $\mathcal{H}$) is a linear, injective map *onto* $\mathbf{R}^n$ and thus define an isomorphism between $\mathcal{H}$ and $\mathbb{R}^n$.

(sometimes called the completeness relation). However, when mathematicians talk about completeness, they are almost always refering to the fact that any Cauchy sequence in $\mathcal{H}$ converges *in* $\mathcal{H}$.

We note that for a infinite dimensional, separable Hilbert space, there exists an isomorphism between $\mathcal{H}$ and ℓ^2, the Hilbert space of *square summable sequences*.

In the following, we will take $|\psi\rangle$ to denote an abstract state vector. Expanding any such vector in terms of the the basis of position-eigenstates (basically just an enumeration of all possible positions available to the system) yields what we will call the *wave function*: $\psi(x) = \sum_i |x_i\rangle\langle x_i|\psi\rangle = \sum_i c_i|x_i\rangle$, with $c_i \equiv \langle x_i|\psi\rangle$.

From dynamical variables to operators

The classical observables, the generalized coordinates and conjugate momenta, are promoted to *operators* acting on state vectors in the aforementioned Hilbert space. Working in the position basis, the position *operator* becomes a simple multiplication operator: $\hat{x}\psi = x\psi$. The momentum operator becomes a differential operator, $\hat{p}\psi = -i\hbar(\partial\psi/\partial x)$. In addition, the old Possion brackets for classical mechanics are promoted to *commutator relations*,

$$\{f, g\} \quad \rightarrow \quad \frac{1}{i\hbar}\left[\hat{f}, \hat{g}\right]. \tag{2.4}$$

This means the fundamental Poisson bracket, $\{q_i, p_j\} = \delta_{ij}$, is enforced as the *fundamental commutator relation*

$$\{q_i, p_j\} = \delta_{ij} \quad \rightarrow \quad [\hat{x}_i, \hat{p}_j] = i\hbar\delta_{ij}. \tag{2.5}$$

It is striking to consider that the preceding three steps is all that is needed to take us from the classical Hamilton equations of motion, and to the Heisenberg equations of motion in the Heisenberg picture of quantum mechanics[5]. Indeed the classical Eq. (2.2) *directly* yields the quantum equation of motion by promoting the classical observables to operators, $q, p \rightarrow \hat{x}, \hat{p}$, and the Poisson brackets to commutator relations, $\{f, g\} \rightarrow -i/\hbar[\hat{f}, \hat{g}]$, as[6]

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \{A, \mathcal{H}\} \quad \rightarrow \quad \frac{\mathrm{d}A}{\mathrm{d}t} = \frac{1}{i\hbar}\left[\hat{A}, \hat{H}\right]. \tag{2.6}$$

Taking the expectation value (relative to some quantum state vector) of the Heisenberg equation of motion yields the familiar Ehrenfest theorem, which essentially

[5]The Heisenberg picture is a formulation of quantum mechanics in which the state vectors are all constant, but the operators evolve in time according to the Heisenberg equation of motion (the Heiseberg picture analogue to the Schrödinger equation).

[6]Assuming no *explicit* time dependence. If any such depenence is present, we need to add a $\partial A/\partial t$ term to the right hand side of both the classical and quantum equations.

states that the quantum *expectation values* evolve in time in the same way the classical observables do [44],

$$\left\langle \hat{A} \right\rangle = \frac{1}{i\hbar} \left\langle \psi \middle| \left[\hat{A}, \hat{H} \right] \middle| \psi \right\rangle . \tag{2.7}$$

As usual, we will denote by $\langle \cdot | \cdot \rangle$ the integral $\mathcal{L}^2$ inner product, while $\langle \cdot \rangle = \langle \psi | \cdot | \psi \rangle$ will denote the expectation value.

2.3 Schrödinger picture

The straight forward application of cannonical quantization, starting from Hamiltonian classical mechanics, landed us in the Heisenberg picture of quantum mechanics. This is the formulation of quantum theory in which the operators carry the time dependence, with state vectors being constant in time with the equation governing the evolution being the Heisenberg equation, Eq. (2.6). This is the quantum mechanics formalism that behaves the most like classical mechanics does; classically, the observables themselves, q and p (and derived quantities), evolve in time (the phase space point moves along a trajectory according to the Hamilton equations of motion).

But there is another, more familiar, formulation of quantum mechanics in which the operators are constant in time but the state vectors carry the time dependence. This is called the Schrödinger picture, with the corresponding equation of motion being the *Schrödinger equation.*

The wave function, a representation of the state vector which will be made rigorous in 2.6, $\Psi(\mathbf{R}; t)$ describes the state of a system at time t, where the vector $\mathbf{R}$ encodes all the relevant degrees of freedom of the system. The Schrödinger equation can be derived by employing two key assumptions: The state $\Psi(\mathbf{R}; t)$ evolves in time according to a *linear* and *unitary* time evolution operator, $\hat{\mathcal{U}}(t, t_0) : \mathcal{L}^2 \to \mathcal{L}^2$ such that $\Psi(\mathbf{R}; t) = \hat{\mathcal{U}}(t, t_0)\Psi(\mathbf{R}; t_0) \equiv \hat{\mathcal{U}}(t)\Psi(\mathbf{R})$ [41]. Two other physically motivated properties of $\hat{\mathcal{U}}$ are also assumed, namely that $\lim_{t \to t_0} \hat{\mathcal{U}}(t, t_0) = \mathbb{1}$ and $\hat{\mathcal{U}}(t_2, t_0) = \hat{\mathcal{U}}(t_2, t_1)\hat{\mathcal{U}}(t_1, t_0)$. These properties are all satisfied if we assume $\hat{\mathcal{U}}$ to take the form

$$\hat{\mathcal{U}}(t + \Delta t, t_0) = \mathbb{1} - i\hat{\Omega}\Delta t, \tag{2.8}$$

with $\hat{\Omega}$ being *some Hermitian operator* [45]. This is essentially nothing more than a guess, but guided by the intuition from the classical analogue of our system, we notice that $\hat{\Omega}$ has dimensions of frequency and postulate that we are really dealing with $\hat{H}/\hbar$. This is after all pretty natural, since the classical Hamiltonian is what governs time evolution before the quantization.

Taking the composition of $\hat{\mathcal{U}}(t_2, t_1)$ and $\hat{\mathcal{U}}(t_1, t_0)$ with $t_1 \to t$ and $t_2 \to t_1 + \Delta t$

now yields

$$\hat{\mathcal{U}}(t+\Delta t,t)\hat{\mathcal{U}}(t,t_0)=\hat{\mathcal{U}}(t+\Delta t,t_0)=\left(\mathbb{1}-\frac{i\hat{H}\Delta t}{\hbar}\right)\hat{\mathcal{U}}(t,t_0) \tag{2.9}$$

which we can rearrange as

$$\hat{\mathcal{U}}(t+\Delta t,t_0)-\hat{\mathcal{U}}(t,t_0)=-\Delta t\frac{i}{\hbar}\hat{H}\hat{\mathcal{U}}(t,t_0). \tag{2.10}$$

Dividing by Δt and taking the limit $\Delta t \to 0$ yields the familiar definition of the derivative of $\hat{\mathcal{U}}(t,t_0)$ in terms of the Hamiltonian, i.e.

$$i\hbar\frac{\partial}{\partial t}\hat{\mathcal{U}}(t,t_0)=\hat{H}\hat{\mathcal{U}}(t,t_0). \tag{2.11}$$

This is known as the Schrödinger equation for the time evolution operator, $\hat{\mathcal{U}}$ and is the fundamental equation from which all things connected to time evolution follows [45].

The more familiar Schrödinger equation which govern the time evolution of states emerges after we right-multiply by the wavefunction $\Psi(\mathbf{R})$,

$$\begin{aligned} i\hbar\frac{\partial}{\partial t}\hat{\mathcal{U}}(t,t_0)\Psi(\mathbf{R})&=\hat{H}\hat{\mathcal{U}}(t,t_0)\Psi(\mathbf{R})\\ i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{R};t)&=\hat{H}\Psi(\mathbf{R};t). \end{aligned} \tag{2.12}$$

We will denote Eq. (2.12) by the time dependent Schrödinger equation (TDSE). For the important special case where $\hat{H}$ is time independent, the TDSE is separable in spatial and temporal variables and admits the formal solution [14]

$$\Psi(\mathbf{R};t)=\hat{\mathcal{U}}(t,t_0)\Psi(\mathbf{R})=e^{-it\hat{H}/t}\Psi(\mathbf{R}). \tag{2.13}$$

It is a central postulate of quantum mechanics that any observable is associated with a Hermitian operator, $\hat{O}$, and that it's spectrum spans the entirety of $\mathcal{L}^2$. This will be discusssed more thorougly in section 2.6, but for now we will anticipate things to come and use the *completeness* of the operator $\hat{O}$'s spectrum:

$$\sum_i |O_i\rangle\langle O_i| = \mathbb{1}. \tag{2.14}$$

Let us now consider the energy, with corresponding Hermitian operator $\hat{H}$. The spectrum, the energy-eigenstates, are labeled by $|E_i\rangle$. Inserting the unity of Eq. (2.14) realized in terms of the energy eigen-states on both sides of the exponential expression

for $\hat{\mathcal{U}}(t, t_0)$ yields

$$\begin{aligned} e^{-it\hat{H}/\hbar} &= \sum_i \sum_j |E_i\rangle\langle E_i| e^{-it\hat{H}/\hbar} |E_j\rangle\langle E_j| \\ &= \sum_i \sum_j |E_i\rangle\langle E_i| \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{t\hat{H}}{i\hbar} \right)^n |E_j\rangle\langle E_j|, \end{aligned} \tag{2.15}$$

where we have used the normal definition of the exponential in terms of it's power series $e^x = \sum_n x^n/n!$. Since $\hat{H}|E_i\rangle = E_i|E_i\rangle$ and $\hat{H}^n = E_i\hat{H}^{n-1}|E_i\rangle = E_i^2\hat{H}^{n-2}|E_i\rangle = \cdots = E_i^n|E_i\rangle$, we find from Eq. (2.15) that

$$\begin{aligned} e^{-it\hat{H}/\hbar} &= \sum_i \sum_j |E_i\rangle\langle E_i| \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{t\hat{E}_i}{i\hbar} \right)^n |E_j\rangle\langle E_j| \\ &= \sum_i \sum_j e^{-itE_i/\hbar} |E_i\rangle \underbrace{\langle E_i|E_j\rangle}_{\delta_{ij}} \langle E_j| \\ &= \sum_i |E_i\rangle e^{-itE_i/\hbar} \langle E_i|. \end{aligned} \tag{2.16}$$

This shows that if we can somehow find the energy-eigenstates and expand our wave function in this basis, finding the time evolution, governed by the TDSE, is trivial [45]. Applying the result from Eq. (2.16) in Eq. (2.13) gives

$$\begin{aligned} \Psi(\mathbf{R}; t) &= \hat{\mathcal{U}}(t, t_0)\Psi(\mathbf{R}) = e^{-it\hat{H}/\hbar}\Psi(\mathbf{R}) \\ &= \sum_i |E_i\rangle e^{-itE_i/\hbar} \langle E_i| \left(\sum_j |E_j\rangle\langle E_j|\Psi\rangle \right) \\ &= \sum_i \sum_j |E_i\rangle e^{-itE_i/\hbar} \underbrace{\langle E_i|E_j\rangle}_{\delta_{ij}} \langle E_j|\Psi\rangle = \sum_i |E_i\rangle e^{-itE_i/\hbar} \langle E_i|\Psi\rangle. \end{aligned} \tag{2.17}$$

As illustrated above, the problem of computing the time evolution of a state when the eigen-states of the Hamiltonian are known consists of calculating a series of integrals ($\langle E_i|\Psi\rangle$). For us, however, finding the eigen-states and the corresponding eigenvalues will be the fundamental task. The governing equation is simply the eigenvalue equation

$$\hat{H}|\Psi\rangle = E|\Psi\rangle, \tag{2.18}$$

which is called the time independent Schrödinger equation (TISE). This is the spatial result of the separation of variables we used to derive the TDSE [46].

Canonical quantization as a semi-classical model of quantum mechanics

In short, first (or canonical) quantization describes an attempt to construct a mathematical formulation of a quantum mechanical system emergent from the classical description. It is important to note however that this is essentially a semi-classical formulation in that the surrounding environment is treated classicaly. E.g. the first quantized formulation of N electrons confined in a harmonic oscillator treats the external oscillator potential in a classical manner—despite the fact that this potential inevitbly arises from some *quantum* effect.

A general assumption is that the surrounding system is *large* enough to be accurately described by a classical treatment.

2.4 The quantum Hamiltonian

Since the Hamiltonian is the fundamental quantity which governs the dynamics of any quantum mechanical system, the natural question is now: What does it look like? In the simplest possible case, a free particle of mass m constrained to move in one spatial dimension, it takes the form

$$\hat{H}_{\text{free}} = -\frac{\hbar^2}{2m}\frac{\partial}{\partial x}, \tag{2.19}$$

a straight forward to application of the operator promotion scheme for $p \to \hat{p} = -i\hbar(\partial/\partial x)$. For a particle of mass m moving in a position dependent potential $V(x)$, the Hamiltonian takes the form

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial}{\partial x} + V(x). \tag{2.20}$$

In the present work, we will be focused on systems of interacting electrons in the presence of one or more positively charged nuclei. Under first quantization, the classical Hamiltonian of N particles of mass m_e moving in an external potential changes from that shown in Eq. (2.3). The total kinetic energy of all N electrons with masses m_e becomes

$$\hat{T} = \sum_{i=1}^{N} -\frac{\hbar^2}{2m_e}\nabla_i^2, \tag{2.21}$$

where ∇_i denotes differentiation w.r.t. the coordinates of particle i. Since the electrons are negatively charged, the inter-particle potential will be the Coulomb potential

$$\hat{W} = \sum_{i=1}^{N}\sum_{j=i+1}^{N} \frac{k_e e^2}{|\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j|}, \tag{2.22}$$

where $k_e = 1/4\pi\varepsilon_0$ is the Coulomb constant. The sum limits ensure no double counting is done. The positive nuclei give rise to a similar Coulomb potential, namely

$$\hat{V} = -\sum_{i=1}^{N}\sum_{A=1}^{M}\frac{k_e Z_A e^2}{|\hat{\mathbf{r}}_A - \hat{\mathbf{r}}_i|}, \tag{2.23}$$

with M denoting the number of nuclei of (possibly differing) charge(s) $+Z_A e$. Putting it all together yields the total electron and nuclear Hamiltonian

$$\begin{aligned}\hat{H} = -\sum_{i=1}^{N}\frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_{A=1}^{M}\frac{\hbar^2}{2m_A}\nabla_A^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{k_e Z_A e^2}{|\hat{\mathbf{r}}_A - \hat{\mathbf{r}}_i|} + \\ \sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{k_e e^2}{|\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j|} + \sum_{A=1}^{M}\sum_{B=A+1}^{M}\frac{k_e e^2 Z_A Z_B}{|\mathbf{r}_A - \mathbf{r}_B|}.\end{aligned} \tag{2.24}$$

The mass of nucleus A is here denoted m_A.

2.4.1 Accuracy of molecular Hamiltonian

Of course, there are some imperfections. We have for example not included any relativistic effects. The classical non-relativistic kinetic energy term takes the form $|\mathbf{p}|^2/2m = m\mathbf{v}^2/2$. However, in order to account for relativistic effects, we should really use

$$T_{\text{classical, relativistic}} = \frac{mc^2}{\sqrt{1-(v/c)^2}} - mc^2, \tag{2.25}$$

with c being the vacuum speed of light. Expressing T in terms of the relativistic momentum yields

$$\begin{aligned}T_{\text{classical, relativistic}} &= \sqrt{p^2c^2 + m^2c^4} - mc^2 = mc^2\left[\sqrt{1+\left(\frac{p}{mc}\right)^2} - 1\right] \\ &= mc^2\left[1 + \frac{1}{2}\left(\frac{p}{mc}\right)^2 - \frac{1}{8}\left(\frac{p}{mc}\right)^3 + \cdots - 1\right] \\ &= \frac{p^2}{2m} - \frac{p^4}{8m^3c^2} + \mathcal{O}\left(\frac{p^6}{m^5c^4}\right).\end{aligned} \tag{2.26}$$

Evidently, the first order relativistic correction to the Hamiltonian is on the order of p^4/m^3c^2 [46].

We may also consider the charged nuclei moving (in the frame of refence of an electron) setting up a magnetic field $\mathbf{B}$. The energy involved in the interaction between this magnetic field and the dipole moment of the electron, μ_e, is $\hat{H}_{\text{spin-orbit}} =$

$\mu_e \cdot \mathbf{B}$. This is called spin-orbit coupling. The electron magnetic dipole moment has the magnitude

$$\mu_e = -\frac{e}{m_e}\mathbf{S}, \tag{2.27}$$

with $\mathbf{S}$ being the electron's spin angular momentum. The magnetic field strenght set up by the (apparent) motion of the nucleus (relative to the electron) is

$$\mathbf{B} = \frac{Zk_e e}{mc^2 r^3}\mathbf{L}, \tag{2.28}$$

with $\mathbf{L}$ being the electron's orbital angular momentum. The correction to the Hamiltonian due to this effect becomes [46]

$$H_{\text{spin-orbit}} = \frac{Zk_e e^2}{2m_e^2 c^2 r^3}\mathbf{S}\cdot\mathbf{L}. \tag{2.29}$$

Combining both effects gives what is know as the *fine structure*.

There is an additional relativistic effect which is derived by expanding the Dirac equation Hamiltonian in powers of mc^2 and taking the non-relativistic limit. This is called the Darwin term, and for a central potential it can be written as [47]

$$H_{\text{Darwin}} = -\frac{\pi\hbar^2 k_e e^2}{2m_e^2 c^2}\delta^3(\mathbf{r}-\mathbf{r}_A), \tag{2.30}$$

with $\delta^3(\mathbf{r}-\mathbf{r}_A)$ being a Dirac delta function at the position of the nucleus. Since only $n = 1$ (s symmetry) states have a non-vanishing magnitude at the origin, the Darwin term only affects s states.

Furthermore we may take into account the Lamb shift, that is the splitting of 2s and 2p states which is related to the quantization of the electric field. The charged electrons interact with the vacuum fluctuations of the quantized electromagnetic field which partially shield the Coulomb interactions between the electrons and the nuclei [46, 48]. Even though quantum field theory is needed in order to handle this effect for real[7], we may use an effective Hamiltonian term which generates the shift [50]

$$\hat{H}_{\text{Lamb shift}} = \frac{4}{3}\alpha^2 m_e c^2 \left(\frac{\hbar}{m_e c}\right)^3 \ln(1/\alpha^2)\delta^3(\mathbf{r}-\mathbf{r}_A), \tag{2.31}$$

with $\alpha = k_e e^2/\hbar c \simeq 1/137$ denoting the *fine structure constant*.

Finally, let us consider the interaction between the electrons and the magnetic dipoles of the protons. The proton has a magnetic dipole of magnitude $\mu_p = (g_p e/2m_p)\mathbf{S}_p$ and sets up the magnetic field (according to classical electrodynamics) [46]

$$\mathbf{B} = \frac{k_e\mu_0}{r^3}\left[3(\mu\cdot\hat{r})\hat{r}-\mu\right] + \frac{2\mu_0}{3}\mu\delta^3(\mathbf{r}-\mathbf{r}_A), \tag{2.32}$$

[7] In fact, the Dirac equation does not predict the shift, but a semi-classical model due to Welton does [47, 49].

Table 2.1: Order of magnitude energy corrections to the Hamiltonian due to various effects un-accounted for by the ordinary atomic Hamiltonian.

Correction term	Expression	Dimensions	Approximate magnitude $[E_h]$
$\hat{H}_{\text{Relativistic}}$	$-\frac{p^4}{8m^3c^2}$	$\frac{m_e k_e^4 e^8}{\hbar^4 c^2}$	$5 \cdot 10^{-6}$
$\hat{H}_{\text{Spin-orbit}}$	$\frac{Z k_e e^2}{2m_e^2 c^2 r^3}\mathbf{S}\cdot\mathbf{L}$	$\frac{k_e e^2 \hbar^2}{m_e^2 c^2 a_0^3}$	$5 \cdot 10^{-5}$
$\hat{H}_{\text{Darwin}}$	$-\frac{\pi\hbar^2 k_e e^2}{2m_e^2 c^2}\delta^3(\mathbf{r}-\mathbf{r}_A)$	$\frac{\hbar^2 k_e e^2}{m_e^2 c^2 a_0^3}$	$5 \cdot 10^{-5}$
$\hat{H}_{\text{Lamb}}$	$\frac{4}{3}\alpha^2 m_e c^2 \left(\frac{\hbar}{m_e c}\right)^3 \ln(1/\alpha^2)\delta^3(\mathbf{r}-\mathbf{r}_A)$	$\frac{\alpha^2\hbar^3}{m_e^2 c a_0^3}$	$1 \cdot 10^{-6}$
$\hat{H}_{\text{hyperfine}}$	Eq. (2.33)	$\frac{\mu_0 e^2 \hbar^2}{m_p m_e a_0^3}$	$5 \cdot 10^{-7}$

where μ_0 denotes the permeability of free space and $g_p \approx 5.59$ is the gyromagnetic ratio of the proton. The Hamiltonian correction term becomes

$$\hat{H}_{\text{hyperfine}} = \frac{\mu_0 g_p e^2}{m_p m_e}\left(\frac{[3(\mathbf{S}_p\cdot\hat{r})(\mathbf{S}_e\cdot\hat{r}) - \mathbf{S}_p\cdot\mathbf{S}_e]}{8\pi r^3} + \frac{1}{3}\mathbf{S}_p\cdot\mathbf{S}_e\delta^3(\mathbf{r}-\mathbf{r}_A)\right). \quad (2.33)$$

We may now use dimensional analysis in order to work out the rough size of these effects relative to the ground state energies of ordinary atoms. We will take electronic distances to be on the order of the Bohr radius, $r \sim a_0$ and use the natural time scale $t \sim \hbar a_0/k_e e^2$ in order to work out the typical momentum of an electron as $p \sim m_e k_e e^2/\hbar$. The hydrogenic radial wave function of principal quantum number n takes the value $R_n(\mathbf{0}) = 4Z^3/a_0^3 n^3$, so we take the delta function to be on the order of $\delta^3(\mathbf{r}) \sim 1/a_0^3$ [46]. All $\mathbf{S}$ and $\mathbf{L}$ terms carry units of $\hbar$.

Rough back of the envelope estimations of the order of magnitude of the correction terms are shown in Table 2.1. The binding energy of e.g. hydrogen is on the order of $\sim 1E_h$, so these are all miniscule compared to the typical energy scale we are working with.

2.4.2 Born-Oppenheimer approximation

Let us now consider again the atomic Hamiltonian of Eq. (2.24). For a system of N electrons in the presence of M nuclei in three spatial dimensions, we have a system

of $3(N+M)$ degrees of freedom (treating the nuclei as point particles). The solution of the TDSE is the wave function, a function depending on at least $3(N+M)$ variables. If we were to seek the solution to the Schrödinger equation for, say an isolated caffeine molecule $C_8H_{10}N_4O_2$, then $\Psi(\mathbf{R};t)$ would depend on a bit more than 300(!) coordinates. This is an unbelievably monumentous task, and we are only asking for the solution of an *isolated*, fairly small molecule consisting of first row atoms. As it turns out, the wave function is a "very, very, *very* complicated function" and solving the Schrödinger equation is a very, very, *very* hard problem for all but the smallest of systems [14]. This was in fact realized in the very early days of quantum mechanics. Dirac, for example, wrote in 1929 that [51]

> “The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these equations leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation.”
>
> *P. A. M. Dirac*

In general, for an arbitrary Hamiltonian (for example the full Hamiltonian of Eq. (2.18) with the five correction terms of section 2.4.1), solving the corresponding Schrödinger equation is a problem classified in complexity theory as **NP**-hard [52]. Heuristically, we may consider this to be a problem that is not feasably soluble *even in theory* using a deterministic algorithm.[8] As it turns out, we will need to make certain simplifications before we can proceed.

In a molecular system, little momentum transfer will happen between the nuclei and the electrons on account of their differing masses [16]. As protons are almost 2000 times heavier than electrons, their movement will be drastically slower than their lighter counterparts (assuming the momenta of the two are similar). As the electrons will relax to a stationary state much faster than the typical time scales involved in nucleonic motion, we may assume the nucleus to be effectively fixed and treat only the electronic degrees of freedom. This is very similar to the idea of a quasistatic process in thermodynamics in where change is forced upon a system sufficiently slowly to allow it to continuously equilibrate throughout: at every instant, we may regard the system to be equilibrated despite the continual change it is undergoing [53]. In the framework of quantum mechanics, this is known as the adiabatic theorem

[8] The **P** class problems are problems which can be solved in *polynomial* time using a deterministic algorithm. Assuming that **P**≠**NP** (which is belived to be true), it means that **NP** problems cannot and **NP**-hard problems are by definition as hard as the hardest **NP** problems. In reality, this most likely means we are stuck with a theoretical best case scenario of an algorithm with runs in exponential (w.r.t. the system size) time.

and sometimes the fixed nuclei approximation is called an adiabatic approximation [45].

The separation of the nuclear and electronic degrees of freedom is known as the Born-Oppenheimer approximation[54]. Recall that the full molecular Hamiltonian reads (Eq. (2.24))

$$\hat{H} = -\sum_{i=1}^{N} \frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_{A=1}^{M} \frac{\hbar^2}{2m_A}\nabla_A^2 - \sum_{i=1}^{N}\sum_{A=1}^{M} \frac{k_e Z_A e^2}{|\hat{\mathbf{r}}_A - \hat{\mathbf{r}}_i|} + \sum_{i=1}^{N}\sum_{j=i+1}^{N} \frac{k_e e^2}{|\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j|} + \sum_{A=1}^{M}\sum_{B=A+1}^{M} \frac{k_e e^2 Z_A Z_B}{|\mathbf{r}_A - \mathbf{r}_B|}. \tag{2.34}$$

Taking the limit $m_A \to \infty$ freezes out the proton motion, creating stationary *clamped nuclei*. The Born-Oppenheimer Hamiltonian is given as

$$\hat{H}_{\text{Born-Oppenheimer}} = -\sum_{i=1}^{N} \frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M} \frac{k_e Z e^2}{|\hat{\mathbf{r}}_A - \hat{\mathbf{r}}_i|} + \sum_{i=1}^{N}\sum_{j=i+1}^{N} \frac{k_e e^2}{|\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j|} + \text{constant}, \tag{2.35}$$

where the constant is just the nucleus-nucleus interaction term,

$$\text{constant} = \sum_{A=1}^{M}\sum_{B=A+1}^{M} \frac{k_e e^2 Z_A Z_B}{|\mathbf{r}_A - \mathbf{r}_B|}. \tag{2.36}$$

If the nuclei are all stationary, this is just a constant and can be disregarded as a constant term in the Hamiltonian wont affect the dynamics. Crucially, the Born-Oppenheimer Hamiltonian and the nuclei position operators $\hat{\mathbf{R}}_A$ commute so it is possible to find solutions which are simultaneously eigenfunctions of $\hat{H}_{\text{Born-Oppenheimer}}$ *and* have a definite value of the $\mathbf{R}_A$s [55]. It is important to note that solution of the corresponding Schrödinger equation now depends only *parametrically* on the nuclei positions, $\mathbf{R}_A$. This reduces the number of degrees of fredom by $3M$ [22].

If we are able to somehow find a solution to the electronic Schrödinger equation, $\Psi(\mathbf{R}; \mathbf{R}_A)$, it would then be possible to use a similar argument as before to find a solution to the nuclear Schrödinger equation. When considering the motion of the nucleus, it is reasonable to approximate the electronic influence by an averaging of their coordinates over the electronic wave function [22]. This gives the nuclear

Hamiltonian

$$\begin{aligned}\hat{H}_{\text{Nuclear}} &= -\sum_{A=1}^{M}\frac{\hbar^2}{2m_A}\nabla_A^2 + \sum_{A=1}^{M}\sum_{B=A+1}^{M}\frac{k_e e^2 Z_A Z_B}{|\mathbf{r}_A - \mathbf{r}_B|} \\ &\quad + \left\langle -\sum_{i=1}^{N}\frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{k_e Z e^2}{|\hat{\mathbf{r}}_A - \hat{\mathbf{r}}_i|} + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{k_e e^2}{|\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j|}\right\rangle \\ &= -\sum_{A=1}^{M}\frac{\hbar^2}{2m_A}\nabla_A^2 + \sum_{A=1}^{M}\sum_{B=A+1}^{M}\frac{k_e e^2 Z_A Z_B}{|\mathbf{r}_A - \mathbf{r}_B|} + E_{\text{electronic}}(\mathbf{R}_A). \end{aligned} \tag{2.37}$$

The key insight to glean from this is that the electronic energy acts as an effective potential for the nucleonic motion: in the words of Szabo & Ostlund "the nuclei in the Born-Oppenheimer approximation move on the potential energy surface obtained by solving the electronic problem" [22, 55]. This lies at the heart of, and forms the foundation for, most of quantum chemistry. Note carefully that all we need to find is the energy, in principle we dont need to concern ourselves with the elctronic wave function. All molecular dynamics (disregarding vibration and rotation) emerge effectively from $E_{\text{electronic}}(\mathbf{R}_A)$.

Accuracy of the Born-Oppenheimer approximation

In somewhat more tehcnical terms, the Born-Oppenheimer approximation can be stated as: Assuming the full molecular Schrödinger equation has a solution on the form $\psi(\mathbf{R}, \mathbf{R}_A) \equiv \Psi(\mathbf{R})\Phi(\mathbf{R}_A)$, with $\Psi(\mathbf{R})$ being the solution of the electronic problem, then insertion of this total wave function into the full Hamiltonian yields

$$\begin{aligned}\hat{H}_{\text{full}}\,\psi(\mathbf{R}, \mathbf{R}_A) &= \left[\hat{T}_{\text{electronic}} + \hat{T}_{\text{nucleonic}} + \hat{V}(\mathbf{R}) + \hat{V}(\mathbf{R}_A) + \hat{V}(\mathbf{R}, \mathbf{R}_A)\right]\Psi(\mathbf{R})\Phi(\mathbf{R}_A) \\ &= \Big[\underbrace{\hat{T}_{\text{nucleonic}} + \hat{V}(\mathbf{R}_A)}_{\hat{H}_{\text{nucleonic}}} + \hat{H}_{\text{electronic}}\Big]\Psi(\mathbf{R})\Phi(\mathbf{R}_A). \end{aligned} \tag{2.38}$$

Since $\hat{T}$ involves a differential operator, we need to consider the operation of a differentiation of $\Psi(\mathbf{R})$ w.r.t. $\mathbf{R}_A$ and $\Phi(\mathbf{R}_A)$ w.r.t. $\mathbf{R}$. It turns out that the latter vanishes exactly, however the former will have a non-zero contribution to the total molecular energy [55]. The Born-Oppenheimer approximation consists of disregarding this differential cross-term and taking the full Hamiltonian acting on the product state to be [45]

$$\left[\hat{H}_{\text{nucleonic}} + E_{\text{electronic}}(\mathbf{R}_A)\right]\Phi(\mathbf{R}_A) \approx E_{\text{Total}}\Phi(\mathbf{R}_A). \tag{2.39}$$

Let us now consider for a moment how much effect this intentional oversight will have on the total energy of the system. Again we turn to dimensional analysis in order

Atom	Atomic number	supression ratio
H	1	0.1527
He	2	0.1081
Be	4	0.0883
Ne	10	0.0722
Ar	18	0.0608
Sn	50	0.0463
U	92	0.0389
Og	118	0.0369

Table 2.2: Values of the Born-Oppenheimer supression ratio, $S = (m_e/m_A)^{1/4}$ for various different atomic systems. Note the *very* slow scaling of quarter power, despite the widely differing masses.

to get a rough estimate. For the electronic wave function, we take the characteristic length to be the only combination of $\hbar$, m_e, k_e, and e with units of distance. This is the Bohr radius, $a_0 = \hbar^2/m_e e^2$. For the vibrational nucleonic motion however, the characteristic length is taken to be $b_0 \equiv \hbar^2/e^2 m_A^{1/4} m_e^{3/4}$ [55]. Differentiation w.r.t. $\mathbf{R}_A$ will then yield appreciable change in $\Psi(\mathbf{R})$ if the change is on the order of a_0, but differentiation of $\Phi(\mathbf{R}_A)$ will cause it to change appreciably if the the variations in $\mathbf{R}_A$ are on the order of b_0. It is thus the ratio a_0 to b_0 which determines how good of an approximation the Born-Oppenheimer scheme is. As b_0 depends, albeit weakly, on the typical nucleonic mass, this changes depending on the molecular system in question.

The ratio

$$S \equiv \frac{b_0}{a_0} = \frac{\left(\frac{\hbar^2}{e^2 m_A^{1/4} m_e^{3/4}}\right)}{\left(\frac{\hbar^2}{m_e e^2}\right)} = \left(\frac{m_e}{m_A}\right)^{1/4} \tag{2.40}$$

depends on the mass ratio to the one quarter power. Despite the almost three order of magnitude difference in the electron and proton masses, the low exponent means this *supression ratio* worryingly large values for small atoms. The S ratio is calculated for a few atoms of differing sizes in Table 2.2.

However, Born-Oppenheimer works because vibrational energies of atoms arent large enough to excite electrons across electronic energy levels. The vibrational energies are smaller than the electronic ones roughly by a factor of $\sqrt{m_e/m_A}$. Even larger is the discrepancy between typical electronic energy scales and those of molecular rotational energies, which carry an additional factor of $\sqrt{m_e/m_A}$ meaning they are an overall factor of m_e/m_A smaller [55].

2.5 Anti-symmetry and the Pauli principle

Let us now consider a system of N *indistinguishable* particles, i.e. particles that are fundamentally identical as to make telling them apart from each other is impossible. If our theory is to handle such a system with any logical consistency, we must require our wave function (and thus also the observables we derive from it) to be *permutation invariant* (up to a phase factor, which does not affect the physics [as per the postulates of QM]). If we cannot tell the particles apart, it does not make sense to say particle one is *here*, while particle two is *over there*—essentially we need our theory to account for both cases simultaneously. We may do this by constructing a wave function that is non-commital as to which particle is where [46].

Following Kvaal we may define this mathematically by defining $\sigma \in S_N$, a permutation of the indices in a set of N such indices [14]. S_N here denotes the symmetric group of degree N.[9] We must demand that $|\Psi|^2$ be permutation invariant, that is

$$\begin{aligned} |\Psi(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_N)|^2 &= \left|\Psi(\mathbf{r}_{\sigma(1)}, \mathbf{r}_{\sigma(2)}, \mathbf{r}_{\sigma(3)}, \ldots, \mathbf{r}_{\sigma(N)})\right|^2 \\ \Rightarrow \quad \Psi(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_N) &= \alpha\Psi(\mathbf{r}_{\sigma(1)}, \mathbf{r}_{\sigma(2)}, \mathbf{r}_{\sigma(3)}, \ldots, \mathbf{r}_{\sigma(N)}), \end{aligned} \tag{2.41}$$

with $\alpha \in \mathbb{C}$ (possibly σ-dependent) with $|\alpha| = 1$.

For each permutation in S_N, we define a linear operator $\hat{P}_\sigma$ that evaluates the wave function with permuted indices

$$\hat{P}_\sigma \left[\Psi(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_N)\right] = \Psi(\mathbf{r}_{\sigma(1)}, \mathbf{r}_{\sigma(2)}, \mathbf{r}_{\sigma(3)}, \ldots, \mathbf{r}_{\sigma(N)}. \tag{2.42}$$

Thus we can formulate particle indistinguishability in terms of $\hat{P}_\sigma$ by demanding that $\Psi(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_N)$ be an eigenfunction of $\hat{P}_\sigma$. According to the *postulates* of quantum mechanics, a fermionic wave function is totally anti-symmetric w.r.t. exchange of particles, meaning this eigenvalue is $(-1)^{|\sigma|}$, with $|\sigma|$ being the minimal number of *transpositions*[10] needed to perform the full permutation σ. This is known as the **Pauli exclusion principle**.

2.5.1 Slater determinants

A one-electron system is described by a one-body Hamiltonian operator, $\hat{h}$. This one-body Hamiltonian is a differential operator w.r.t. the coordinates of this one electron (kinetic energy operator), and possibly contains an external potential. It's spectrum forms a set of single-particle wave functions, $\phi_i(\mathbf{r})$. According to the postulates of QM, this set is complete in the sense that it spans the enclosing Hilbert space. We will denote the ϕ_is by spatial *orbitals*.

[9] The symmetric group on a finite set M is a mathematical group, and consists of all possible permutations of the elements of the set M. Mathematically, these permutations are bijections from M onto M itself. There are $N!$ unique permutations in the group, including the identity permutation (which just leaves the set un-changed).

[10] A transposition is defined as a permutation of *only two indices*.

In order to completely specify the quantum state of an electron, it is neccessary to also specify it's spin state. The electron being a spin-$1/2$ particle, it admits only two distrinct spin projections: Spin up, $\chi(\uparrow)$, and spin down, $\chi(\downarrow)$ [46]. The wave function of the electron, completely specifying all of it's properties is the product of a spatial orbital and a spin function—a **spin-orbital**—$\psi_i(\mathbf{r}, \sigma) = \phi(\mathbf{r})\chi(\sigma)$, with σ labelling the spin-projection. Mathematically, we may consider the original Hilbert space of the spatial orbitals to be extended (in the cartesian product sense) with a simple two-dimensional Hilbert space of spin-states. Thus strictly speaking we should be writing the spin-orbitals as direct products, $\phi_i(\mathbf{r}) \otimes \chi(\sigma)$, but we will ommit this notation for the entirety of the present work and let simple juxtapositioning represent this product.

Assume now that a many-electron quantum system is described by a Hamiltonian of the form

$$\hat{H} = \sum_{i=1}^{N} \hat{h}_i = \sum_{i=1}^{N} \left[-\frac{\nabla_i^2}{2} + V_{\text{ext}}(\mathbf{r}) \right], \tag{2.43}$$

where the operators $\hat{h}_i$ act only on the coordinates of particle i. It is natural to take the one-electron states as a starting point when looking for the spectrum of the full Hamiltonian, $\hat{H}$. And in fact, it turns out that a simple product

$$\Psi(\mathbf{R}, \boldsymbol{\sigma}) = \psi_a(\mathbf{r}_1, \sigma_1)\psi_b(\mathbf{r}_2, \sigma_2) \cdots \psi_c(\mathbf{r}_N, \sigma_N) \tag{2.44}$$

is an eigenfunction of $\hat{H}$: it is known as a *Hatree product* [22]. The vector of spin labels, $\boldsymbol{\sigma} = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$, here denotes the collection of all spin projections in the same way $\mathbf{R}$ denotes the collection of all electronic coordinates. However, the Hartree product does not have the correct (anti-)symmetry. In particular, the product assigns to specific electrons (labelled $1, 2, 3, \ldots, N$) specific quantum states (labelled $\psi_a, \psi_b, \ldots$). In accordance with Pauli, we may anti-symmetrize the Hartree product by considering every possible permutation of the electron labels, [14]

$$\Psi(\mathbf{R}, \boldsymbol{\sigma}) = \frac{1}{\sqrt{N!}} \sum_{\gamma \in S_N} (-1)^{|\gamma|} \hat{P}_\gamma \psi_a(\mathbf{r}_1, \sigma_1)\psi_b(\mathbf{r}_2, \sigma_2) \cdots \psi_c(\mathbf{r}_N, \sigma_N). \tag{2.45}$$

The $|\gamma|$ factor denotes the total number of transpositions in $\hat{P}_\gamma$, and $(-1)^{|\gamma|}$ is the sign of the permutation. The normalization is neccessary in order to ensure $\langle \Psi | \Psi \rangle = 1$, as there are $N!$ total possible permutations we must consider. Eq. (2.45) is known as a **Slater determinant**, and is more commonly written as [56]

$$\Psi(\mathbf{R}, \boldsymbol{\sigma}) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(\mathbf{r}_1) & \psi_2(\mathbf{r}_1) & \psi_3(\mathbf{r}_1) & \ldots & \psi_N(\mathbf{r}_1) \\ \psi_1(\mathbf{r}_2) & \psi_2(\mathbf{r}_2) & \psi_3(\mathbf{r}_2) & \ldots & \psi_N(\mathbf{r}_2) \\ \psi_1(\mathbf{r}_3) & \psi_2(\mathbf{r}_3) & \psi_3(\mathbf{r}_3) & \ldots & \psi_N(\mathbf{r}_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \psi_1(\mathbf{r}_N) & \psi_2(\mathbf{r}_N) & \psi_3(\mathbf{r}_N) & \ldots & \psi_N(\mathbf{r}_N) \end{vmatrix}. \tag{2.46}$$

We could of course have allowed the permutations to operate on the spin-orbital labels instead of the electron labels, leading to analogous expressions. It is important however that it act on only electron labels *or* orbital labels, since acting on both simultaneously leaves the product entirely unchanged [57].

Before moving on, let us note some key properties of the Slater determinant. Interchanging two electrons give rise to an overall minus sign, since this corresponds to interchanging two rows of the determinant [58]. Some authors choose to represent the Slater determinant with rows corresponding to orbitals, and columns representing electron labels. The two representations are equal, since the determinant is invariant under tranposition [43]. The Hatree products form a basis for the full $\mathcal{L}^2$ Hilbert space of the N-electron system, and the Slater determinants span the sub-space of anti-symmetric functions in the same space [14]. This means we can represent *any* square integrable, normalized, anti-symmetric function of N electronic coordinates as a sum over Slaters,

$$f(\mathbf{R}, \boldsymbol{\sigma}) = \sum_{i=1}^{\infty} c_i \Psi_i(\mathbf{R}, \boldsymbol{\sigma}). \tag{2.47}$$

Some numbering $\{\Psi_i\}_{i=1}^{\infty}$ has here been chosen. We note that this is valid only when the entire spectrum of $\hat{h}$ is considered for the single-particle orbitals occupying the determinant.

The Slater determinant is normalized whenever the spin-orbitals are normalized. Consider one of the integral terms in the total $\langle \Psi(\mathbf{R}, \boldsymbol{\sigma}) | \Psi(\mathbf{R}, \boldsymbol{\sigma}) \rangle$, i.e.

$$\begin{aligned}
\langle \psi_{i_1} \psi_{i_2} \dots \psi_{i_N} | \psi_{j_1} \psi_{j_1} \dots \psi_{j_1} \rangle &= \int \mathrm{d}\mathbf{x}_1 \dots \mathrm{d}\mathbf{x}_N \, \psi^*_{i_1}(\mathbf{r}_1, \sigma_1) \psi_{j_1}(\mathbf{r}_1, \sigma_1) \\
&\qquad \cdots \psi^*_{i_N}(\mathbf{r}_N, \sigma_N) \psi_{j_N}(\mathbf{r}_N, \sigma_N) \\
&= \int \mathrm{d}\mathbf{x}_1 \psi^*_{i_1}(\mathbf{r}_1, \sigma_1) \psi_{j_1}(\mathbf{r}_1, \sigma_1) \\
&\qquad \int \mathrm{d}\mathbf{x}_2 \psi^*_{i_2}(\mathbf{r}_2, \sigma_2) \psi_{j_2}(\mathbf{r}_2, \sigma_2) \cdots \\
&\qquad\qquad \int \mathrm{d}\mathbf{x}_N \psi^*_{i_N}(\mathbf{r}_N, \sigma_N) \psi_{j_N}(\mathbf{r}_N, \sigma_N) \\
&= \delta_{i_1 j_1} \delta_{i_2 j_2} \cdots \delta_{i_N j_N}.
\end{aligned} \tag{2.48}$$

The $\mathbf{x}$ coordinate here denotes the collection of spatial and spin coordinates for each electron, $\mathbf{x} = \{\mathbf{r}, \sigma\}$, and position in the string of orbitals—$|\psi_1 \psi_2 \dots \psi_N\rangle$—determines which electron occupies which state. E.g. in the state $|\psi_i \psi_j \psi_k\rangle$, spin-orbital i is evaluated at the coordinates of electron one, the j orbital at electron two, and the k orbital at the coordinates of electron three. The last equality follows from the spin-orbitals

being orthonormal. In terms of the total Slater, this means[11]

$$\begin{aligned}\langle\Psi(\mathbf{R},\boldsymbol{\sigma})|\Psi(\mathbf{R},\boldsymbol{\sigma})\rangle &= \left\langle \frac{1}{\sqrt{N!}}\sum_{\gamma_1\in S_N}(-1)^{|\gamma_1|}\hat{P}_{\gamma_1}\psi_{a_1}\psi_{a_2}\cdots\psi_{a_N}\right| \\ &\qquad \left.\frac{1}{\sqrt{N!}}\sum_{\gamma_2\in S_N}(-1)^{|\gamma_2|}\hat{P}_{\gamma_2}\psi_{b_1}\psi_{b_2}\cdots\psi_{b_N}\right\rangle \\ &= \left(\frac{1}{\sqrt{N!}}\right)^2\left\langle \sum_{\gamma_1\in S_N}(-1)^{|\gamma_1|}\psi_{\gamma_1(a_1)}\psi_{\gamma_1(a_2)}\cdots\psi_{\gamma_1(a_N)}\right| \\ &\qquad \left.\sum_{\gamma_2\in S_N}(-1)^{|\gamma_2|}\psi_{\gamma_2(b_1)}\psi_{\gamma_2(b_2)}\cdots\psi_{\gamma_2(b_N)}\right\rangle \\ &= \frac{1}{N!}N! = 1, \end{aligned} \tag{2.49}$$

where we used that the only non-zero terms are the ones in which the delta functions of Eq. (2.48) all survive, meaning $\gamma_1 = \gamma_2$. There are exactly $N!$ possible such terms in which the two permutations coincide, and they all carry an overall $+1$ sign since $(-1)^{|\gamma_1|}(-1)^{|\gamma_2|} = 1$ for any $\gamma_1 = \gamma_2$.

2.6 Postulates of Quantum Mechanics

The axiomatic formulation of QM was set up by British physicist P. Dirac and Hungarian-American mathematician-physicist-computer scientist J. von Neumann in the 1930s [59]. The fundamental postulates can be expressed as follows [41]:

(i) A *quantum state* of an isolated system is described fully by a state vector of unit norm in some separable Hilbert space $\mathcal{H}$.

> In the notation due to Dirac, these vectors are represented as "kets," $|\psi\rangle$. The kets can be expanded in any basis $\{|i\rangle\}_i$ $|\psi\rangle = \sum_i c_i|i\rangle$, with $c_i \in \mathbb{C}$. For any ket, a corresponding "bra" vector, $\langle\psi|$ can be assigned in the *dual space* of $\mathcal{H}$ by an anti-linear mapping
>
> $$|\psi\rangle \to \langle\psi| = \sum_i c_i^*\langle i|. \tag{2.50}$$
>
> The (complex valued) inner product is a composition of a bra and a ket, $\langle\psi_1|\psi_2\rangle$.

[11] Using for the moment a permutation operator acting on the orbital indices as opposed to the prior permutations which acted on the electron indices.

(ii) Each *physical observable* of the system is associated with a hermitian operator on $\mathcal{H}$. The spectrum of each such operator spans the Hilbert space.

If the spectrum is discrete, the eigenfunctions of such an operator are orthogonal and can always be made orthonormal e.g. by a Gram-Schmidt process [42]. The eigenfunctions form a complete set, in the sense that the unit operator can be represented by

$$\sum_i |i\rangle\langle i| = \mathbb{1}. \tag{2.51}$$

(iii) The time evolution of a state vector $|\psi\rangle = |\psi(t)\rangle$ is governed by the time dependent Schrödinger equation,

$$\mathrm{i}\hbar\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle. \tag{2.52}$$

The Hamiltonian, $\hat{H}$, is a *linear, hermitian* operator on $\mathcal{H}$.

(iv) The *measurable* (physical) values associated with observables are defined by the eigenvalues.

With a quantum system in a state $|\psi\rangle$ before measurement, the probability of measuring the value a_n corresponding to the $|n\rangle$ eigenvector of the hermitian operator $\hat{A}$ is given by

$$\text{probability of } a_n = |\langle n|\psi\rangle|^2. \tag{2.53}$$

If the geometric multiplicity of a_n is > 1, the probability is the sum over all corresponding eigenvectors, $|n\rangle, |m\rangle, \ldots, |k\rangle$, with the same eigenvalue [58].

(v) An *ideal* measurement of $\hat{A}$ resulting in the value a_n projects the state vector onto the sub-space of $\mathcal{H}$ spanned by all eigenvectors of $\hat{A}$ with eigenvalue a_n.

If a_n has geometric multiplicity unity, the resulting state *is* the corresponding eigenfunction,

$$|\psi\rangle \xrightarrow{\text{measurement}} |\psi'\rangle = \hat{P}_n|\psi\rangle = |n\rangle, \tag{2.54}$$

where $\hat{P}_n$ is the projection on the eigenstate $|n\rangle$. This is often referred to as "collapse of the wave function."

2.7 The variational principle

For a given Hamiltonian, calculating the expectation value of $\langle\Psi|\hat{H}|\Psi\rangle$ (for *any* $\Psi \in \mathcal{L}_2$) gives an upper bound on the ground state energy. Following Griffiths, a simple proof goes as follows: Since the spectrum of $\hat{H}$ spans[12] all of $\mathcal{L}_2$. We assume also that the eigenstates are orthonormalized. This means that *any* normalized Ψ we choose can be expressed in terms of the eigenstates of $\hat{H}$,

$$\Psi = \sum_{n=0}^{\infty} c_n \Phi_n, \quad \text{where} \quad \hat{H}\Phi_n = E_n \Phi_n. \tag{2.55}$$

The ground state energy is by definition the lowest eigenvalue of $\hat{H}$, we have $E_0 \leq E_n$ for all $n = 1, 2, \ldots$

Calculating the expectation value in terms of the E_ns, we find

$$\begin{aligned}
\big\langle \Psi|\hat{H}|\Psi\big\rangle &= \left\langle \sum_{n=0}^{\infty} c_n^* \Phi_n \middle| \hat{H} \middle| \sum_{n'=0}^{\infty} c_{n'} \Phi_{n'} \right\rangle \\
&= \sum_{n=0}^{\infty}\sum_{n'=0}^{\infty} c_n^* c_{n'} \big\langle \Phi_n|\hat{H}|\Phi_{n'}\big\rangle = \sum_{n=0}^{\infty}\sum_{n'=0}^{\infty} c_n^* c_{n'} E_{n'} \underbrace{\big\langle \Phi_n|\Phi_{n'}\big\rangle}_{\delta_{nn'}} \\
&= \sum_{n=0}^{\infty} c_n^* c_n E_n = \sum_{n=0}^{\infty} |c_n|^2 E_n.
\end{aligned} \tag{2.56}$$

Since Φ is assumed to be normalized, we know that $\sum_{n=0}^{\infty} |c_n|^2 = 1$, meaning $|c_n| \leq 1$ for any $n = 0, 1, 2, \ldots$ This means that the expectation value $\langle\Psi|\hat{H}|\Psi\rangle = |c_0|^2 E_0 + |c_1|^2 E_1 + \cdots \geq E_0$.

The variational principle is an invaluable tool, and forms the basis for almost all electronic structure methods.

The variational principle can be stated concisely as

$$E_0 = \min_{\Psi \in \mathcal{L}^2} \big\langle \Psi|\hat{H}|\Psi\big\rangle, \tag{2.57}$$

where we can consider the energy to be a functional of the wave function, $E[\Psi]$. More generally, the variational principle can be expressed as: The energy functional is stationary at all eigenvalues of the Hamiltonian, [60, 61]

$$\delta\, E[\Psi]\Big|_{\Phi_n} = 0. \tag{2.58}$$

Explicitly calculating the variation gives simply the time independent Schrödinger equation as the formal condition for $\delta E[\Phi]$ to vanish exactly [24]. Thus the general variational principle and the Schrödinger equation are in a sense two sides of the same coin.

[12] Asumming *crucially* that the Hamiltonian is hermitian [46]. This is not the case for e.g. the similarity transformed $\bar{H}$ used in coupled cluster theory, predictably leading to all kinds of difficulties.

Chapter 3

Wave functions

In ordinary quantum mechanics, the wave function is the primary quantity of interest. It constitutes the *solution* of the Schrödinger equation and encodes within it all information about the state of the isolated quantum system in question. Mathematically speaking, the wave function is the complex valued spatial projection of the abstract state vector which is a unitary vector in some separable Hilbert space [14, 59]. Formally, it is the solution to the Schrödinger equation, $\hat{H}\psi_k = E_k\psi_k$, and it has a probabilistic interpretation originally after German physicist M. Born, which states that the magnitude squared is a probability density, i.e. [55, 62, 63]

$$\mathrm{d}P(\mathbf{r}) = |\psi(\mathbf{r})|^2 \, \mathrm{d}^3\mathbf{r}. \tag{3.1}$$

The probability $\mathrm{d}P(\mathbf{r})$ denotes here the probability of finding the particle described by the wave function in an infinitesimal volume $\mathrm{d}^3\mathbf{r}$ around the position $\mathbf{r}$.

Even though we are unable (in the overwhelming majority of cases) to find closed form solutions to the Schrödinger equation, we may nevertheless write down a set of conditions we know the exact solution must adhere to. In the following section, we will go through properties of the exact wave function which are most relevant for atomic and molecular systems. Thereafter, we will consider the most common bases used to form approximate wave functions for many-body quantum systems.

3.1 Properties of the exact wave function

In the words of Helgaker and co-workers: Even though we are forced to make approximations in the solution of the Schrödinger equation, such... [13]

> “ Approximations should not be made in a haphazard manner. Rather we should seek to retain in our wave function as many symmetries and properties of the exact solution as possible. Indeed, some of the characteristics of the exact wave function are so important that we should try to incor-

> porate them at each level of theory, and a few are so fundamental that they are introduced into our models without thought. ”
>
> *T. Helgaker, P. Jørgensen, and J. Olsen*

Some of the most fundamental properties involve anti-symmetry and square integrability. These are normally relatively easily accounted for. However, there are more subtle ones which may be easily missed without a thorough analysis of the known properties of the exact wave function. We present here a(n incomplete) list of properties and conditions for the exact solution of the Schrödinger equation.

Eigenfunction of the number operator, $\hat{N}$

The exact wave function is a function of the spatial and spin degrees of freedom of the particles it describes. For an atomic or molecular system, under the Born-Oppenheimer approximation, the wave function depends only parametrically on the positions of the nuclei, $\Psi = \Psi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N, \sigma_1, \sigma_2, \ldots, \sigma_N; \mathbf{r}_A, \mathbf{r}_B, \ldots, \mathbf{r}_C)$. The approximation we choose should thus be an eigenfunction of the number operator, $\hat{N}|\psi\rangle = N|\psi\rangle$. The number operator is defined under second quantization as $\hat{N} = \sum_q c_q^\dagger c_q$, where the sum is taken to run over all possible single particle states [14].

Totally anti-symmetric under exchange of particles

A fermionic wave function must be totally anti-symmetric w.r.t. exhange of two particles [46]. In accordance with this, we must choose the approximating wave function to be an eigenfunction of the permutation operator, $\hat{P}_{ij}$, which interchanges particles i and j. We must also demand that the eigenvalue is -1, i.e. $\hat{P}_{ij}|\psi\rangle = -1|\psi\rangle$.

Both of the aforementioned conditions are satisfied if we take the approximation to be an $N \times N$ (or a linear combination of) Slater determinant(s) filled with single electron orbitals.

Square integrability and normalization

For a bound state, the exact wave function is square integrable and normalized to unity, $\langle\Psi|\Psi\rangle = 1$ [13]. This means that the exact wave function is finite almost everywhere[1] w.r.t. the $\mathcal{L}^2$ norm. A sufficient and natural way to ensure this holds for the approximating wave function is to build it from finite single-electron orbitals, i.e. populate the Slater determinant(s) with spin-orbitals which are themselves parts of $\mathcal{L}^2$.

[1] Mathematically, it is finite except possibly on a set of measure zero such that the integral over space is not affected by the divergent value.

Size-extensivity

The exact wave function is size-extensive in that a system of non-interacting subsystems have the same total energy as the sum of the energies of the subsystems [13]. It is thus reasonable to demand of the the approximate wave function that (in some chosen calculational scheme) the energy found by calculating the total energy of a system of non-interacting subsystems to exactly coincide with the energies of the subsystems themselves. In practice, we may check such a condition by separating subsystems by a large distance and comparing the calculated energy with the energy resulting from calculating the energies of the subsystems individually.

Eigenfunction of the total spin and spin projection operators, $\hat{S}^2$ and $\hat{S}_z$

In non-relativistic theory, the exact wave function is an eigenfunction of the total spin operator $\hat{S}^2$ and the spin projection operator $\hat{S}_z$ [64]. It would be natural to demand that the approximating wave function also be an eigenfunction of these two operators. Taking the approximation to be a single Slater determinant, populated with spin-orbitals of definite spin projection automatically means the total wave function is an eigenfunction of $\hat{S}_z$. However, single determinants are not *necessarily* eigenfunctions of $\hat{S}^2$ [22]. Linear combinations of determinants may be formed which by construction are eigenfunctions of $\hat{S}^2$ [13]. Such wave functions are called spin-adapted.

Asymptotic behaviour of the electronic density

Katriel and Davidson [65] showed that the electron density decays exponentially as

$$\rho(\mathbf{r}) \approx \exp\left(-2\sqrt{2I}r\right), \tag{3.2}$$

in the limit of large r. Here I denotes the first ionization potential of the molecule, i.e. the energy needed in order to remove the least tightly bound electron. Since the ionization potential is not known before the solution to the Schrödinger equation is found, an a priori treatment of the long-range exponential decay of the density is impossible [13].

Virial theorem

The exact wave function obeys the *virial theorem*, which states that (for a Coulombic potential $\hat{V}$) [66]

$$\langle \hat{T} \rangle = -\frac{1}{2}\langle \hat{V} \rangle. \tag{3.3}$$

A simple proof[2] by dimensional analysis due to Weinberg for the one-particle case illustrates the condition: Since the square of the wave function has to integrate over

[2]More precisely, a heuristic (not entirely rigorous) justification.

space to a probability, it must have dimensions of Length$^{-3/2}$ [55]. Letting a denote the chosen length scale, we can express the wave function as $\psi(\mathbf{r}) = a^{-3/2} f(\mathbf{r}/a)$, with $\mathbf{z} \equiv \mathbf{r}/a$ and $f(\mathbf{z})$ being a dimensionless function of a dimensionless argument. Changing integration variables in the expressions for $\langle \hat{V} \rangle$ and $\langle \hat{T} \rangle$,

$$\langle \psi | \hat{V} | \psi \rangle = \langle \hat{V} \rangle_\psi = \frac{\int \mathrm{d}^3\mathbf{r}\, V(\mathbf{r}) |\psi(\mathbf{r})|^2}{\int \mathrm{d}^3\mathbf{r}\, |\psi(\mathbf{r})|^2}, \quad \text{and} \tag{3.4}$$

$$\langle \psi | \hat{T} | \psi \rangle = \langle \hat{T} \rangle_\psi = \frac{\int \mathrm{d}^3\mathbf{r}\, \frac{\hbar^2}{2m} \left(|\frac{\partial \psi}{\partial x}|^2 + |\frac{\partial \psi}{\partial y}|^2 + |\frac{\partial \psi}{\partial z}|^2 \right)}{\int \mathrm{d}^3\mathbf{r}\, |\psi(\mathbf{r})|^2}, \tag{3.5}$$

from $\mathbf{r} \to \mathbf{z} = \mathbf{r}/a$ gives a single factor of a^{-1} in the former and a^{-2} in the latter integral. For the denominators, the integration measure $\mathrm{d}^3\mathbf{r}$ carries dimensions of a^3 which exactly cancel the $(a^{-3/2})^2 = a^{-3}$ from the wave function squared (by neccessity, since the integral represents a probability). In the $\langle V \rangle$ integral, the same happens in the numerator, and we are left with only the Coulomb potential a^{-1} contribution. The numerator in the $\langle T \rangle$ integral has dimensions of a^{-2}, since each coordinate differentiation carries a single inverse a.

As the exact wave function is a variational minimum of the Hamiltonian, the expectation value of $\langle H \rangle_\psi$ (note carefully that this is true only when evaluated *at the exact wave function* [ground or excited states]) must be independent of variations in ψ. Namely, they must be independent of variations of a since $\psi(\mathbf{r}) = a^{-3/2} f(\mathbf{z})$ [55]. The derivative of $\langle \hat{T} \rangle_\psi + \langle \hat{V} \rangle_\psi$ taken at the exact wave function must vanish, giving Eq. (3.3).

The virial theorem generalizes to N particles in the same form as Eq. (3.3).

Cusp conditions

In general, when charged particles approach each other the Coulombic $1/r$ term of the interaction energy diverges. In order for the energy to remain finite, the wave function needs to obey very specific sets of conditions dictating the behaviour of the discontinous derivatives at the collision points. First described by Kato, such *cusp conditions* describe known properties of the quantum system and wave function at the divergent points of the inter-electron and electron-nucleus Coulomb potentials [67]. These two cases will be described in depth in sections 3.1.2 and 3.1.1.

3.1.1 Electron-nucleus cusp

We will now consider in some detail the issue of the cusp condition arising from the singular Coulombic potential at the position of point-like nuclei.

Let us consider a system of a single atom of charge $+Z$ with N bound electrons. It will be useful in the following to define the *local energy*, E_{local}, as a spatially dependent

measure of the "instantaneous" energy of a system. We take

$$E_{\text{local}}(\mathbf{R}) \equiv \frac{1}{\Psi(\mathbf{R})}\hat{H}\Psi(\mathbf{R}) \tag{3.6}$$

to be the local energy, and note that for any eigenfunction $\Phi(\mathbf{R})$ of $\hat{H}$, the local energy is constant for all configurations $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N, \sigma_1, \sigma_2, \ldots, \sigma_N\}$ [68]. This is simply a trivial result of the Schrödinger equation, since $\hat{H}\Phi(\mathbf{R}) = E\Phi(\mathbf{R})$ we find that

$$E_{\text{local}}(\mathbf{R}) = \frac{1}{\Phi(\mathbf{R})}\hat{H}\Phi(\mathbf{R}) = \frac{1}{\Phi(\mathbf{R})}E\Phi(\mathbf{R}) = E. \tag{3.7}$$

We cannot normally find an approximate wave function for which $E_{\text{local}}(\mathbf{R}) = E$ holds, but we should at least make sure that it is *well behaved*. There are certain critical electronic configurations for which the Coulombic potential diverges. Keeping the local energy finite at these points lead to what is known as cusp conditions on the wave function.

The first critical configuration we will consider is the class of electronic positions for which $|\mathbf{r}_i - \mathbf{r}_A| \to 0$ for some electron i and nucleus A. Since the electron-nucleus Coulomb potential diverges there must be a corresponding divergent term in the laplacian which exactly cancels it. Let us consider the Born-Oppenheimer Hamiltonian for a single electron in the presence of a charge-Z atom,

$$\hat{H}(\mathbf{r}) = -\frac{\nabla^2}{2} - \frac{Z}{|\mathbf{r}|}, \tag{3.8}$$

where we take the atom to be situated at the origin. The *radial* Schrödinger equation can be written as [57]

$$\left[\frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} + \frac{2Z}{r} - \frac{l(l+1)}{r^2} + 2E\right] R(r) = 0. \tag{3.9}$$

For $l = 0$ states, we note that the two $1/r$ terms must exactly cancel if the local energy is to remain finite when $r \to 0$. This means that

$$E_{\text{local}}(\mathbf{r} \to 0) = \lim_{r\to 0}\left\{\frac{1}{R(r)}\left(\frac{2}{r}\frac{\partial}{\partial r} + \frac{2Z}{r} + \text{ finite terms}\right)R(r)\right\}, \tag{3.10}$$

and the exact wave function obeys [17]

$$\lim_{r\to 0}\left\{\frac{1}{R(r)}\frac{\partial R}{\partial r}\right\} = -Z. \tag{3.11}$$

We see that a wave function of s-type symmetry which does not vanish at $r = 0$ must be exponential in r in the limit of $r \to 0$. What happens if $l \neq 0$ or $R(0) = 0$?

It turns out that considering the latter issue automatically resolves the first, so let us take the case of $R(0) = 0$ [17]. We may factor the leading r dependence out of $R(r)$, and define $\tilde{R}(r) \equiv r^m R(r)$, so that $\tilde{R}(0) \neq 0$. Three applications of the derivative product rule gives

$$\frac{\partial}{\partial r} R(r) = \frac{\partial}{\partial r}\left[\tilde{R}(r) r^m\right] = \frac{\partial \tilde{R}}{\partial r} r^m + m\tilde{R}(r) r^{m-1}, \tag{3.12}$$

and

$$\frac{\partial^2}{\partial r^2} R(r) = \frac{\partial^2}{\partial r^2}\left[\tilde{R}(r) r^m\right] = \frac{\partial^2 \tilde{R}}{\partial r^2} r^m + 2\frac{\partial \tilde{R}}{\partial r} m r^{m-1} + m(m-1)\tilde{R}(r) r^{m-2}. \tag{3.13}$$

Insertion into the radial Schrödinger equation, Eq. (3.7), we find that

$$\begin{aligned} \frac{\partial^2 \tilde{R}}{\partial r^2} r^m + \frac{\partial \tilde{R}}{\partial r} \frac{2(m+1)}{r} r^m + \tilde{R}(r) \frac{m(m+1)}{r^2} r^m + \\ \tilde{R}(r) \frac{2Z}{r} r^m - \tilde{R}(r) \frac{l(l+1)}{r^2} r^m + 2E\tilde{R}(r) r^m = 0. \end{aligned} \tag{3.14}$$

If the local energy is to remain finite once again, we need the inverse powers of r to cancel, which for $1/r^2$ yields $m = l$. Furthermore, for the $1/r$ terms, we have the condition

$$\lim_{r \to 0} \left\{ \frac{1}{\tilde{R}(r)} \frac{\partial \tilde{R}}{\partial r} \right\} = -\frac{Z}{l+1}. \tag{3.15}$$

3.1.2 Electron-electron cusp

In the limit of two colliding electrons $r_{12} \to 0$, another cusp condition is found. It turns out that the corresponding radial equation for the inter-electronic separation carries the same dependence on r_{12} as the in the electron-nucleus case. With the only difference being the $-Z/r$ potential being replaced by a repulsive $1/r_{12}$ and kinetic term being twice as large [57].

We can write down the divergent parts of the local energy as

$$\begin{aligned} E_{\text{local}}(\mathbf{r}_{12} \to 0) = \lim_{r_{12} \to 0} \left\{ \frac{1}{R(r_{12})} \left(\frac{2}{r_{12}} \frac{\partial}{\partial r_{12}} - \frac{2}{r_{12}} - \right. \right. \\ \left. \left. \frac{l(l+1)}{r_{12}^2} + \text{ finite terms} \right) R(r_{12}) \right\}, \end{aligned} \tag{3.16}$$

where $l = 1$ if the spin-projections of electrons 1 and 2 are equal, and $l = 0$ otherwise [68]. A derivation analogue to the previous one reveals a corresponding cusp condition:

$$\lim_{r_{12} \to 0} \left\{ \frac{\partial R}{\partial r_{12}} \right\} = -\frac{R(r_{12})}{2(l+1)}. \tag{3.17}$$

This can be satisfied by a term in the wave function proportional to

$$R(r_{12}) \propto \begin{cases} \exp\left(\frac{r_{12}}{2}\right) & \text{if} \quad \sigma_i = \sigma_j \\ \\ \exp\left(\frac{r_{12}}{4}\right) & \text{if} \quad \sigma_i \neq \sigma_j \end{cases}. \tag{3.18}$$

3.1.3 Higher order coalescence conditions

In a system of $N+M$ charged particles, N electrons and M nuclei there will in general be a lot of such cusp conditions or *coalescence points*, where two or more electrons or nuclei coalesce with each other. Assuming all nuclei have the same charge Z, and disregarding nucleus-nucleus coalescence (recall that the Born-Oppenheimer wave function depends only parametrically on the positions of the nuclei), leaves us with: $r_{ij} \to 0$, $r_{ia} \to 0$, $r_{ij} \to 0$ and simulaneously $r_{ik} \to$, $r_{ia} \to 0$ and simulaneously $r_{ak} \to$, etc. The $i, j, k, \ldots$ indices here denote electronic coordinates, while $a, b, c, \ldots$ denote nucleonic coordinates.

We will not consider higher order conditions in the present work, but refer the reader to e.g. [17, 69].

3.2 Jastrow factor

Multiple functional forms which account explicitly for the electron-electron cusp condition described in section 3.1.2 are used in the literature. Some examples include the Boys-Handy function, the double exponential, or the Gaussian geminal form [17]. However, the most commonly used form is the Jastrow factor, sometimes called the Padé-Jastrow factor. Although originally proposed by Bijl in 1940, the form is commonly attributed to American physicist R. Jastrow [19, 70, 71].

The two-body Jastrow factor used in the current work has the form

$$J(\mathbf{R}) = \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{a_{ij}r_{ij}}{1+\beta r_{ij}}\right], \tag{3.19}$$

where $\beta > 0$ is a tunable parameter and a depends on the relative spin-projections of electrons i and j as [68]

$$a_{ij} = \begin{cases} 1/4 & \text{if} \quad \sigma_i = \sigma_j \\ 1/2 & \text{if} \quad \sigma_i \neq \sigma_j \end{cases}. \tag{3.20}$$

In general, it is possible to add higher order polynomials terms to the exponent resulting in [17]

$$J(\mathbf{R}) = \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{a_1 r_{ij} + a_2 r_{ij}^2 + \ldots}{1+\beta_1 r_{ij} + \beta_2 r_{ij}^2 + \ldots}\right]. \tag{3.21}$$

For optimized β_k values this may yield a more precise approximation to the true wave function, but it comes at the cost of more difficult computation and parameter optimization.

3.3 Orbitals

As noted in section 2.5.1, the Slater determinant—populated with eigenfunctions of the one-electron operator $\hat{h}$—is an exact solution for a non-interacting N-electron problem. But Slater determinants are also used as wave function ansatzes for interacting systems. The question of which functions should occupy the determinants however is a fairly non-trivial one. It turns out—somewhat surprisingly—that the chief concern is that of computational efficiency.

Before diving in, we detail briefly the *spherical harmonics*.

3.3.1 Spherical and solid harmonics

The spherical harmonics are a set of functions defined on the surface of a sphere. They are complete in the sense that they span the space of complex-valued continous functions on the unit sphere, $\text{span}\{Y_l^m(\theta,\phi)\} = C(\mathbb{S})$, and the space of complex-valued square integrable functions on $\mathbb{S}$, $\text{span}\{Y_l^m(\theta,\phi)\} = L^2(\mathbb{S})$ [72]. The surface of the unit sphere is here denoted by $\mathbb{S} = \{\mathbf{r} = (x,y,z) : |\mathbf{r}|^2 = 1\}$. The spherical harmonics thus naturally arise as an expansion basis for functions defined on the sphere.

Even more importantly, the spherical harmonics are eigenfunctions of the both the total angular momentum operator, $\hat{\mathbf{L}}^2$, and the z-projection of the angular momentum, $\hat{L}_z$ [43]. For QM problems involving *central potentials*—i.e. $V_{\text{ext}} = V_{\text{ext}}(r)$—the spherical harmonics are solutions to the angular part of the time independent Schrödinger equation (arising from separation of variables).

Formally, the spherical harmonics are functions of θ and ϕ: $Y_l^m(\theta,\phi)$ proprotional to $P_l^{|m|}(\theta)\mathrm{e}^{\mathrm{i}m\phi}$ (up to a normalization constant), where $P_l^{|m|}$ satisfies

$$-\frac{1}{\sin\theta}\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\sin\theta\frac{\mathrm{d}P_l^{|m|}}{\mathrm{d}\theta}\right) + \frac{m^2}{\sin^2\theta}P_l^{|m|} = l(l+1)P_l^{|m|}. \tag{3.22}$$

The parameters l and m are both integers, with $l \geq 0$ and $-l \geq m \geq l$. The polynomials P_l^m are known as the associated Legendre functions. Explicit expressions for the normalized spherical harmonics can be written out as

$$Y_l^m(\theta,\phi) = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}}\mathrm{e}^{\mathrm{i}m\phi}P_l^m(\cos\theta). \tag{3.23}$$

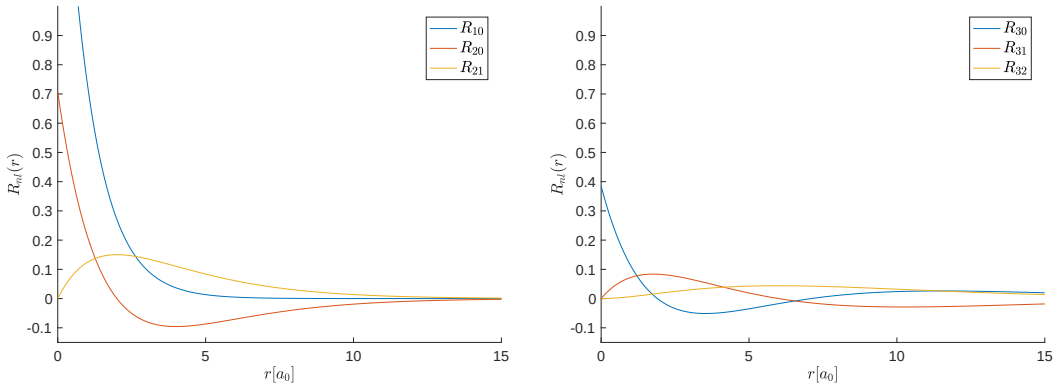

Figure 3.1: The first six hydrogenic orbitals for $Z = 1$, $R_{nl}(r)$. The general expression for $R_{nl}(r)$ is given in Eq. (3.25). Explicit expressions for these orbitals are given in Table 3.2.

The combination $r^l Y_l^m$ is a homogenous polynomial[3] of order l in the cartesian unit vector [55, 72]. The combinations $r^l Y_l^m$ which are also *harmonic* (i.e. solve the Laplace equation, $\nabla^2 r^l Y_l^m = 0$) are called the *solid harmonics* (up to a normalization), and can be made real by taking linear combinations of $\pm m$ terms. The first few real solid harmonics, $S_l^m(x, y, z)$ are shown in Table 3.1.

The real solid harmonics are often a more convenient form to work with, simply because they are real and defined in terms of cartesian coordinates. Please note that the solid harmonics span (the real-valued part of) $L^2(\mathbb{S})$ in the same way the spherical harmonics do, so we can do this without any loss of generality or applicability.

3.3.2 Hydrogenic orbitals

The non-interacting hydrogen-like Hamiltonian

$$\hat{H} = -\frac{\nabla^2}{2} - \frac{Z}{|\mathbf{r} - \mathbf{r}_A|}, \tag{3.24}$$

[3] A homogenous polynomial is a polynomial in which all terms have the same degree, i.e. $x^2 + xy + y^2$ or $x^3 + y^3 + z^3 + x^2 z$.

Table 3.1: Examples of the first few *normalized real solid harmonics*, $S_l^m(x,y,z) = r^l Y_l^m(\theta,\phi)$. We note that the solid harmonics of order l are simply linearly independent homogenous polynomials in x, y, and z, of order l.

Real solid harmonic	**Azimuthal, magnetic quantum numbers**, (l,m)	**Expression**
S_0^0	0, 0	1
S_1^{-1}	1, -1	y
S_1^0	1, 0	z
S_1^1	1, 1	x
S_2^{-2}	2, -2	$\sqrt{3}xy$
S_2^1	2, 1	$\sqrt{3}yz$
S_2^0	2, 0	$-\frac{1}{2}(x^2+y^2)+z^2$
S_2^{-1}	2, 1	$\sqrt{3}xz$
S_2^{-2}	2, 2	$\frac{\sqrt{3}}{2}(x-y)(x+y)$
S_3^{-3}	3, -3	$-\frac{1}{2}\sqrt{\frac{5}{2}}(-3x^2+y^2)y$
S_3^2	3, -2	$\sqrt{15}xyz$
S_3^1	3, -1	$-\frac{1}{2}\sqrt{\frac{3}{2}}(x^2+y^2-4z^2)y$
S_3^0	3, 0	$-\frac{3}{2}(x^2+y^2)z+z^3$
S_3^{-1}	3, 1	$-\frac{1}{2}\sqrt{\frac{3}{2}}(x^2+y^2-4z^2)x$
S_3^{-2}	3, 2	$\frac{1}{2}\sqrt{15}(x-y)(x+y)z$
S_3^{-3}	3, 3	$\frac{1}{2}\frac{\sqrt{5}}{2}(x^2-3y^2)x$

with $\mathbf{r}_A = \mathbf{0}$ being the position of a single nucleus of charge $+Z$, has normalized *radial* eigenfunctions

$$R_{nl}(r) = \sqrt{\left(\frac{2Z}{n}\right)^3 \frac{(n-l-1)!}{2n[(n+l)!]^3}} \exp\left[-\frac{Zr}{n}\right] \left(\frac{2Zr}{n}\right)^l L_{n-l-1}^{2l+1}\left(\frac{2Zr}{n}\right). \quad (3.25)$$

The L_n^α here denotes the (generalized) Laguerre polynomials[4] In order to produce the full spin-orbitals, we need to append the spherical harmonic $Y_l^m(\theta, \phi)$ for appropriate quantum numbers l (the azimuthal quantum number) and m (the magnetic quantum number) in addition to a spin function, $\chi(\sigma)$. The corresponding eigenvalues depend famously only on the principal quantum number n, as [46]

$$E = -\frac{Z^2}{2n^2}. \quad (3.28)$$

The first six radial orbitals are shown in Fig. 3.1, with explicit expressions for the first ten being shown in Table 3.2. Fig. 3.2 shows a few examples of the full orbitals, i.e. the radial functions multiplied by spherical harmonics.

[4]The (generalized) Laguerre polynomials are the solutions to the differential equation

$$x\frac{\mathrm{d}^2 y(x)}{\mathrm{d}x^2} + (1+\alpha-x)\frac{\mathrm{d}y(x)}{\mathrm{d}x} + ny(x) = 0, \quad (3.26)$$

with n a non-negative integer and α an arbitrary real constant [73]. An explicit expression for the polynomials themselves can be found by the so-called Rodrigues formula:

$$L_n^\alpha(x) = x^{-\alpha}\frac{1}{n!}\left(\frac{\mathrm{d}}{\mathrm{d}x} - 1\right)^n x^{n+\alpha}. \quad (3.27)$$

Table 3.2: Explicit analytical expressions for the first few hydrogenic radial wave functions, $R_{nl}(r)$ [46]. The first six—R_{10} to R_{32}—are shown for $Z = 1$ in Fig. 3.1.

Radial orbital	**Principal, azimuthal quantum numbers**, (n, l)	**Expression**
R_{10}	1, 0	$2\sqrt{Z^3}\mathrm{e}^{-Zr}$
R_{20}	2, 0	$\sqrt{\frac{Z^3}{2}}\left(1-\frac{Zr}{2}\right)\mathrm{e}^{-Zr/2}$
R_{21}	2, 1	$\sqrt{\frac{Z^5}{24}}r\mathrm{e}^{-Zr/2}$
R_{30}	3, 0	$\frac{2\sqrt{Z^3}}{\sqrt{27}}\left(1-\frac{2Zr}{3}+\frac{2Z^2r^2}{27}\right)\mathrm{e}^{-Zr/3}$
R_{31}	3, 1	$\frac{8\sqrt{Z^5}}{27\sqrt{6}}\left(1-\frac{Zr}{6}\right)r\mathrm{e}^{-Zr/3}$
R_{32}	3, 2	$\frac{4\sqrt{Z^7}}{81\sqrt{30}}r^2\mathrm{e}^{-Zr/3}$
R_{40}	4, 0	$\frac{\sqrt{Z^3}}{4}\left(1-\frac{3Zr}{4}+\frac{Z^2r^2}{8}-\frac{Z^3r^3}{192}\right)\mathrm{e}^{-Zr/4}$
R_{41}	4, 1	$\frac{\sqrt{5Z^5}}{16\sqrt{3}}\left(1-\frac{Zr}{4}+\frac{Z^2r^2}{80}\right)r\mathrm{e}^{-Zr/4}$
R_{42}	4, 2	$\frac{\sqrt{Z^7}}{64\sqrt{5}}\left(1-\frac{Zr}{12}\right)r^2\mathrm{e}^{-Zr/4}$
R_{43}	4, 3	$\frac{\sqrt{Z^9}}{768\sqrt{35}}r^3\mathrm{e}^{-Zr/4}$

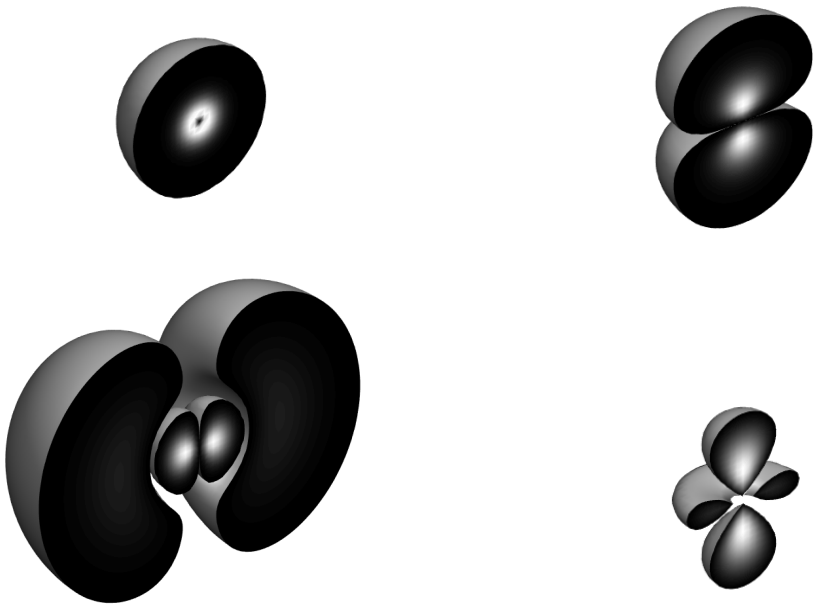

Figure 3.2: Examples of full hydrogenic orbitals, $\psi_{nlm}(r, \theta, \phi) = R_{nl}(r)Y_l^m(\theta, \phi)$: ψ_{100} (top left), ψ_{210} (top right), ψ_{311} (bottom left), and ψ_{420} (bottom right). The relative scaling is not accurate. Each plot is sliced in the x-y-plane and a colormap showing the density is inset. The outer contour shows the isosurface of each orbital at $|\psi_{nlm}|^2 = 10^{-5}$.

In many ways, the hydrogenic orbitals appear to be *natural* orbitals to work with. They obey—crucially—the electron-nucleus cusp condition of section 3.1.1. They also fall off exponentially as per the correct long range limit. However—as noted by Helgaker and co-workers—because of some key deficiencies, they turn out to not be very useful as basis functions [13]. Firstly, they do not span the entire one-electron Hilbert space. In order for *completeness* to be achieved, the unbounded positive energy continuum states need to be appended to the set. Secondly, they spread out and become very *diffuse* very qucikly for increasing n because of the inverse term in the exponential. This means a (very) large number of terms need to be considered in order to obtain a flexible description of the core regions of the many-body wave function.

In addition to this, integral evaluation with the hydrogenic orbitals turns out to be unfeasibly slow compared to more efficient basis sets.

3.3.3 Slater type orbitals

It is possible to create a complete set of hydrogen-like orbitals by considering instead of the $2Z/n$ factor, a constant exponential term $\mathrm{e}^{-\zeta r}$, with $\zeta \in \mathbb{R}$. These orbitals are sometimes known as the Laguerre fuctions. However, the single constant exponent means it becomes exceedingly difficult to approximate orbitals of widely different

nature. With a chosen large exponent, the compact core orbitals may be well approximated. But convergence for the more wide-spread diffuse valence orbitals will be horribly slow, and vice versa.

A relative of the hydrogenic orbitals which use *variable* exponents are the Slater type orbitals (STO). Introduced by American physicist J. C. Slater in 1930, they have the same exponential decay but forego the nodal structure of the hydrogenic orbitals [74]. Building on the previous work of American physicist C. M. Zener—who had noted that the radial nodes of the generalized hydrogenic orbitals in general had little impact on the Hatree-Fock-like integrals used to construct variational wave functions—Slater proposed a much simpler polynomial radial structure [75]. In addition to this, once variable exponents are introduced, the orthogonality of the Laguerre polynomials are lost. Since the orthononality alone is the reason for the complicated nodal structure, it does not make much sense to keep the Laguerre polynomials in the wave function once this is lost [13].

Slater instead proposed a much simpler polynomial structure: r^{n-1}. The general expression for the normalized STO with exponent ζ is given by [76]

$$R_n(r;\zeta) = \frac{(2\zeta)^{n+1/2}}{[(2n)!]^{1/2}}(2\zeta r)^{n-1}\mathrm{e}^{-\zeta r}. \tag{3.29}$$

These $R_n(r)$s are obviously radial functions only, and need to be paired with e.g. the spherical or solid harmonics in order to produce a full one-electron wave function.

The variable exponent STOs are complete under certain conditions on the ζ and the sequence of n-s used, see [77]. In short, the STOs remedy some of the weaknesses of the hydrogenic orbitals. However, it turns out that the most important property of a basis set is the ability to perform efficient integrations on them. Ultimately thus, we will end up working with a basis set intrinsicly *less suited* to the task (arguing from a physical interpretation point of view) because the neccessary integrals are possible to evaluate very quickly.

Please note that this picture may be set to change in the not so distant future, as a lot of work is being put into making integral evaluation of STOs more feasible, see e.g. [78, 79].

3.3.4 Gaussian type orbitals

The basis sets used in most modern electronic structure calculations are comprised of Gaussian distributions, $\exp(-\alpha r^2)$. This is also the basis sets we will chiefly employ in the present work. Although the radial decay is proportional to e^{-r^2} (qualitatively wrong, the correct [long range] behaviour is $\sim \mathrm{e}^{-r}$) the Gaussians nevertheless have the upper hand due to the ease with which many-center integrals in terms of them can be performed. The Gaussian basis functions were first introduced in the context of electronic structure calculations by Boys in 1950 as an alternative to STO-like functions [80].

A general (normalized) Cartesian Gaussian type orbital (GTO) is given as

$$g_{ijk}^{\alpha}(x,y,z) \equiv \left(\frac{2\alpha}{\pi}\right)^{3/4} \left[\frac{(8\alpha)^{i+j+k}(i!j!k!)}{(2i)!(2j)!(2k)!}\right]^{1/2} x^i y^j z^k \mathrm{e}^{-\alpha(x^2+y^2+z^2)}, \tag{3.30}$$

where i, j, and k are non-negative integers and α determines the width according to the variance $\sigma^2 = 1/(2\alpha)$ [76]. We will denote these functions by **Gaussian primitives**. The coordinates x, y, and z are in general given w.r.t. a nucleus, i.e. $x = \tilde{x} - x_A$ for some nucleus index A. The global coordinate $\tilde{x}$ denotes a coordinate w.r.t. the global origin. We will largely ignore this, except when we explicitly need to include it in our calculations, in which case we will write out the Gaussian primitives as

$$g_{ijk}^{\alpha}(x,y,z;\mathbf{r}_A) = (x-x_A)^i (y-y_A)^j (z-z_A)^k \mathrm{e}^{-\alpha[(x-x_A)^2+(y-y_A)^2+(z-z_A)^2]}, \tag{3.31}$$

where we omitted the normalization for brevity.

With $i = j = k = 0$, the primitive has spherical symmetry and is a so-called s-type GTO. If i (j) [k] is one, while the other two indices vanish, the Gaussian has axial symmetry along the x (y) [z] direction. This is known as a p-type GTO. More generally, the sum $i + j + k \equiv l$ denotes the angular momentum of any GTO, and as usual we denote $l = 0$ as s, $l = 1$ as p, $l = 2$ as d, $l = 3$ as f, and so on.

Contracted Gaussian functions

In practical calculations, a number of these functions will be used for each atomic center. The Gaussians in general dont look much like the *true* molecular orbitals (which resemble hydrogenic wave functions), but this is remedied by taking linear combinations of GTOs for each orbital [13]. We will call the linear combinations **contracted Gaussians**,

$$G(\mathbf{r}) = \sum_{a=1}^{L} d_a g_{i_a j_a k_a}^{\alpha_a}(x,y,z). \tag{3.32}$$

A key fault of the Gaussians is that any Gaussian of s-type symmetry fails to satisfy the electron-nucleus cusp condition described in section 3.1.1. As the derivative at $r = 0$ of any s-type primitive vanishes, even linear combinations will not remedy this fault. However, we can get *close* in a sense by taking combinations of more and more primitives of varying exponents α. Even if at the limit of infinite primitives, the origin derivative still vanishes, we can still construct a combination such that the integrals involved in the self-consistent field (SCF) methods become arbitrarily close to their STO counterparts.

Since SCF theories are in practice solved via the weak integral formulation and application of the Galerkin method, the overall form of the orbitals are more important than their failure to accurately capture the physics at very small r [81].

Table 3.3: The number of Gaussian primitives of total angular momentum l, $(l+1)(l+2)/2$, and the number of linearly independent homogenous harmonic polynomials of order l, $2l+1$.

Angular momentum, l	Type	Total primitives	Linearly independent combinations
0	s	1	1
1	p	3	3
2	d	6	5
3	f	10	7
4	g	15	9
5	h	21	11
6	i	28	13
7	k	36	15
8	l	45	17

We note before going on that a contracted Gaussian is uniquely determined by an array of coefficients, $\mathbf{d} = \{d_a\}_a$, an array of exponents, $\boldsymbol{\alpha} = \{\alpha_a\}_a$, and the coefficients i, j, and k.

Number of Cartesian Gaussian primitives of angular momentum l

For p-type orbitals, the hydrogenic or Slater type orbitals—when paired with the appropriate $l = 1$ spherical or solid harmonics—are three-fold degenerate. The same is true of the Gaussian primitives. This means that any p-type orbital produces three distinct (contracted) Gaussian functions: one for (linear combinations of) $g^{\alpha}_{100}(\mathbf{r}) = x\mathrm{e}^{-\alpha r^2}$, one for (linear combinations of) $g^{\alpha}_{010} = y\mathrm{e}^{-\alpha r^2}$, and finally one for (linear combinations of) $g^{\alpha}_{001} = z\mathrm{e}^{-\alpha r^2}$. In terms of the *real solid harmonics*, this corresponds naturally to the Slater type orbitals

$$S_1^m(x,y,z)\psi(r) = \begin{cases} y\mathrm{e}^{-\zeta r} & \text{for} \quad m=-1 \\ z\mathrm{e}^{-\zeta r} & \text{for} \quad m=\ \ 0 \\ x\mathrm{e}^{-\zeta r} & \text{for} \quad m=+1 \end{cases}. \tag{3.33}$$

For higher angular momentum values, there are in general $(l+1)(l+2)/2$ possible combination[5] of i, j, and k which gives $i+j+k=l$. However, there are only $2l+1$ linearly independent spherical (or real solid) harmonics of degree l. Essentially, this is

[5]The problem is identical to the combinatorial problem: "Given N indistinguishable objects, how many possible ways can we distribute them into M distinguishable bins?" For three "bins" and l

just stating the fact that there exist $D_{\mathrm{H}}(d,v)$ linearly independent homogenous polynomials of degree d in v variables, but only $D_{\mathrm{HH}}(d,v)$ linearly independent *harmonic* homogenous polynomials, where [82]

$$D_{\mathrm{H}}(d,v) = \binom{v+d-1}{v-1}, \quad \text{while} \quad D_{\mathrm{HH}}(d,v) = \frac{2d+v-2}{d}\binom{d+v-3}{d-1}. \tag{3.35}$$

Examples of the dimensions of the spaces of homogenous (D_{H}) and homogenous harmonic (D_{HH}) polynomials are shown in Table 3.3, for degree l in 3 variables. Since the spherical (real solid) harmonics span the space of complex-valued (real-valued) spherical functions, using a basis that is larger than that of the harmonic homogenous polynomials is essentially a waste of computational effort [72]. In this sense the Cartesian Gaussians are *over-complete*, and it is possible to construct from the $(l+1)(l+2)/2$ Gaussians of degree l a more compact set of $2l+1$ linearly independent Gaussian combinations which are sufficient for the expansion of any function on $\mathbb{S}$ [13].

As an example, consider the $l=2$, d-type, Gaussian primitives. The three functions $g^{\alpha}_{110}(\mathbf{r}) = xy\mathrm{e}^{-\alpha r^2}$, $g^{\alpha}_{101}(\mathbf{r}) = xz\mathrm{e}^{-\alpha r^2}$, and $g^{\alpha}_{011}(\mathbf{r}) = yz\mathrm{e}^{-\alpha r^2}$ coincide with the corresponding *harmonic* set (as can be seen from the solid harmonics of Table 3.1). However, there are three more d-type Cartesian primitives, but only two additional spherical harmonics. Instead of the primitives $g^{\alpha}_{200}(\mathbf{r}) = x^2\mathrm{e}^{-\alpha r^2}$, $g^{\alpha}_{020}(\mathbf{r}) = y^2\mathrm{e}^{-\alpha r^2}$, and $g^{\alpha}_{002}(\mathbf{r}) = z^2\mathrm{e}^{-\alpha r^2}$, we can form the linear combinations

$$\begin{aligned} \bar{g}^{\alpha}_1(\mathbf{r}) &= g^{\alpha}_{200} - g^{\alpha}_{020} \\ &= \left(x^2 - y^2\right)\mathrm{e}^{-\alpha r^2} \qquad (3.36) \\ \bar{g}^{\alpha}_2(\mathbf{r}) &= 2g^{\alpha}_{002} - g^{\alpha}_{200} - g^{\alpha}_{020} \\ &= \left[3z^2 - \left(x^2+y^2+z^2\right)\right]\mathrm{e}^{-\alpha r^2}. \qquad (3.37) \end{aligned}$$

We note that these are exactly the same combinations as the solid harmonics of degree $l=2$ (up to a normalization, which we have omitted from the expressions of Eq. (3.36)-(3.37)). A last linear combination may be formed by taking $g^{\alpha}_{200}(\mathbf{r}) + g^{\alpha}_{020}(\mathbf{r}) + g^{\alpha}_{002}(\mathbf{r}) = (x^2+y^2+z^2)\mathrm{e}^{-\alpha r^2}$, but we note that this has spherical symmetry and thus is really an s-type orbital [76].

As can be seen by Table 3.3, this is especially important for higher angular momentum numbers $l>3$. As we primarily use f-type or lower angular momentum primitives in the present work, we choose to keep the larger sets of $D_{\mathrm{H}}(l,3)$ Gaussians.

"ojects," this is

$$\binom{N+M-1}{M-1} = \binom{l+3-1}{3-1} = \binom{l+2}{l} = \frac{(l+1)(l+2)}{2}. \tag{3.34}$$

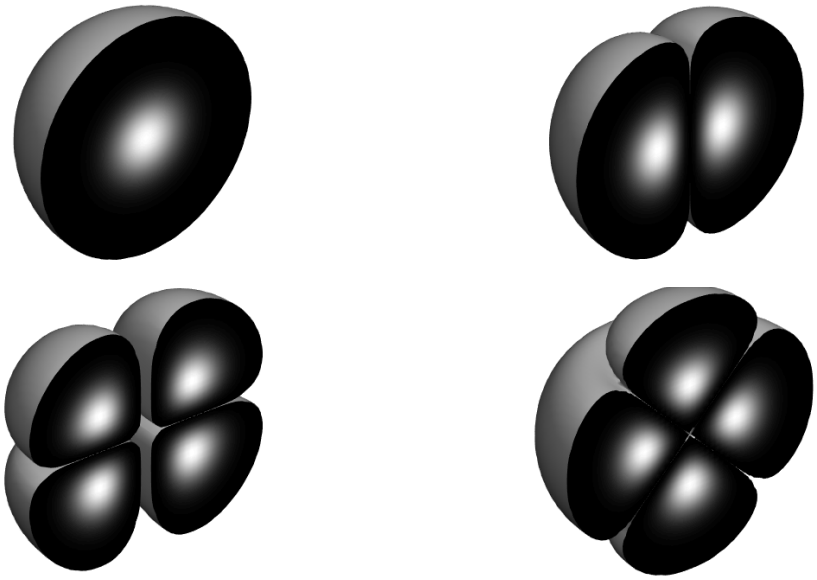

Figure 3.3: Examples of Gaussian orbitals with $\alpha = 1.0$, $G_{ijk}(x, y, z) = x^i y^j z^k \mathrm{e}^{-\alpha r^2}$: G_{000} (top left), G_{100} (top right), G_{201} (bottom left), and $G_{002} - G_{020} - G_{200}$ (bottom right). The latter combination is one of the five linearly independent set of d-type Gaussian orbitals (there are a total of six Gaussian primitives with $l = 2$—G_{200}, G_{020}, G_{002}, G_{110}, G_{101}, and G_{011}—but the set is linearly dependent: a linearly independent set may be formed by taking G_{110}, G_{101}, G_{011}, $G_{200} - G_{020}$, and $G_{002} - G_{020} - G_{200}$). Each plot is sliced in the x-y-plane and a colormap showing the density is inset. The outer contour shows the isosurface of each orbital at $|G_{ijk}|^2 = 10^{-5}$.

3.3.5 Some properties of Gaussians

Cartesian Gaussians

Before we go on, we will state some properties of the Gaussians which will be crucial for us in the implementation of the Hartree-Fock machinery in section 8. First of all, each Cartesian primitive factorizes in the Cartesian coordinates,

$$g^{\alpha}_{ijk}(\mathbf{r}) = g^{\alpha}_{i}(x)g^{\alpha}_{j}(y)g^{\alpha}_{k}(z), \tag{3.38}$$

where $g^{\alpha}_{i}(x) \equiv x^{i}\mathrm{e}^{-\alpha x^2}$ and similarly for y and z [83]. Please note that this is *not* true for either of the hydrogenic, Slater type, or spherical harmonic Gaussians[6]. The *Gaussian components* adhere to the the simple recurrence relation

$$xg^{\alpha}_{i}(x) = g^{\alpha}_{i+1}(x), \tag{3.39}$$

and from this it is immideately clear that differentiation w.r.t. x yields [84]

$$\begin{aligned}\frac{\partial g^{\alpha}_{i}(x;\mathbf{r}_A)}{\partial x_A} &= \frac{\partial}{\partial x_A}\left((x-x_A)^{i}\mathrm{e}^{-\alpha(x-x_A)^2}\right) = -\frac{\partial g^{\alpha}_{i}(x;\mathbf{r}_A)}{\partial x} \\ &= 2\alpha g^{\alpha}_{i+1}(x;\mathbf{r}_A) - ig^{\alpha}_{i-1}(x;\mathbf{r}_A).\end{aligned} \tag{3.40}$$

Note that we have briefly re-inserted the $\mathbf{r}_A$ back into the primitive at this point because of the explicit dependence on the nucleonic position.

Hermite Gaussians

The Cartesian Gaussians can be written in terms of **Hermite Gaussians**—products of exponentials and Hermite polynomials[7]—which are defined as follows: [85]

$$\Lambda^{\alpha}_{tuv}(\mathbf{r};\mathbf{P}) = \left(\frac{\partial}{\partial P_x}\right)^{t}\left(\frac{\partial}{\partial P_y}\right)^{u}\left(\frac{\partial}{\partial P_z}\right)^{v}\mathrm{e}^{-\alpha|\mathbf{r}-\mathbf{P}|^2}. \tag{3.43}$$

[6]The spherical harmonic Gaussian primitives account for the angular dependency in the wave function by appending to the Gaussian function spherical harmonics $Y^{m}_{l}(\theta,\phi)$.

[7]The Hermite polynomials are a family of orthogonal real-valued polynomials, solutions of the Hermite differential equation [82]

$$\frac{\partial^2 H_n(x)}{\partial x^2} - \frac{\mathrm{d}H_n(x)}{\mathrm{d}x} + nH_n(x) = 0. \tag{3.41}$$

An explicit formula for the n-th Hermite polynomial can be written down as [73]

$$H_n(x) = (-1)^{n}\mathrm{e}^{x^2}\frac{\mathrm{d}^{n}}{\mathrm{d}x^{n}}\mathrm{e}^{x^2}. \tag{3.42}$$

The $\mathbf{P} = (P_x, P_y, P_z)$ simply denotes *some* real-valued vector representing a point in space. It is clear that the Hermite Gaussians also factorize in Cartesian coordinates, with

$$\Lambda_t^\alpha(x; P_x) = \left(\frac{\partial}{\partial P_x}\right)^t \mathrm{e}^{-\alpha(x-P_x)^2}, \tag{3.44}$$

and similar for y and z coordinates, such that $\Lambda_{tuv}^\alpha(\mathbf{r}; \mathbf{P}) = \Lambda_t^\alpha(x; P_x)\Lambda_u^\alpha(y; P_y)\Lambda_v^\alpha(z; P_z)$. It is possible to use the Hermite Gaussians themselves as basis functions, but for our purposes we consider them only as intermediates involved in the neccessary integrals. Since they are defined in terms of derivatives they will—not surprisingly—lead to tremendous simplifications in that regard. Differentiation of the Hermite Gaussians naturally leads to a recurrence relation similar to Eq. (3.39),

$$\frac{\partial \Lambda_t^\alpha(x; P_x)}{\partial P_x} = -\frac{\partial \Lambda_t^\alpha(x; P_x)}{\partial x} = \Lambda_{t+1}^\alpha(x; P_x), \tag{3.45}$$

and it can be shown that left-multiplying by x_p yields [83]

$$x_p \Lambda_t^\alpha(x; P_x) = \frac{1}{2\alpha}\Lambda_{t+1}^\alpha(x; P_x) + t\Lambda_{t-1}^\alpha(x; P_x). \tag{3.46}$$

Let now $\mathbf{r}_p = \mathbf{r} - \mathbf{P}$ denote the vectorial difference w.r.t. $\mathbf{P}$, such that $\Lambda_t^\alpha(x_p; P_x) = (\partial/\partial P_x)^t \mathrm{e}^{-\alpha x_p^2}$ with $\mathbf{r}_p = (x_p, y_p, z_p)$. Now, we may consider $\Lambda_t^\alpha(x_p; P_x)$ as a product of two Hermite polynomials, $H_t(x_p)$ and $H_0(x_p) = 1$, multiplied by an exponential factor. From this consideration it follows trivially that the integral

$$\begin{aligned}\int_{-\infty}^{\infty} \mathrm{d}x\, \Lambda_t^\alpha(x_p; P_x) &= \int_{-\infty}^{\infty} \mathrm{d}x\, H_t(x_p) H_0(x_p) \mathrm{e}^{-\alpha x_p^2} \\ &\overset{t\neq 0}{=} 0,\end{aligned} \tag{3.47}$$

since the Hermite polynomials are orthogonal w.r.t. the weight function $w(x) = \mathrm{e}^{-x^2/2}$ [73]. The coordinate transformation $x_p \mapsto x_p/\sqrt{2\alpha}$ gives the correct scaling such that the integrand becomes $H_t(x_p)H_0(x_p)w(x_p)$ only changed by a constant pre-factor which does not change the conclusion of Eq. (3.47). In the case of $t = 0$, the integral is simply

$$\int_{-\infty}^{\infty} \mathrm{d}x\, \Lambda_0^\alpha(x_p; P_x) = \int_{-\infty}^{\infty} \mathrm{d}x\, \mathrm{e}^{-\alpha x_p^2} = \sqrt{\frac{\pi}{\alpha}}, \tag{3.48}$$

and in general

$$\int_{-\infty}^{\infty} \mathrm{d}x\, \Lambda_t^\alpha(x_p; P_x) = \delta_{t0}\sqrt{\frac{\pi}{\alpha}}. \tag{3.49}$$

Gaussian product rule

The property which turns out to be most crucial for our use is the fact that a product of Gaussians centered at different points yield again a Gaussian. This is known as the *Gaussian product rule* and can be stated as

$$\mathrm{e}^{-\alpha(x-A_x)^2}\mathrm{e}^{-\beta(x-A_x)^2} = \mathrm{e}^{-\mu x_{AB}^2}\mathrm{e}^{-px_P^2}, \tag{3.50}$$

where $p \equiv \alpha + \beta$, $x_{AB} \equiv A_x - B_x$, $\mu \equiv \alpha\beta/p$, and $\mathbf{P}$ denotes the "center of mass," [57]

$$\mathbf{P} \equiv \frac{\alpha\mathbf{A} + \beta\mathbf{B}}{\alpha + \beta} = \frac{\alpha\mathbf{A} + \beta\mathbf{B}}{p}. \tag{3.51}$$

The product of Cartesian Gaussian primitives is known as an **overlap distribution**, [84]

$$\Omega_{ij}(x) \equiv g_i^\alpha(x; A_x)g_j^\beta(x; B_x) = K_{AB}x_A^i x_B^j \mathrm{e}^{-px_P^2}. \tag{3.52}$$

These overlap distributions can then be expanded in terms of Hermite Gaussians, greatly simplifying the evaluation of so called *two-center integrals*. We will expand on this when the Hartree-Fock implementation is discussed in section 8.

3.4 Gaussian basis sets

STO-nG basis sets

One approach to constructing viable contracted orbital basis sets is to approximate Slater type orbitals. This may be done by applying a least-squares fit of L primitives to a given STO. Such contracted Gaussians are normally denoted **STO-nG**, with n being the number of primitives used. The STO-3G basis sets of Hehre and co-workers have long been considered an efficient comprimise between efficiency and accuracy—useful for running preliminary simulations before bringing out the proverbial big guns [76, 83, 86]. Examples of such fittings[8] can be seen in Fig. 3.5, where all of STO-1G through STO-7G are shown. Normally, the highest number of primitives used is six. From Fig. 3.4 we see that the mean absolute error compared to the Slater orbital, ε, scales approximately as

$$\varepsilon(\mathrm{n}) \sim \sqrt{\frac{1}{10^{\mathrm{n}}}}. \tag{3.53}$$

We note that the overall function shape of STO-3G already closely resembles the actual STO. At higher values of n, they become indistinguishable without extremely

[8] All curve fitting in the present work is done in MATLAB using the LAD (*least absolute deviations*, as opposed to the more familiar *least squared deviations*) approach and the trust-region algorithm proposed by Moré and co-workers [87–89].

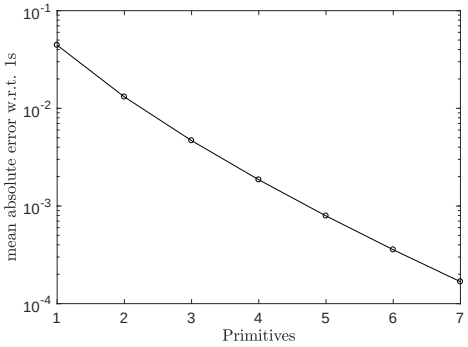

Figure 3.4: Example showing the average absolute error relative to the 1s STO for each of the STO-nG approximations with different number of primitives n shown in Fig. 3.5. The s-type STO has exponent $\zeta = \sqrt[3]{\pi} \approx 1.4646$, arbitrarily chosen simply to ensure $\mathrm{STO}(r) \to 1$ as $r \to 0$.

close inspection. From the insets of Fig. 3.5 we note that the sum of Gaussians always satisfy $\mathrm{d}/\mathrm{d}r \sum_k g_k^\alpha(r)\big|_{r=0} \neq 0$, so in the limit of very small r the STO-nG and the STO digress.

Split-valence basis sets

The electrons mostly involved in chemical bonding in any given atom are the ones occupying the highest principal quantum number (n) states. These are known as **valence electrons**, as opposed to the remaining **core electrons** [90]. Because the former are much more important to the chemical properties of atoms, it is common to provide multiple basis functions to represent the valence orbitals. Orbitals sets for which this is done are called split-valence.

Single and multiple-ζ basis sets

The STO-nG basis sets are known as *minimal* or *single-*ζ, in that for each electron there is only a single basis function. E.g. a minimal H basis only has a single orbital of s-type symmetry. It does not require much imagination to realize that this offers little in the way of flexibility w.r.t. representing atomic and molecular orbitals. In order to add such flexibiliy it is common to add more contracted Gaussians for each electron present.

One way to achieve this is to *de-contract* the STO-nG basis sets. For example, take the STO-2G basis and use each primitive of the contracted Gaussians as a contracted orbital in its own right [76]. This makes the single-ζ STO-2G basis set into a double-ζ basis, in which each atomic orbital is represented by two distinct contracted Gaussian functions.

A fundamentally different approach is taken by e.g. Pople and co-workers in their widely used family of basis sets [91]. Instead of using the Slater type orbitals as a starting point, the primitive exponents and contraction coefficients are instead optimized in variational manner by SCF iteration. The resulting sets are denoted C-$\mathrm{V}_1\mathrm{V}_2$G, where C represents the number of primitives used in the single contracted function representing each core electronic orbital. V_1 and V_2 denote the number of primitives

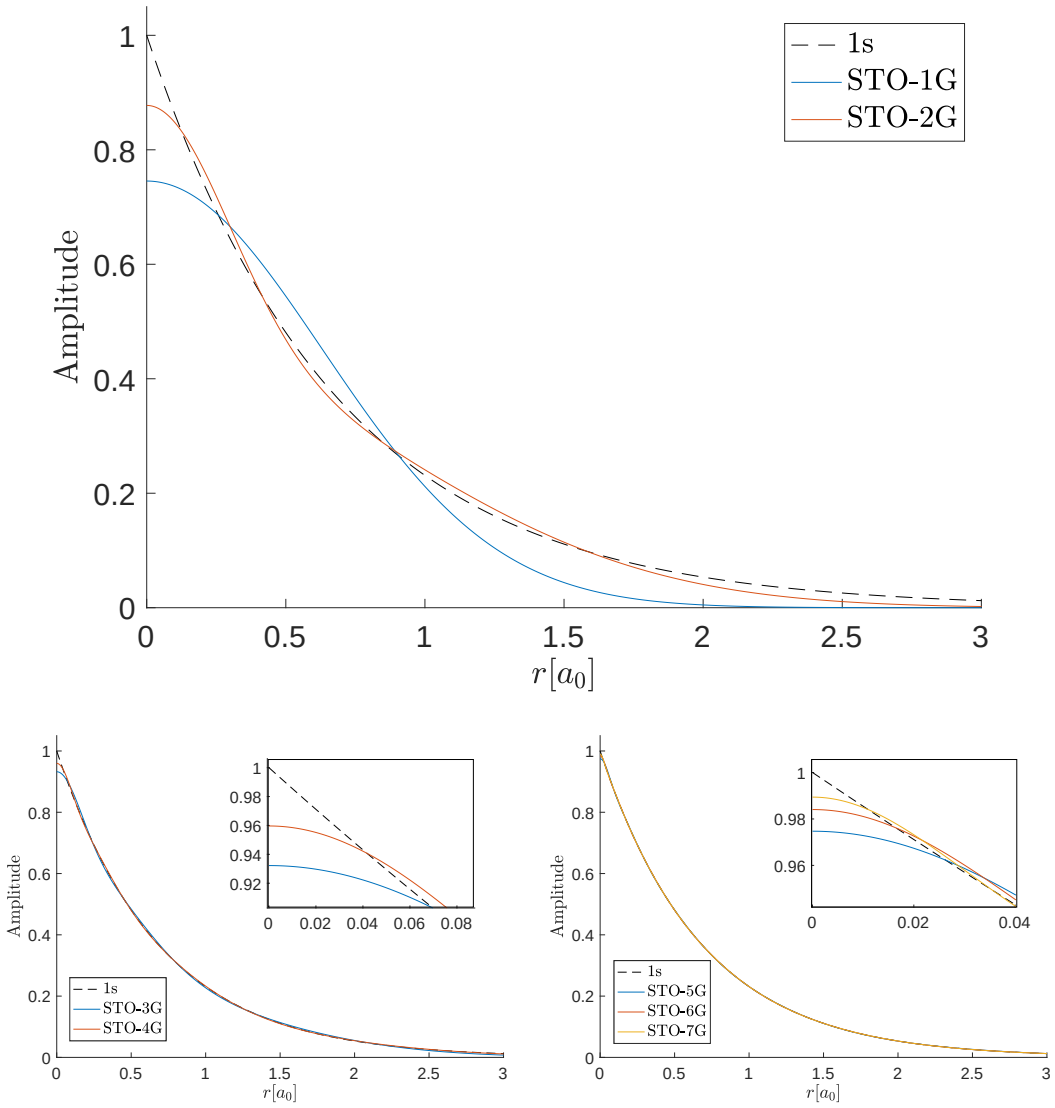

Figure 3.5: Example showing a STO-nG fit with different number of primitives to the 1s STO which we were trying to approximate. The s-type STO has exponent $\zeta = \sqrt[3]{\pi} \approx 1.4646$, arbitrarily chosen simply to ensure $\mathrm{STO}(r) \to 1$ as $r \to 0$.

used in either of the two valence contracted Gaussians. An example is the 3-21G set, where each core orbital is allotted a single contracted function consisting of three primitives. Next, a contracted function of two primitives *and* a contracted function of a single primitive is used to represent each valence orbital.

As an instructive example, consider the electronic structure of B. Of the five electrons in the boron, two occupy the 1s states essentially forming a He core. The outermost electrons inhabit the 2s atomic orbitals. This is denoted B: $1s^2 2s^2 2p$ or more frequently [He]$2s^2 2p$ [46]. In the 6-31G boron basis set of Dill and Pople, the core 1s orbital is represented by one contracted Gaussian of six primitives [92]. The 2s valence orbital is given one contracted of three primitives, and one of a single primitive. The same is done for each of the three different 2p orbitals, yielding two contracted functions for each. In total this gives 22 primitives across 9 total contracted basis functions.

Polarized, diffuse, and correlation-consistent basis sets

Instead of merely adding more Gaussian functions representing the electron orbitals present in any given atom, even more flexibility may be added by considering also orbitals which are not occupied. For example, trying to represent molecular bonding in H_2 with a minimal basis set would be impossible because there is no way to engineer a higher electron density in between the atoms with only s-type orbitals. An orbital of p-type symmetry or higher is needed in order to achive this. Pople and co-workers' notation adds one or more *s to denote the presence of such **polarizing functions**. For example, the 6-31G** basis set adds three single primitive contracted 2p orbitals to H (6-31G* only adds polarization to heavier atoms, making 6-31G and 6-31G* identical for H).

The same philosophy is applied for **diffuse functions**, which have significantly smaller exponents than the most wide-spread valence Gaussians. Such basis functions are neccessary to obtain an adequate description of e.g. negative ions, highly excited states, or loosely bonded (non-covalent, i.e. van der Waals bonds or similar) molecular structures [76]. In the Pople family, addition of diffuse orbitals is denoted by prepending one or more +s to the trailing G, e.g. 6-31++G** which adds a single diffuse 1s function of one primitive to H (again, 6-31+G** only adds diffuse functions to heavier atoms, making it identical to 6-31G** for H).

Lastly, we discuss briefly the correlation-consistent Dunning familiy of basis sets [93]. Correlation-consistent means the exponents and contraction coefficients are variationally optimized for post-Hartree-Fock methods, involving dynamic electron correlations. The notation used by Dunning and co-workers is cc-pVNZ, where N can be D (double ζ), T (triple ζ), Q (quadruple ζ), and so on. The cc-pV stands for correlation-consistent polarized, valence only: polarizing functions are added (more for higher ζ sets), and the basis only describes valence orbitals. For a complete description the cc-pVNZ basis needs to be paired with a corresponding basis set for the core orbitals. Some complete sets are used, denoted cc-pCVXZ, with X signifying the

ζ number for core electrons. Whenever diffuse basis functions are added, aug (for augmented) is prepended resulting in aug-cc-pVNZ.

Part II

Advanced theory

Chapter 4

Hartree-Fock

The Hartree-Fock (HF) method is one of the most important models in all of quantum chemistry, not only because it may yield acceptable approximations in certain scenarios, but because it is also an important stepping stone on the way to more accurate methods. Only a few of the more sophisticated quantum chemistry methods bypass HF entirely, while *most* of them use it as a first step and then build on the HF orbitals to obtain more accurate descriptions [22]. In particular, for larger systems, the Hartree-Fock approach may be the only feasible one and it is the only approximate method that is *routinely* being applied to *large* systems of several hundred atoms and molecules [13].

The Hartree-Fock method is a *mean field* method in that it treats the inter electron interaction only in an averaged way [14]. Any single electron does not feel the effect of every other localized electron, but rather just an averaged potential from all other remaining ones. This is sometimes also called an *independent-particle* model. The Hartree-Fock approximation usually *defines* the dynamical coulomb correlation between electrons by saying the difference between the Hartree-Fock energy and the exact quantum mechanical energy is the correlation energy. Hartree-Fock nevertheless deals exactly with the electron correlations arising from the anti-symmetry condition of Pauli, namely the exchange correlations.

In essence, the Hartree-Fock procedure finds the most energetically favorable electronic configuration under the assumption that the full ground state wave function consists of a *single* Slater determinant populated by orthonormal spin-orbitals. In older litterature, the HF method is often called *self-consitent field* method due to the way the resulting equations are usually solved [94]. However, the self-consistent field iterations are not the *only* way to solve the HF equations, and thus not an essential part of the method itself [13].

In the following we will apply the variational principle to the single Slater determinant ansatz wave function for the interacting system of N electrons. We will then expand the solution in a given basis and derive the Roothan-Hall and Pople-Nesbet equations, for the closed-shell and open-shell systems respectively.

4.1 Single Slater determinant ansatz

The method itself essentially finds the most energetically favorable electronic wave function, under the assumption that the full ground state consists of *a single* Slater determinant populated by orthonormal spin-orbitals, ϕ_i. We denote this Slater determinant $|\Psi\rangle$,

$$|\Psi\rangle = |\phi_0\phi_1\phi_2\dots\phi_{N-1}\phi_N\rangle, \quad \langle\phi_i|\phi_j\rangle = \delta_{ij}. \tag{4.1}$$

We may write down an explicit expression for the determinantal wave function in the position basis as

$$\Psi(\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_N(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_N(\mathbf{x}_N) \end{vmatrix}, \tag{4.2}$$

with $\phi_n(\mathbf{x}_k)$ being the index n spin-orbital evaluated at the spatial and spin-projection coordinates $\mathbf{x}_k$. Under the assumption that the spin-orbitals themselves are orthonormal, the total determinant will also be normalized in the sense that $\langle\Phi|\Phi\rangle = 1$ [14].

Recall from section 2.4.2 that the Born-Oppenheimer Hamiltonian for a system of N electrons subject to the Coulomb potential from M atoms takes the form

$$\begin{aligned} \hat{H} &= \underbrace{-\sum_{i=1}^{N}\frac{\nabla^2}{2} - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_A-\mathbf{r}_i|}}_{\equiv\ \sum_{i=1}^{N}\hat{h}_i} + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} \\ &\equiv \sum_{i=1}^{N}\hat{h}_i + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\hat{w}_{ij}, \end{aligned} \tag{4.3}$$

where we have defined the *one-body operator* $\hat{h}_i = -\nabla^2/2 - \sum_A Z_A/|\mathbf{r}_A - \mathbf{r}_i|$. The *two-body operator* $\hat{w}_{ij}$ represents the Coulombic electron-electron interaction between electrons labelled i and j.

4.1.1 Exchange correlation

With only applying the Slater determinant ansatz, the electrons are already correlated. If we consider the probability of finding two electrons at coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively, [57]

$$\begin{aligned} \rho(\mathbf{x}_1,\mathbf{x}_2) &= \int \mathrm{d}^4\mathbf{x}_3\,\mathrm{d}^4\mathbf{x}_4\,\dots\,\mathrm{d}^4\mathbf{x}_N\,|\Psi(\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N)|^2 \\ &= \frac{1}{N(N-1)}\sum_{k=1}^{N}\sum_{l=1}^{N}\Big[|\phi_k(\mathbf{x}_1)|^2|\phi_l(\mathbf{x}_2)|^2 - \phi_k^*(\mathbf{x}_1)\phi_k(\mathbf{x}_2)\phi_l^*(\mathbf{x}_2)\phi_l(\mathbf{x}_1)\Big]. \end{aligned} \tag{4.4}$$

In order to relate this to spatial coordinates only, we need to sum over the spin variables,

$$\rho(\mathbf{r}_1, \mathbf{r}_2) = \sum_{s_1} \sum_{s_2} \rho(\mathbf{x}_1, \mathbf{x}_2), \tag{4.5}$$

which means the second term vanishes for opposite spin electrons. However, for same spin electrons, the second term of Eq. (4.4) gives rise to a correlation effect: for electrons of the same spin-projection the first and second term cancel exactly for $\mathbf{r}_1 = \mathbf{r}_2$ [57]. This is known as *exchange correlation*, all electrons are surrounded by *exchange holes* where the chance of finding other like-spin electrons is drastically reduced.

4.2 The Hartree-Fock energy

Assuming now the wave function takes the form of a single Slater determinant inhabited by *orthonormal* orbitals, let us work out what the exected value of the energy is. The Hamiltonian consists of two parts—a one-body and a two-body term—which we will handle separately.

One-body Hamiltonian

The electronic one-body part of the Hamiltonian takes the form

$$\hat{H}_0 = \sum_{i=1}^{N} \hat{h}_i = \sum_{i=1}^{N} \left[-\frac{\nabla_i^2}{2} - \sum_{A=1}^{M} \frac{Z_A}{|\mathbf{x}_i - \mathbf{x}_A|} \right], \tag{4.6}$$

where M denotes the number of nuclei. Since $\hat{h}_i$ only acts on the coordinates of particle i, we find

$$\langle \Psi | \hat{H}_0 | \Psi \rangle = \int \mathrm{d}^4\mathbf{x}_1 \ldots \mathrm{d}^4\mathbf{x}_N \, \Psi^*(\mathbf{X}) \sum_{i=1}^{N} \hat{h}_i \Psi(\mathbf{X}) \tag{4.7}$$

with terms

$$\begin{aligned}
(4.7) &= \int \mathrm{d}^4\mathbf{x}_1 \ldots \mathrm{d}^4\mathbf{x}_N \, \Psi^*(\mathbf{X}) \hat{h}_k \Psi(\mathbf{X}) \\
&= \frac{1}{N!} \int \mathrm{d}^4\mathbf{x}_1 \ldots \mathrm{d}^4\mathbf{x}_N \sum_{\mu,\nu \in S_N} (-1)^{|\mu|+|\nu|} \hat{P}_\mu \hat{P}_\nu \Psi^*(\mathbf{x}_{\mu(1)} \ldots \mathbf{x}_{\mu(N)}) \hat{h}_k \Psi(\mathbf{x}_{\nu(1)} \ldots \mathbf{x}_{\nu(N)}) \\
&= \frac{1}{N!} \sum_{\mu,\nu \in S_N} (-1)^{|\mu|+|\nu|} \hat{P}_\mu \hat{P}_\nu \delta_{\mu(1)}^{\nu(1)} \ldots \delta_{\mu(k-1)}^{\nu(k-1)} \delta_{\mu(k+1)}^{\nu(k+1)} \ldots \delta_{\mu(N)}^{\nu(N)} \\
&\qquad \times \int \mathrm{d}^4\mathbf{x}_k \phi_{\mu(k)}^*(\mathbf{x}_k) \hat{h}_k \phi_{\nu(k)}(\mathbf{x}_k).
\end{aligned} \tag{4.8}$$

Note that $\mathbf{x} = \{\mathbf{r}, \sigma\}$ labels both spatial and spin coordinates, with $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_N\}$. Here, $\hat{P}_\mu$ acts on the orbital indices of $\Psi^*(\mathbf{X})$, while $\hat{P}_\nu$ acts on the corresponding indices of $\Psi(\mathbf{X})$. Since all indices must appear exactly once, and there are $N-1$ delta functions, the only suriving terms appear for permutations which satisfy $\mu(k) = \nu(k)$. In fact, μ must be equal to ν on the whole, making $(-1)^{|\mu|+|\nu|} = +1$.

In total, there are $N!$ possible permutations, meaning we have $(N-1)!$ permutations in the sum of Eq. (4.8),

$$\begin{aligned}(4.8) &= \frac{(N-1)!}{N!} \sum_{i=1}^{N} \int \mathrm{d}^4\mathbf{x}_k \phi_i^*(\mathbf{x}_k) \hat{h}_k \phi_i(\mathbf{x}_k) \\ &= \frac{1}{N} \sum_{i=1}^{N} \int \mathrm{d}^4\mathbf{x}_k \phi_i^*(\mathbf{x}_k) \hat{h}_k \phi_i(\mathbf{x}_k). \end{aligned} \tag{4.9}$$

Since we have one such term for each k, we find in total that [22]

$$\langle \Psi | \hat{H}_0 | \Psi \rangle = \sum_{i=1}^{N} \int \mathrm{d}^4\mathbf{x}\, \phi_i(\mathbf{x}) \hat{h} \phi_i(\mathbf{x}), \tag{4.10}$$

where we have omitted the arbitrary subscript on $\mathbf{x}$ and $\hat{h}$.

Two-body Hamiltonian

The electronic two-body part of the Hamiltonian takes the form

$$\hat{W} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \hat{w}_{ij} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{1}{|\mathbf{x}_i - \mathbf{x}_j|}. \tag{4.11}$$

In the same way as before, we insert into $\langle \Psi | \hat{W} | \Psi \rangle$ the definition of the Slater determinants and find

$$\langle \Psi | \hat{W} | \Psi \rangle = \int \mathrm{d}^4\mathbf{x}_1 \ldots \mathrm{d}^4\mathbf{x}_N\, \Psi^*(\mathbf{X}) \sum_{i=1}^{N} \sum_{j=i+1}^{N} \hat{w}_{ij} \Psi(\mathbf{X}), \tag{4.12}$$

with terms

$$\frac{1}{N!} \int \mathrm{d}^4\mathbf{x}_1 \ldots \mathrm{d}^4\mathbf{x}_N\, (-1)^{|\mu|+|\nu|} \hat{P}_\mu \hat{P}_\nu \Psi^*(\mathbf{x}_{\mu(1)} \ldots \mathbf{x}_{\mu(N)}) \hat{w}_{ij} \Psi(\mathbf{x}_{\mu(1)} \ldots \mathbf{x}_{\mu(N)}). \tag{4.13}$$

The orthogonality of the orbitals ensures that all $\mu(k) = \nu(k)$ for all k *except for* i and j. There are now two non-vanishing contributions: $\mu(i) = \nu(i)$ and $\mu(j) = \nu(j)$, or $\mu(i) = \nu(j)$ and $\mu(j) = \nu(i)$. In the latter case, the factor $(-1)^{|\mu|+|\nu|}$ contributes

an overall minus sign. The sum over all non-zero permutations now entails $(N-2)!$ terms,

$$(4.13) = \frac{(N-2)!}{N!}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\int \mathrm{d}^4\mathbf{x}_1\mathrm{d}^4\mathbf{x}_2\,\Big[\phi_i^*(\mathbf{x}_1)\phi_j^*(\mathbf{x}_2)\hat{w}\phi_i(\mathbf{x}_1)\phi_j(\mathbf{x}_2) - \phi_i^*(\mathbf{x}_1)\phi_j^*(\mathbf{x}_2)\hat{w}\phi_j(\mathbf{x}_1)\phi_i(\mathbf{x}_2)\Big], \tag{4.14}$$

with $N(N-1)$ total such terms giving finally [14]

$$\langle\Psi|\hat{W}|\Psi\rangle = \sum_{i=1}^{N}\sum_{j=i+1}^{N}\int \mathrm{d}^4\mathbf{x}_1\mathrm{d}^4\mathbf{x}_2\,\phi_i^*(\mathbf{x}_1)\phi_j^*(\mathbf{x}_2)\hat{w}\Big[\phi_i(\mathbf{x}_1)\phi_j(\mathbf{x}_2) - \phi_j(\mathbf{x}_1)\phi_i(\mathbf{x}_2)\Big]. \tag{4.15}$$

Total expression for the Hatree-Fock energy

Combining the one-, and two-body expectation values, we find

$$\begin{aligned}\langle\Psi|\hat{H}|\Psi\rangle &= \left\langle\Psi\middle|\sum_{i=1}^{N}\left[-\frac{\nabla_i^2}{2} - \sum_{A=1}^{M}\frac{Z_A}{|\mathbf{x}_A-\mathbf{x}_i|} + \sum_{j=i+1}^{N}\frac{1}{|\mathbf{x}_i-\mathbf{x}_j|}\right]\middle|\Psi\right\rangle \\ &= \sum_{i=1}^{N}\langle\phi_i|\hat{h}|\phi_i\rangle + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\Big[\langle\phi_i\phi_j|\hat{w}|\phi_i\phi_j\rangle - \phi_i\phi_j|\hat{w}|\phi_j\phi_i\rangle\Big] \\ E_{\text{HF}} &= \sum_{i=1}^{N}\langle i|\hat{h}|i\rangle + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\langle ij|\hat{w}|ij\rangle - \langle ij|\hat{w}|ji\rangle,\end{aligned} \tag{4.16}$$

where we used the notation

$$\langle\phi_i|\hat{h}|\phi_i\rangle = \langle i|\hat{h}|i\rangle = \int \mathrm{d}^4\mathbf{x}\,\phi_i^*(\mathbf{x})\hat{h}\phi_i(\mathbf{x}), \tag{4.17}$$

and

$$\langle\phi_i\phi_j|\hat{w}|\phi_i\phi_j\rangle = \langle ij|\hat{w}|ij\rangle = \int \mathrm{d}^4\mathbf{x}_1\mathrm{d}^4\mathbf{x}_2\,\phi_i^*(\mathbf{x}_1)\phi_j^*(\mathbf{x}_2)\hat{w}\phi_i(\mathbf{x}_1)\phi_j(\mathbf{x}_2). \tag{4.18}$$

4.3 Variational minimization of E_{HF}

We now vary the spin-orbitals $\phi_k \to \phi_k + \delta\phi_k$ in order to find the variational minimum. Recall from section 2.7 that the energy functional satisfies $\delta E[\Psi]|_{\Psi=\Psi_0} = 0$ evaluated at *the true ground state* and that finding such a wave function constitutes solving the Schrödinger equation.

In order to ensure orthonormality still holds during and after the variation, we introduce Lagrange multipliers ε_{ij}, one for each integral

$$\int \mathrm{d}^4\mathbf{x}\, \phi_i^*(\mathbf{x})\phi_j(\mathbf{x}) \stackrel{!}{=} \delta_{ij}. \tag{4.19}$$

The resulting Lagrange functional takes the form

$$\begin{aligned}\mathcal{L}\big[\phi_1, \phi_2, \ldots, \phi_N\big] &= \langle\Psi|\hat{H}|\Psi\rangle - \sum_{i=1}^{N}\sum_{j=1}^{N}\varepsilon_{ij}\Big(\langle\phi_i|\phi_j\rangle - \delta_{ij}\Big) \\ &= \sum_{i=1}^{N}\langle i|\hat{h}|i\rangle + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\langle ij|\hat{w}|ij - ji\rangle - \sum_{i=1}^{N}\sum_{j=1}^{N}\varepsilon_{ij}\Big(\langle\phi_i|\phi_j\rangle - \delta_{ij}\Big),\end{aligned}$$

where we included a factor $1/2$ in front of the $\hat{W}$ term because the sum is now taken from $j = 1$ [68].

It turns out that varying only ϕ_k^* is sufficient to derive all the Hartree-Fock equations [14]. In addition, due to the symmetry of the Langrange multipliers, the multiplier matrix ε can be assumed to be Hermitian [57]. Assume now that ϵ is a small complex number and η a normalized orbital, and take $\delta\phi_k$ to be $\epsilon\eta$. We define the function

$$f(\epsilon) = \mathcal{L}\Big[\phi_1, \phi_2, \ldots, \phi_{k-1}, \phi_k + \epsilon\eta, \phi_{k+1}, \ldots, \phi_N\Big], \tag{4.20}$$

and note that to first order in ϵ, $f(\epsilon) = f(0) + \epsilon f'(0) + \mathcal{O}(\epsilon^2)$. At the variational minimum, $f'(0)$ for *any* η. For a (very) brief introduction to functional derivatives, see appendix C.

Keeping the other orbitals fixed, we consider now the Taylor expansion of f to first order in ϵ:

$$\begin{aligned}f(\epsilon) &= \sum_{i=1}^{N}\langle i + \varepsilon_{ik}\epsilon\eta|\hat{h}|i\rangle + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big\langle\big(i + \varepsilon_{ki}\epsilon\eta\big)\big(j + \varepsilon_{kj}\epsilon\eta\big)\big|\hat{w}\big|ij - ji\Big\rangle \\ &\qquad - \sum_{i=1}^{N}\sum_{j=1}^{N}\Big[\big\langle\big(i + \varepsilon_{ik}\epsilon\eta\big)\big|j\big\rangle - \varepsilon_{ij}\Big] \\ &= \sum_{i=1}^{N}\langle i|\hat{h}|i\rangle + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\big\langle ij\big|\hat{w}\big|ij - ji\big\rangle + \epsilon\langle\eta|\hat{h}|\phi_k\rangle \\ &\qquad + \frac{\epsilon}{2}\left[\sum_{i=1}^{N}\langle\eta i|\hat{w}|ki\rangle + \langle i\eta|\hat{w}|ik\rangle - \langle\eta i|\hat{w}|ik\rangle - \langle i\eta|\hat{w}|ki\rangle\right] \\ &\qquad - \sum_{j=1}^{N}\varepsilon_{jk}\epsilon\langle\eta|j\rangle - \sum_{i=1}^{N}\sum_{j=1}^{N}\varepsilon_{ij}\Big(\langle\phi_i|\phi_j\rangle - \delta_{ij}\Big) + \mathcal{O}(\epsilon^2).\end{aligned} \tag{4.21}$$

By the symmetry of the $\hat{w}$ integrals we obtain

$$f(\epsilon) = \sum_{i=1}^{N}\langle i|\hat{h}|i\rangle + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\langle ij|\hat{w}|ij - ji\rangle - \sum_{i=1}^{N}\sum_{j=1}^{N}\varepsilon_{ij}\Big(\langle\phi_i|\phi_j\rangle - \delta_{ij}\Big) + \epsilon\langle\eta|\hat{h}|\phi_k\rangle + \epsilon\sum_{j=1}^{N}\Big[\langle\eta j|\hat{w}|kj\rangle - \langle\eta j|\hat{w}|jk\rangle\Big] - \sum_{j=1}^{N}\varepsilon_{jk}\epsilon\langle\eta|j\rangle + \mathcal{O}(\epsilon^2), \tag{4.22}$$

where we note that $f(0) = \mathcal{L}[\phi_1, \phi_2, \ldots, \phi_N, \varepsilon]$ and we can read off $f'(0)$ as [14]

$$f'(0) = \langle\eta|\hat{h}|\phi_k\rangle + \sum_{j=1}^{N}\Big[\langle\eta j|\hat{w}|kj\rangle - \langle\eta j|\hat{w}|jk\rangle\Big] - \sum_{j=1}^{N}\varepsilon_{jk}\langle\eta|j\rangle \stackrel{!}{=} 0. \tag{4.23}$$

Without changing the Slater determinant (up to a phase which doesnt affect the physics) we may apply a unitary transformation to the orbitals such that

$$\tilde{\phi}_k = \sum_{j=1}^{N}\phi_j U_{jk}. \tag{4.24}$$

Since $\det U = \mathrm{e}^{\mathrm{i}\theta}$ for some *real* θ—and we know that the Slater transforms as $|\tilde{\Phi}\rangle = \det U|\Phi\rangle$—the state does not change under this transformation and so the energy must also remain the same. As mentioned, ε can be assumed to be Hermitian, which means we may choose U such that $\varepsilon = UEU^\dagger$ with $E_{ij} = \delta_{ij}\varepsilon_i$ (with ε_i being the eigenvalues of the ε matrix).

Since Eq. (4.23) must hold for *any* η, the only possible solution is for all the integrands to vanish. Notice that

$$\langle\,\cdot\, i|\hat{w}|jk\rangle = \int \mathrm{d}^4\mathbf{x}_2\,\phi_i^*(\mathbf{x}_2)\hat{w}\phi_j(\mathbf{x}_1)\phi_k(\mathbf{x}_2) \tag{4.25}$$

defines a single particle operator where the inner product with any $\langle l|$ gives the two-body integral $\langle li|\hat{w}|jk\rangle$. The condition that the integrands of Eq. (4.23) must exactly vanish can be written in terms of these operators, as

$$\sum_{j=1}^{N}\varepsilon_{jk}|\phi_j\rangle = \hat{h}|\phi_k\rangle + \sum_{j=1}^{N}\Big[\langle\,\cdot\,\phi_j|\hat{w}|\phi_k\phi_j\rangle - \langle\,\cdot\,\phi_j|\hat{w}|\phi_j\phi_k\rangle\Big]. \tag{4.26}$$

Under the unitary transformation of Eq. (4.24), this yields finally the **canonical Hartree-Fock equations**

$$\varepsilon_i|\phi_i\rangle = \hat{h}|\phi_i\rangle + \sum_{j=1}^{N}\Big[\langle\,\cdot\,\phi_j|\hat{w}|\phi_i\phi_j\rangle - \langle\,\cdot\,\phi_j|\hat{w}|\phi_j\phi_i\rangle\Big]. \tag{4.27}$$

4.3.1 Defining $\hat{J}$, $\hat{K}$ and the Fock operator

The expression for the Hartree-Fock energy and the Hartree-Fock equations can be simplified by the introduction of three operators. The *Coulomb* and *exchange* operators,

$$\begin{aligned}\hat{J}_k(\mathbf{x})\phi(\mathbf{x}) &= \langle \cdot\, \phi_k|\hat{w}|\phi\phi_k\rangle = \int \mathrm{d}^4\mathbf{x}_2\, \phi_k^*(\mathbf{x}_2)\hat{w}\phi(\mathbf{x}_1)\phi_k(\mathbf{x}_2) \\ &= \int \mathrm{d}^4\mathbf{x}_2\, |\phi_k(\mathbf{x}_2)|^2\hat{w}\phi(\mathbf{x}_1), \quad \text{and} \end{aligned} \tag{4.28}$$

$$\hat{K}_k(\mathbf{x})\phi(\mathbf{x}) = \langle \cdot\, \phi_k|\hat{w}|\phi_k\phi\rangle = \int \mathrm{d}^4\mathbf{x}_2\, \phi_k^*(\mathbf{x}_2)\hat{w}\phi(\mathbf{x}_2)\phi_k(\mathbf{x}_1) \tag{4.29}$$

are defined such that the HF equations can be written in the succinct form [57]

$$\left[\hat{h} + \hat{J} - \hat{K}\right]\phi_i = \varepsilon_i\phi_i. \tag{4.30}$$

The $\hat{J}$ and $\hat{K}$ with no subscripts are sums over all orbitals,

$$\hat{J} = \sum_{k=1}^{N}\hat{J}_k, \quad \text{and} \quad \hat{K} = \sum_{k=1}^{N}\hat{K}. \tag{4.31}$$

In this notation, the HF energy may also be expressed with brevity:

$$E\big[\Psi\big] = \sum_{k=1}^{N}\left\langle \phi_k\middle|\hat{h} + \frac{1}{2}\Big(\hat{J} - \hat{K}\Big)\middle|\phi_k\right\rangle. \tag{4.32}$$

The Coulomb operator represents the averaged electronic repulsion from all the other electrons. This also includes un-physical *self-interaction* with $i = j$, but this is luckily cancelled exactly by corresponding terms in the exchange term. The exchange operator is present simply because of the anti-symmetry properties of the wave function—as was mentioned briefly in section 4.1.1—which makes same-spin electrons repel each other. This is nothing more than a manifestation of the Pauli principle, see section 2.5 [68].

The sum of $\hat{h}$, and the Coulomb and exchange operators defines the **Fock operator**, [57]

$$\hat{F} = \hat{h} + \hat{J} - \hat{K}, \tag{4.33}$$

and we note that the HF equations can be written as $\hat{F}\phi_k = \varepsilon_k\phi_k$.

4.4 Restricted Hartree-Fock

Under the assumption that all spatial orbitals are doubly occupied—i.e. the even index spin-orbitals represent $\phi_{2n'}(\mathbf{x}) = \psi_n(\mathbf{r})\chi(\uparrow)$, with odd numbered $\phi_{2n'+1}(\mathbf{x}) =$

$\psi_n(\mathbf{r})\chi(\downarrow)$, for *spatial* orbitals ψ—the Hartree-Fock framework simplifies considerably. Under Restricted Hartree-Fock (RHF), we may explicitly integrate out the spin-dependency of the Fock operator.

Let us now consider the Coulomb operator, $\hat{J}$. Since the spin-orbitals entering $\hat{J}$ are always lined up in the sense that the integral over $\mathbf{x}_2$ happens over $\phi^*_{k'}(\mathbf{x}_2)\phi_{k'}(\mathbf{x}_2) = \psi^*_k(\mathbf{r}_2)\chi^*(\sigma_k)\psi_k(\mathbf{r}_2)\chi(\sigma_k)$, the spin functions $\chi(\sigma_k)$ are always equal. This means they integrate out, and we can write the *spatial* Coulomb operator $\tilde{J}$ as [57]

$$\begin{aligned}\hat{J}(\mathbf{x})\phi(\mathbf{x}) &= \sum_{i=1}^{N}\int \mathrm{d}^4\mathbf{x}_2\, \phi_i^*(\mathbf{x}_2)\phi_i(\mathbf{x}_2)\hat{w}\phi(\mathbf{x}) \\ &= \sum_{i=1}^{N}\int \mathrm{d}\sigma_2 \int \mathrm{d}^3\mathbf{r}_2\, \psi_i^*(\mathbf{r}_2)\chi^*(\sigma_2)\psi_i(\mathbf{r}_2)\chi(\sigma_2)\hat{w}\psi(\mathbf{r})\chi(\sigma) \\ &= 2\sum_{l=1}^{N/2}\int \mathrm{d}^3\mathbf{r}_2\, \psi_l^*(\mathbf{r}_2)\psi_l(\mathbf{r}_2)\hat{w}\psi(\mathbf{r}) = \tilde{J}(\mathbf{r})\psi(\mathbf{r}). \end{aligned} \tag{4.34}$$

Note the changed sum limits in the last equation, we are now only summing over half l up to $N/2$. A corresponding doubling of the exchange operator does *not* happen, since the spin integral only evaluates to one half of the time. We find thus [22]

$$\begin{aligned}\hat{K}(\mathbf{x})\phi(\mathbf{x}) &= \sum_{i=1}^{N}\int \mathrm{d}^4\mathbf{x}_2\, \phi_i^*(\mathbf{x}_2)\phi(\mathbf{x}_2)\hat{w}\phi_i(\mathbf{x}) \\ &= \sum_{i=1}^{N}\int \mathrm{d}\sigma_2 \int \mathrm{d}^3\mathbf{r}_2\, \psi_i^*(\mathbf{r}_2)\chi^*(\sigma_2)\psi(\mathbf{r})\chi(\sigma)\hat{w}\psi(\mathbf{r}_2)\chi(\sigma_2) \\ &= \sum_{l=1}^{N/2}\int \mathrm{d}^3\mathbf{r}_2\, \psi_l^*(\mathbf{r}_2)\psi(\mathbf{r})\hat{w}\psi_l(\mathbf{r}_2) = \tilde{K}(\mathbf{r})\psi(\mathbf{r}). \end{aligned} \tag{4.35}$$

This partial supression of the exchange operator can be understood by the fact that exchange correlation is only felt by same-spin electrons. Thus explicitly integrating away the spin degrees of freedom reveals that for any spin-orbital $\phi(\mathbf{x}) = \psi(\mathbf{r})\chi(\sigma)$, the $\hat{K}_k$ operator only has an effect for orbitals of corresponding spin projection σ. The number of such orbitals is exactly $N/2$.

Since the one-body operator $\hat{h}$ has no explicit *or* implicit spin-dependence, it remains unchanged in the RHF scheme. We may now write down the spatial, restricted, Fock operator

$$\tilde{F}(\mathbf{r}) = \hat{h}(\mathbf{r}) + 2\tilde{J}(\mathbf{r}) - \tilde{K}(\mathbf{r}) \tag{4.36}$$

and the restricted Hartee-Fock energy

$$\tilde{E}_{\text{HF}} = 2\sum_{k=1}^{N/2}\langle\psi_k|\hat{h}|\psi_k\rangle + \sum_{k=1}^{N/2}\Big[2\big\langle\psi_k|\tilde{J}|\psi_k\big\rangle - \big\langle\psi_k|\tilde{K}|\psi_k\big\rangle\Big]. \tag{4.37}$$

Note carefully the subtle (read: lazy) redefinition of $\langle \cdot | \cdot \rangle$ to now only include *spatial* integrals when discussing RHF.

4.4.1 The Roothan-Hall equations

In order to discretize the RHF equations in a way suitable for numerical solution, we apply the method of Galerkin [81]. This method was independently developed by both Dutch physicist C. C. J. Roothan and Irish mathematician G. C. Hall in 1951 [95, 96].

Consider a finite basis set of L spatial orbitals, $\{\varphi_k\}_{k=1}^{L}$. Whereas the number of occupied restricted Hartree-Fock orbitals is constrained to be $N/2$, this basis set can in general be any size. Let us now expand ψ_i in terms of this basis,

$$\psi_i = \sum_{k=1}^{L} C_{ki}\varphi_k(\mathbf{r}), \tag{4.38}$$

which gives us the RHF equations projected onto the Hilbert subspace spanned by the new basis set—$\mathcal{H}' = \text{span}\{\varphi_k\}$—as

$$\sum_{k=1}^{L} C_{ki}\tilde{F}(\mathbf{r})\varphi_k(\mathbf{r}) = \varepsilon_i \sum_{k=1}^{L} C_{ki}\varphi_k(\mathbf{r}). \tag{4.39}$$

Taking the inner product with $\varphi_q(\mathbf{r})$ gives the *weak formulation* of the RHF equations on $\mathcal{H}'$:

$$\sum_{k=1}^{L} C_{ki} \int \mathrm{d}^3\mathbf{r}\, \varphi_q(\mathbf{r})\tilde{F}(\mathbf{r})\varphi_k(\mathbf{r}) = \varepsilon_i \sum_{k=1}^{L} C_{ki} \int \mathrm{d}^3\mathbf{r}\, \varphi_q(\mathbf{r})\varphi_k(\mathbf{r}). \tag{4.40}$$

We can identify Eq. (4.40) as a matrix equation in terms of the **Fock matrix** F, the **coefficient matrix** C, and the **overlap matrix** S,

$$FC = \boldsymbol{\varepsilon} SC. \tag{4.41}$$

The components of the (restricted) Fock matrix are sums of one-, and two-body integrals in terms of the new basis functions

$$F_{pq} = h_{pq} + \sum_{k=1}^{N/2}\sum_{r=1}^{L}\sum_{s=1}^{L} \underbrace{C_{rk}^{*}C_{sk}}_{\equiv D_{rs}/2} \Big[2\langle pr|\hat{w}|qs\rangle - \langle pr|\hat{w}|sq\rangle\Big], \tag{4.42}$$

with

$$h_{pq} = \langle p|\hat{h}|q\rangle = \int \mathrm{d}^3\mathbf{r}\, \varphi_p^{*}(\mathbf{r}) \left[-\frac{\nabla^2}{2} - \sum_{A=1}^{M} \frac{Z_A}{|\mathbf{r}-\mathbf{r}_A|}\right] \varphi_q(\mathbf{r}). \tag{4.43}$$

It is convenient to define the density matrix, $D_{pq} = 2\sum_{k=1}^{N/2} C_{pk}^* C_{qk}$ with a factor 2 coming from the spin-restricted nature of the Roothan-Hall equations [14]. In terms of the density matrix, and the one-, and two-body integrals over φs, the energy is given as

$$E_{\text{HF}} = \sum_{pq} D_{pq} h_{pq} + \frac{1}{2}\sum_{pqrs} D_{pq} D_{sr} \Big[\langle pr|\hat{w}|qs\rangle - \frac{1}{2}\langle pr|\hat{w}|sq\rangle\Big]. \tag{4.44}$$

We outline in detail how such an equation may be solved in section 8.3.4.

4.5 Unrestricted Hartree-Fock and the Pople-Nesbet equations

Relaxing the condition that all occupied spatial orbitals be paired leads to what is known as *unrestricted Hartree-Fock*. In the weak formulation—à la the Roothan-Hall equation—the unrestricted scheme gives rise to the Pople-Nesbet equations [97]. These are essentially two sets of coupled Roothan-Hall equations, one for spin-up and one for spin-down electrons,

$$F^{\uparrow} C^{\uparrow} = \boldsymbol{\varepsilon}^{\uparrow} S C^{\uparrow}, \quad \text{and} \tag{4.45}$$

$$F^{\downarrow} C^{\downarrow} = \boldsymbol{\varepsilon}^{\downarrow} S C^{\downarrow}. \tag{4.46}$$

Since $\hat{K}$ only couples same-spin electrons, there is no cross-term. The Coulumb term, however, couples opposite *and* same-spin electrons, and gives rise to a cross-term in $F^{\uparrow}$ depending on $C^{\downarrow}$ [57]

$$\begin{aligned} F_{pq}^{\uparrow} = h_{pq} &+ \sum_{k=1}^{N^{\uparrow}} \sum_{r=1}^{L} \sum_{s=1}^{L} C_{rk}^{\uparrow} C_{sk}^{\uparrow} \Big[\langle pq|\hat{w}|qs\rangle - \langle pq|\hat{w}|sq\rangle\Big] \\ &+ \sum_{k=1}^{N^{\downarrow}} \sum_{r=1}^{L} \sum_{s=1}^{L} C_{rk}^{\downarrow} C_{sk}^{\downarrow} \Big[\langle pq|\hat{w}|qs\rangle\Big]. \end{aligned} \tag{4.47}$$

The $F^{\downarrow}$ results from exchanging $\uparrow \leftrightharpoons \downarrow$ in Eq. (4.47).

4.6 Choice of orbital basis set

In the present work we use Gaussian type orbitals exclusively because of their favorable integration properties. This is discussed in depth in chapter 3. The specific basis sets in use are discussed alongside the HF code implementation in chapter 8.

4.7 The *Hartree-Fock limit*

In order to achieve completeness in the basis set representation of the Hartree-Fock orbitals, we need to (in general) use an infinite set. This is clearly not computationally feasible, so a cut-off is always chosen. However, using only L orbitals limits the accuracy with which we can represent arbitray spin-orbitals $\psi \in \mathcal{H}$. Expansion in the $\{\varphi_k\}_{k=1}^{L}$ basis constitutes a projection onto the L-dimensional Hilbert subspace $\mathcal{H}' = \text{span}\{\varphi_i\}$. This means we introduce an error in the Hartree-Fock orbitals ψ proportional to $\psi \in \mathcal{H} \setminus \mathcal{H}' = \{\psi \in \mathcal{H} \text{ and } \psi \notin \mathcal{H}'\}$.

This is known as *basis set truncation error* [68]. Taking larger and larger basis sets will predictably reduce this error and the limiting value—with an infinite basis—is known as the Hartree-Fock limit [22]. Even though we can never evaluate E_{HF} at the limit, there are ways to estimate the limiting value, see e.g. [98, 99].

The difference between the *true* (non-relativistic) ground state energy and the Hartree-Fock limit is known as the **correlation energy**,

$$E_{\text{corr}} = E_{\text{exact}} - E_{\text{HF (limit)}}. \tag{4.48}$$

Calling HF an un-correlated method altogether is a bit of a misnomer, as electron-electron correlation is taken into account albeit in the mean-field sense. Also worth noting is that HF fully accounts for the exchange correlation of same-spin electrons, c.f. section hfexchange. However, so-called *dynamic correlation* effects are completely neglected at the Hartree-Fock level of theory. Dynamic correlation refers the instantaneous Coulomb repulsion of two electrons moving around in space.

A separate correlation effect which HF fails to take into account is what is known as *static correlation*. A single Slater determinant may be fundamentally unable to accurately represent the true ground state wave function of any given quantum system. In some systems, only a linear combination of (nearly-)degenerate Slater determinants may describe the state well. This may be important in certain molecular systems, for which the single-Slater HF approximation is qualitatively wrong.

Chapter 5

Density functional theory

Because the wave function is such an unbelievably complicated function, depending on $4N$ degrees of freedom (of which $3N$ are spatial coordinates), it is natural to ask the question: Is it possible to represent the state of an electronic system in a more succinct way? A natural candidate for such an entity is the electronic number density, $\rho(\mathbf{r})$, which we will mostly refer to as simply *the density*. It would be remarkable if we could deal with enormously complex quantum mechanical systems by means of a function depending only on three spatial coordinates and spin(!) [61].

It turns out that exactly this is possible. There is a one-to-one correspondance between the ground state density, $\rho_0(\mathbf{r})$, and the external potential (up to an additive constant) and thus also the Hamiltonian [100]. Since the Hamiltonian uniquely determines all properties of a quantum mechanical system, we can in principle determine all the information in the many-body wave function (of the ground state and *all* excited states) from the ground state density alone [24]. The fact that the density can be determined from the wave function is almost trivially true, but that the converse is true is the content of the Hohenberg-Kohn theorems.

However, the theorems of Hohenberg and Kohn guarantee only the existence and uniqueness of an *energy functional*, $E[\rho]$, which can be used to determine the energy from the density. Without knowing the form of $E[\rho]$ and a computational scheme for calculating it, we are still no closer to being able to use the electron density as the basic variable in electronic structure calculations. This is where the Kohn-Sham ansatz comes in, making it possible for us to calculate structure properties of electronic systems by essentially solving a different system: a non-interacting system with the electrons moving in an effective potential which by construction yields the same ground state density as the original system.

We will begin our discussion of **Density functional theory** (DFT [sometimes prepended KS, making it Kohn-Sham density functional theory {KS-DFT}]) by considering the theoretical framework and the Hohenberg-Kohn theorems. Then we will consider the Kohn-Sham ansatz and how DFT calculations are performed in practice.

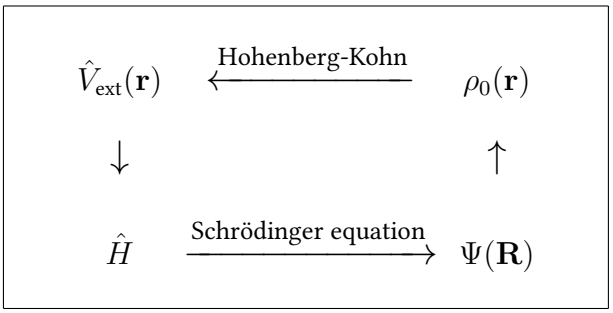

Figure 5.1: Schematic representation of the Hohenberg-Kohn theorems. Knowing the external potential fixes the Hamiltonian, from which we can extract the spectrum. The ground state density may then be extracted from the ground state wave function. The Hohenberg-Kohn theorems allows us (in principle) to find the potential if we know only the ground state wave function, making a one-to-one correspondence, $\hat{V}_{\text{ext}}(\mathbf{r}) \Leftrightarrow \rho_0(\mathbf{r})$. Adapted from a similar figure in [24].

5.1 The Hohenberg-Kohn theorems

Recall from 2.4.2 that the *electronic* Hamiltonian under the Born-Oppenheimer approximation takes the form

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i - \mathbf{r}_A|}. \tag{5.1}$$

We will in the following relabel the last term, and define

$$\sum_{i=1}^{N}\sum_{A=1}^{M}\frac{-Z_A}{|\mathbf{r}_i - \mathbf{r}_A|} \equiv \hat{V}_{\text{ext}}. \tag{5.2}$$

Since the first two terms of the Hamiltonian are the same for any system of N electrons, the external potential term $\hat{V}_{\text{ext}}$ completely fixes the electronic Hamiltonian as a whole. In fixing $\hat{H}$, we also fix the spectrum and so the ground state and all its derived properties are determined by N and $\hat{V}_{\text{ext}}$ alone [60]. One such property is of course the ground state electronic density.

In the following, we will denote by $v(\mathbf{r})$ the spatial (position basis) representation of the external potential $\hat{V}_{\text{ext}}$

The first Hohenberg-Kohn theorem

As mentioned already, the external potential trivially determines the electron density. The first theorem of Hohenberg and Kohn proves the highly non-trivial converse

statement: the external potential is uniquely determined (up to an additive constant) by the ground state electron density [101]. We outline the deceptively simple proof in the following paragraphs.

Consider two potentials, $\hat{V}_1$ and $\hat{V}_2$, differing by more than an additive constant, $\hat{V}_1 \neq \hat{V}_2 + \text{const}$ and denote the ground state energies of the corresponding Hamiltonians by E_1 and E_2 respectively. Assume now that the ground state wave functions of the two Hamiltonians are *the same*, $\Psi_1(\mathbf{R}) = \Psi_2(\mathbf{R})$. Since all parts of the Hamiltonian apart from $\hat{V}_{\text{ext}}$ coincide, subtracting the two Schrödinger equations give

$$\hat{H}_1|\Psi\rangle - \hat{H}_2|\Psi\rangle = \left(\hat{V}_1 - \hat{V}_2\right)|\Psi\rangle = (E_1 - E_2)\,|\Psi\rangle. \tag{5.3}$$

In terms of the wave functions (projecting the equation onto the position basis) this yields

$$\sum_{i=1}^{N} \big[\hat{v}_1(\mathbf{r}_i) - \hat{v}_2(\mathbf{r}_i)\big]\,\Psi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N) = (E_1 - E_2)\,\Psi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N), \tag{5.4}$$

which means $\hat{V}_1 - \hat{V}_2 = \text{const}$, in contradiction with the assumption.

We proceed thus with $\Psi_1(\mathbf{R})$ and $\Psi_2(\mathbf{R})$ neccessarily different. Assume now that $\Psi_1(\mathbf{R})$ and $\Psi_2(\mathbf{R})$ have the same ground state electronic density, $\rho_0(\mathbf{r})$. From the variational principle (c.f. section 2.7), we know that

$$\begin{aligned} E_1 = \big\langle \Psi_1|\hat{H}_1|\Psi_1\big\rangle &< \big\langle \Psi_2|\hat{H}_1|\Psi_2\big\rangle \\ &= \big\langle \Psi_2|\hat{H}_2 + \hat{V}_1 - \hat{V}_2|\Psi_2\big\rangle \\ &= E_2 + \int \mathrm{d}^3\mathbf{r}\big[v_1(\mathbf{r}) - v_2(\mathbf{r})\big]\,|\Psi_2(\mathbf{R})|^2 \\ \Rightarrow \quad E_1 &< E_2 + \int \mathrm{d}^3\mathbf{r}\big[v_1(\mathbf{r}) - v_2(\mathbf{r})\big]\rho_0(\mathbf{r}). \end{aligned} \tag{5.5}$$

Exchanging the arbitrary indices $1 \leftrightarrow 2$, gives rise to

$$E_2 < E_1 + \int \mathrm{d}^3\mathbf{r}\big[v_2(\mathbf{r}) - v_1(\mathbf{r})\big]\rho_0(\mathbf{r}), \tag{5.6}$$

and adding Eq. (5.5) from Eq. (5.6) gives finally the contradiction $E_1 + E_2 < E_1 + E_2$ since the integrals differ only by a sign. But this means that clearly, different $\hat{V}_{\text{ext}}$ neccessarily produce differing $\rho_0(\mathbf{r})$ [100].[1]

[1] As a hands-on example of the first theorem in practice, it is instructive to consider the Coulombic external potential of M stationary nuclei, $\hat{V} = -\sum_{A=1}^{M} Z_A/|\mathbf{r} - \mathbf{r}_A|$. In order to uniquely determine the system, we need to know the number of electrons, N, the position of the nuclei, $\mathbf{r}_A$, and their charges, Z_A. The local maxima of the ground state density coincides perfectly with the nuclei positions. Furthermore, the Kato cusp condition states that at the nuclei (c.f. section 3.1) $(\partial\bar{\rho}_0(r_A)/\partial r_A)_{r_A\to 0} = -2Z_A\bar{\rho}_0(0)$, where $\bar{\rho}(r_A)$ denotes the spherical average of density around nucleus A and $r_A = |\mathbf{r} - \mathbf{r}_A|$. This determines Z_A from the ground state density also. Finally, the integral over the density itself gives the number of electrons, $N = \int \mathrm{d}^3\mathbf{r}\rho_0(\mathbf{r})$.

This is known as E. Bright Wilson's observation [102].

The statement of the first Hohenberg-Kohn theorem is represented in diagram form in Fig. 5.1.

The second Hohenberg-Kohn theorem

Since, by the first theorem, the Hamiltonian is determined uniquely by the density, that means the wave function can be considered a functional of the density also, $\Psi[\rho]$. Following the original article by Hohenberg and Kohn[101], we note that this means the kinetic energy operator and the electron-electron interaction operators are also both functionals of the density. These can be combined into the *universal functional*,

$$F[\rho(\mathbf{r})] \equiv \left\langle \Psi | \hat{T} + \hat{W} | \Psi \right\rangle. \tag{5.7}$$

A further energy functional, $E_V[\rho]$, can be defined as

$$E_V[\rho(\mathbf{r})] \equiv F[\rho(\mathbf{r})] + \int \mathrm{d}^3\mathbf{r}\, v(\mathbf{r})\rho(\mathbf{r}), \tag{5.8}$$

where $v(\mathbf{r})$ is the position basis representation of an arbitrary external potential. It is clear that minimizing $E_V[\rho]$ will yield the ground state energy of the system with Hamiltonian $\hat{H} = \hat{T} + \hat{W} + \hat{V}_{\text{ext}}$. In the present section we restrict ourselves to densities representing a system of N electrons, i.e. fixing $\int \mathrm{d}^3\mathbf{r}\, \rho(\mathbf{r}) = N$.

We now consider the energy functional evaluated at some other density, $\rho(\mathbf{r}) \neq \rho_0(\mathbf{r})$, with $\rho_0(\mathbf{r})$ being the corresponding density of the ground state of $\hat{H}$, $\rho_0(\mathbf{r}) = \int |\Psi_0(\mathbf{R})|^2$. The other density, $\rho(\mathbf{r})$, is taken to be the corresponding ground state density of *some other* external potential, $\hat{V}'_{\text{ext}}$. Evaluating the functional gives

$$E_V[\rho(\mathbf{r})] = \left\langle \Psi | \hat{T} + \hat{W} | \Psi \right\rangle + \left\langle \Psi | \hat{V}_{\text{ext}} | \Psi \right\rangle = \left\langle \Psi | \hat{H} | \Psi \right\rangle. \tag{5.9}$$

Having established that the external potential is a functional of the density, this gives us now almost trivially that $E_V[\rho] > E_V[\rho_0]$ for any density that is not the one associated with the true ground state corresponding to $\hat{V}_{\text{ext}}$ [24].

The way forward

Essentially, the two Hohenberg-Kohn theorems say that all properties of a electronic quantum mechanical system is uniquely determined by the ground state density alone. Also, a *universal functional* for the energy in terms of the density, valid for any external potential, exists and it attains a global minimum for exactly the true ground state density $\rho_0(\mathbf{r})$. In the words of Hohenberg and Kohn [101]:

> “If $F[\rho]$ were a known and sufficiently simple functional of ρ, the problem of determining the ground-state energy and density in a given external potential would be rather easy since it requires merely the minimization of a functional of the three-dimensional density function.”
>
> *P. Hohenberg & W. Kohn*

Indeed, all of electronic structure theory would have been *solved* in one fel swoop. Predictably, this is not the case. Determination of the universal functional is anything but trivial. In order to make real progress with the approach of considering the electronic density as the primary variable, we turn now to the framework proposed by Kohn and Sham.

5.2 Kohn-Sham ansatz

In their seminal 1965 paper [23], Kohn and Sham outlined a way to obtain a set of single electron equations in the density that can be solved self-consitently to obtain the total electronic energy. In order to accomplish this, they consider an auxiliary non-interacting system in place of the original one [24].

Kohn and Sham tell us to consider a non-interacting system for which the ground state density is the same as the ground state density for the interacting system in question. The existance of such a non-interacting system of electrons is far for obvious. In fact, determining neccessary constraints on $\hat{V}_{\text{ext}}$ for it to be *non-interacting* v*-representable* is still an open question in the theoretical foundations of density functional theory. For almost all real world problems, it is neccessary to simply assume the existance of a non-interacting system for which the ground state density coincides with that of $\hat{V}_{\text{ext}}$ [24, 103]. The assumption that such a non-interacting system exists constitutes the Kohn-Sham ansatz.

We denote the non-interacting Hamiltonian and the corresponding effective potential by $\hat{H}_s$ and $\hat{V}_s$, with spatial representation $v_s(\mathbf{r})$. Since the electrons in this auqiliary system dont feel the coulombic inter-electron force, we know that the *exact* ground state is represented by a Slater determinant filled with the energetically lowest N single electron solutions of the single electron Schrödinger equation [60]. Note that the Schrödinger equation is by construction separable, since it only involves the kinetic energy operator and a (multiplicative) external potential operator. The one-electron Hamiltonian takes the form

$$\hat{h}_s\psi_i = \left[-\frac{1}{2}\nabla^2 + v_s(\mathbf{r})\right]\psi_i = \varepsilon_i\psi_i, \tag{5.10}$$

with eigenstates ψ_i. Since $\hat{H}_s$ is spin-independent[2] and obviously commutes with $\hat{S}_z$, we may choose the ψ_i to be products of spatial orbitals and the normalized spinor eigenstates of $\hat{S}_z$, $\psi_i(\mathbf{r},\sigma) = \phi_n(\mathbf{r})\chi(\sigma)$. We note that the quantum number i represents both spatial and spin quantum numbers. This means we can write the exact

[2]If the original Hamiltonian is spin-dependent, the effective potential needs to also carry spin-dependency in order to give the required spin-density. This is a complication we will fully ignore here.

ground state wave function and the density as

$$\Psi_0(\mathbf{R}) = \frac{1}{\sqrt{N}} \begin{vmatrix} \psi_1(\mathbf{r}_1) & \psi_2(\mathbf{r}_1) & \dots & \psi_N(\mathbf{r}_1) \\ \psi_1(\mathbf{r}_2) & \psi_2(\mathbf{r}_2) & \dots & \psi_N(\mathbf{r}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(\mathbf{r}_N) & \psi_2(\mathbf{r}_N) & \dots & \psi_N(\mathbf{r}_N) \end{vmatrix}, \tag{5.11}$$

with

$$\rho_0(\mathbf{r}) = \sum_{i=1}^{N} |\psi_i(\mathbf{r})|^2 = \sum_{\sigma} \sum_{n=1}^{N/2} |\phi_n(\mathbf{r})|^2 = 2 \sum_{n=1}^{N/2} |\phi_n(\mathbf{r})|^2. \tag{5.12}$$

In general, the kinetic energy operator as a functional of the density is not known. However, in the auqiliary non-interacting system, we can write down the closed form expression in terms of the ψ_is [60]:

$$T_s[\rho(\mathbf{r})] = -\sum_{i=1}^{N} \left\langle \psi_i \right| - \frac{1}{2} \nabla_i^2 \left| \psi_i \right\rangle. \tag{5.13}$$

Recall that the universal energy functional has the form $F[\rho] = T[\rho] + W[\rho]$, i.e. the sum of the kinetic and the inter-electronic potential functionals. Kohn and Sham now suggest we rewrite $F[\rho]$ in terms of the known non-interacting kinetic energy functional as

$$F[\rho] = T_s[\rho] + J[\rho] + E_{\text{xc}}[\rho], \tag{5.14}$$

where $J[\rho]$ is the classical coulomb interaction functional and $E_{\text{xc}}[\rho]$ is essentially *everything that is left over* [101]. The $J[\rho]$ functional we will shortly see is the functional form of the $\hat{J}$ operator in the Hartree-Fock approximation (see section 4), while $E_{\text{xc}}[\rho]$ is called the exchange-correlation energy and contains the (hopefully small) correction needed for $T_s[\rho]$ in order to make it into $T[\rho]$ and the non-classical part of the electron-electron interaction $\hat{W}$ [24, 60].

The *exact* (recall that the density of the auqiliary system is the same as the original density) energy functional of the fully interacting system can now be written as

$$E[\rho] = T_s[\rho] + J[\rho] + \int \mathrm{d}^3\mathbf{r}\, \rho(\mathbf{r}) v(\mathbf{r}) + E_{\text{xc}}[\rho], \tag{5.15}$$

where $v(\mathbf{r})$ is the position basis representation of $\hat{V}_{\text{ext}}$. For completeness, we state also the explicit form of the exchange-correlation energy

$$E_{\text{xc}}[\rho] = (T[\rho] - T_s[\rho]) + (W[\rho] - J[\rho]), \tag{5.16}$$

where $W[\rho] = J[\rho] - K[\rho]$ is the functional corresponding to the two-body term in the original Hamiltonian.

It may seem like we have made no progress at all: We started off with an unknown functional $F[\rho]$, and after introducing the Kohn-Sham ansatz we are still left with an unknown (albeit different) functional, $E_{\text{xc}}[\rho]$. The exchange-correlation functional, however, may be easier to approximate. Also, we have introduced a separable (by construction) Hamiltonian from which we can extract a set of coupled one-electron Schrödinger equations which we may solve in order to obtain the true ground state density (and in principle all other properties of the system). Note carefully that at this point, no approximations (beyond Born-Oppenheimer and the assumption that the original density was non-interacting v-representible) have been made, and so we may consider the preceding results to be exact. This represents a conceptually major difference between DFT and Hartree-Fock theory: The latter is an approximate set of equations which we solve exactly, while DFT constitutes a set of exact equations which we are forced to solve approximately (because we dont know closed form expressions for E_{xc}).

5.3 The Kohn-Sham equations

In the same way Roothan and Hall introduced orbitals to the Hartree-Fock formalism 4, we now introduce a set of spin-orbitals in the DFT scheme. The single determinantal wave function is constructed from N orthonormal spin-orbitals, $\{\psi_i(\mathbf{r})\}_{i=1}^N$, where we have absorbed the spin quantum number into the label i. In the restricted case, this means that i and $i+1$ are spatially pair-wise identical, with differing spin index. The energy functional can now be written in terms of the orbitals as

$$E[\rho,\psi] = T_s[\psi] + J[\rho] + E_{\text{xc}}[\rho] + \int \mathrm{d}^3\mathbf{r}\, v(\mathbf{r}), \tag{5.17}$$

with

$$\begin{aligned} J[\rho] &= \int\int \mathrm{d}^3\mathbf{r}_1 \mathrm{d}^3\mathbf{r}_2 \, \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{|\mathbf{r}_1-\mathbf{r}_2|} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \int\int \mathrm{d}^3\mathbf{r}_1 \mathrm{d}^3\mathbf{r}_2 \, \frac{|\psi_i(\mathbf{r}_1)|^2|\psi_j(\mathbf{r}_2)|^2}{|\mathbf{r}_1-\mathbf{r}_2|}, \end{aligned} \tag{5.18}$$

and

$$T_s[\psi] = -\frac{1}{2}\sum_{i=1}^{N} \int \mathrm{d}^3\mathbf{r}\, \psi_i^*(\mathbf{r})\nabla^2\psi_i(\mathbf{r}). \tag{5.19}$$

Minimizing this functional—analogous to the Hartree-Fock minimization procedure of chapter 4—yields the Kohn-Sham orbital equations [24]

$$\underbrace{\left[-\frac{\nabla^2}{2} - v(\mathbf{r}) + \int \mathrm{d}^3\mathbf{r}' \, \frac{\rho(\mathbf{r}')}{|\mathbf{r}'-\mathbf{r}|} + \frac{\delta E_{\text{xc}}[\rho]}{\delta\rho(\mathbf{r})}\right]}_{\equiv \hat{F}^{\text{KS}}} \psi_i(\mathbf{r}) = \sum_{j=1}^{N} \varepsilon_{ij}\psi_j(\mathbf{r}), \tag{5.20}$$

where ε is once again a matrix of Lagrange multipliers. Expanding the Kohn-Sham orbitals in a known basis of atomic orbitals $\{\varphi_k(\mathbf{r})\}_{k=1}^{L}$, applying a unitary transformation diagonalizing the Kohn-Sham operator $\hat{F}^{\text{KS}}$, and writing Eq. (5.20) in weak form (c.f. section 4.4.1) gives the **Kohn-Sham equations**

$$\sum_{k=1}^{L} C_{ki}\hat{F}^{\text{KS}}\varphi_k(\mathbf{r}) = \varepsilon_i \sum_{k=1}^{L} C_{ki}\varphi_k(\mathbf{r}), \tag{5.21}$$

or in matrix representation

$$F^{\text{KS}}C = \boldsymbol{\varepsilon} SC. \tag{5.22}$$

In the restricted case, the matrix elements F_{pq}^{KS} are given by

$$F_{pq}^{\text{KS}} = \langle\varphi_p|\hat{h}|\varphi_q\rangle + 2\sum_{rs} D_{rs}\langle pq|\hat{w}|rs\rangle + \int \mathrm{d}^3\mathbf{r}\,\varphi_p(\mathbf{r})\varphi_q(\mathbf{r})v_{\text{xc}}(\mathbf{r}), \tag{5.23}$$

where $\langle\,\cdot\,|\,\cdot\,\rangle$ denotes integration over *spatial* coordinates only and the exchange-correlation *potential* is defined as the functional derivative (see appendix C) of the exchange-correlation energy

$$v_{\text{xc}}(\mathbf{r}) \equiv \frac{\delta E_{\text{xc}}[\rho]}{\delta\rho(\mathbf{r})}. \tag{5.24}$$

5.4 Local density approximation

The efficacy of the DFT scheme now depends sorely on the quality of the basis set $\{\varphi_k\}_{k=1}^{L}$, and more importantly the form of the exchange-correlation energy functional. Since we do not know the exact form, we are forced to parameterize approximations to it. This can be done in various ways, the simplest of which—the local density approximation (LDA)—is the only one we will discuss in this thesis. The LDA is based on the exchange-correlation energy of the uniform electron gas as per the original suggestion of Kohn and Sham [14, 23]. It is usually written down in terms of the exchange-correlation energy per particle, ϵ_{xc},

$$E_{\text{xc}}^{\text{LDA}}[\rho] = \int \mathrm{d}^3\mathbf{r}\,\rho(\mathbf{r})\epsilon_{\text{xc}}\big[\rho(\mathbf{r})\big]. \tag{5.25}$$

The exchange part was shown by Dirac to be [104]

$$E_{\text{x}}^{\text{LDA}}[\rho] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}\int \mathrm{d}^3\mathbf{r}\,\rho^{4/3}. \tag{5.26}$$

For the correlation part, however, no such simple expression exists. Despite this difficulty, highly accurate quantum Monte Carlo calculations on the correlation energy of the free electron gas were performed in 1980 by Ceperley and Alder [105] and

these were subsequently parameterized in several different ways by Vosko, Wilk, and Nussair (VWN) [106]. This resulted in functional forms usable in practical DFT calculations. In the present work we use the VWN5 parameterization (c.f. the original paper), which depends on the *electron gas parameter*,

$$r_s \equiv \left(\frac{3}{4\pi\rho(\mathbf{r}))}\right)^{1/3}, \tag{5.27}$$

and the *spin polarization*

$$\xi(\mathbf{r}) \equiv \frac{\rho_\uparrow(\mathbf{r}) - \rho_\downarrow(\mathbf{r})}{\rho_\uparrow(\mathbf{r}) + \rho_\downarrow(\mathbf{r})}. \tag{5.28}$$

The gas parameter is simply a rescaling of the density, while the spin polarization represents the relative difference in spin-up and spin-down densities at $\mathbf{r}$. If ξ is large, Vosko and co-workers suggest using the "ferromagnetic" exchange

$$\epsilon_{\mathrm{x}}^{\mathrm{F}}(r_s) = -3(2^{1/3})\left(\frac{9}{32\pi^2}\right)^{1/3}\frac{1}{r_s}, \tag{5.29}$$

while for $\xi = 0$, the "paramagnetic"

$$\epsilon_{\mathrm{x}}^{\mathrm{P}}(r_s) = -3\left(\frac{9}{32\pi^2}\right)^{1/3}\frac{1}{r_s} \tag{5.30}$$

is used. In total, the polarized exchange part of E_{xc} takes the form

$$\epsilon_{\mathrm{x}}(r_s,\xi) = \epsilon_{\mathrm{x}}^{\mathrm{P}}(r_s) + \left[\epsilon_{\mathrm{x}}^{\mathrm{F}}(r_s) - \epsilon_{\mathrm{x}}^{\mathrm{P}}(r_s)\right] f(\xi), \tag{5.31}$$

with

$$f(\xi) \equiv \frac{(1+\xi)^{4/3} + (1-\xi)^{4/3} - 2}{2(2^{1/3}-1)}. \tag{5.32}$$

A similar expression is used for the correlation part, with

$$\epsilon_{\mathrm{c}}^{\mathrm{P}}(r_s) + \left[\epsilon_{\mathrm{c}}^{\mathrm{F}}(r_s) - \epsilon_{\mathrm{c}}^{\mathrm{P}}(r_s)\right] f(\xi). \tag{5.33}$$

The ϵ_{c} terms are parametrized by [60]

$$\begin{aligned}\epsilon_{\mathrm{c}}(r_s) = \frac{A}{2}\Bigg\{&\ln\left(\frac{x}{X(x)}\right) + \frac{2b}{Q}\tan^{-1}\left(\frac{Q}{2x+b}\right) \\ &-\frac{bx_0}{X(x_0)}\left[\ln\left(\frac{(x-x_0)^2}{X(x)}\right) + \frac{2(b+2x_0)}{Q}\tan^{-1}\left(\frac{Q}{2x-b}\right)\right]\Bigg\},\end{aligned} \tag{5.34}$$

with

$$\begin{aligned} x &\equiv \sqrt{r_s}, \\ X(x) &\equiv x^2 + bx + c, \\ Q &\equiv \sqrt{4c - b^2}. \end{aligned}$$

For $\xi = 0$ we have $A = 0.0621814$, $x_0 = -0.409286$, $b = 13.0720$, and $c = 42.7198$, while for $\xi \neq 0$ the parameters change, but the functional form of Eq. (5.34) remains the same.

5.5 Numerical integration grids

Since the integral of the exchange-correlation potential over the density is rarely (if ever) calculable analytically, we are forced to resort to numerical integration schemes in order to evaluate it. In the one dimensional case, strong and robust integration schemes based on gaussian quadrature integrate any sufficiently *smooth* function to high accuracy using a modest number of integration points [16]. However, in multiple dimensions, the problem of numerically integrating a multi-variable function is considerably more challenging. Although the philosophy behind is unchanged, the actual application required a lot more thought. This is in large part because all the nuclei represent singularities which hinder the sucessful application of products of gaussian quadrature rules [107]. Before we begin discussing integrals in density functional theory, we refer the reader to a short introduction to numerical integration in section B of the appendix.

The integrals of interest within the framework of density functional theory are integrals over the electronic density. Since we know a priori that the density falls off exponentially in the long range limit (c.f. section 3.1), this reduces to a locally contained integral in some finite and *small* region in space close to the nuclei. Since the potentials are in general functions of the density and its derivatives (which of course also falls off exponentially) we are guaranteed that the exchange-correlation potentials also vanishes exponentially as we move far away from the nuclei.

5.5.1 Simple spherical grid

When dealing with a single nucleus, we may simply choose some cut-off radius and place integration points either linearly or in some other fashion along the radius up to this cut-off radius. Since we are primarily interested in integrands which fall off exponentially with the radius, having more points closer to the nucleus will yield a better approximation with fewer points. A simple example is logarithmically spaced points along the radius. At every such radius, we place k^2 angular points, m linearly spaced points in the polar and the azimuthal angle respectively.

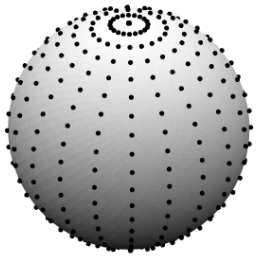

Figure 5.2: Example of a simple spherical shell grid for a single radius r. The full grid employs m total such shell grids, one for each of the logarithmically spaced values r_i. Note the relatively higher density at the pole. This example grid uses 20 linearly spaced points in both the polar and azimuthal angles, θ and ϕ for a total of 400 points.

The weight for each of the points is simply the volume element of the sphere at radius r_i, polar angle θ_j, and azimuthal angle ϕ_k. In addition we need to multiply by the finite change in r, θ, and ϕ. In total, we recover the familiar spherical polar volume element [73] with the addition of $\Delta r \Delta\theta \Delta\phi$,

$$\begin{aligned} w_{ijk} &= r_i^2 \sin\theta_j (\underbrace{r_i - r_{i-1}}_{\Delta r_i})(\underbrace{\theta_j - \theta_{j-1}}_{\Delta\theta_j})(\underbrace{\phi_k - \phi_{k-1}}_{\Delta\phi_k}) \\ &= r_i^2 \sin\theta_j \Delta r_i \left(\frac{\pi}{m-1}\right)\left(\frac{2\pi}{m-1}\right) \\ &= \frac{2\pi r_i^2 \sin\theta_j \Delta r_i}{(m-1)^2}. \end{aligned} \tag{5.35}$$

Under this approximation, the numerical integral over a functional of the density, $F[\rho]$, is given by (in terms of cartesian coordinates)

$$\int F[\rho]\,\mathrm{d}^3\mathbf{r} \approx \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} w_{ijk} F\left[\rho(x_\alpha, y_\beta, z_\gamma)\right], \tag{5.36}$$

with $x_\alpha = r_i \sin\theta_j \cos\phi_k$, $y_\beta = r_i \sin\theta_j \sin\phi_k$, and $z_\gamma = r_i \cos\theta_j$ being the normal transformation form spherical polar $\rightarrow$ cartesian coordinates.

The cut-off radius may be chosen to be the radius at which the most diffuse basis function has fallen to below some pre-assigned threshold value, ε. With cartesian gaussian basis functions, $\{\chi_b(\mathbf{r})\}_{b=1}^{M} = \mathcal{B}$, the cut-off takes the value

$$r_{\text{cut-off}} = \sqrt{\frac{\log\varepsilon}{\min\{|\alpha_b| : \chi_b \in \mathcal{B}\}}}. \tag{5.37}$$

An example of such a grid is shown in Fig. 5.2. Only the shell at $r_i = 1$ is shown, the full grid consists of m such shells at logarithmically spaced r_is.

5.5.2 Efficiency of angular grids and the *product Gaussian quadrature formula*

The naive grid described in the previous section *works*, but is far from optimal. In order to derive a quadrature rule more suited to our purpose, we turn our attention to the general problem of integrating a function $f(\theta, \phi)$ on the surface of a unit sphere $\mathbb{S}^2 = \{\mathbf{r} \in \mathbb{R}^3 : |\mathbf{r}| = 1\}$. Let us now ask: Is there an *optimal* way to distribute points and find associated weights such that the approximation

$$I = \int_0^{\pi} \mathrm{d}\theta \int_0^{2\pi} \mathrm{d}\phi \, f(\theta, \phi) \approx \sum_{i=1}^{n} w_i f(\theta_i, \phi_i) \tag{5.38}$$

is the best possible approximation for a given number of points n? In order to make the question more precise, we note that any such function $f : \mathbb{S}^2 \to \mathbb{R}$ may be expressed in terms of the spherical harmonics (since they form a complete basis for the square integrable functions on $\mathbb{S}^2$, $L^2(\mathbb{S}^2)$ [72]. So let us rephrase the question as: Is there a way to distribute points and find associated weights such that the approximation in Eq. (5.38) integrates *exactly* any linear combination of spherical harmonics with degree up to L?

Before we go on to answer this question, let us define more rigorously what we mean by an expansion in terms of spherical harmonics. The spherical harmonics themselves are given by [73]

$$Y_l^m(\theta, \phi) = \frac{1}{\sqrt{2\pi}} P_l^m(\cos\theta) e^{im\phi}, \tag{5.39}$$

with $-l \leq m \leq l$ and $l, m \in \mathbb{N}$. The polynomials P_l^m are the *normalized associated Legendre polynomials*.[3] Expanding f in terms of these harmonics constitutes finding c_{lm},

$$\tilde{f}(\theta, \phi) = \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{lm} Y_l^m(\theta, \phi) \tag{5.42}$$

such that $e = \|\tilde{f} - f\|$ is minimized [81] (in the sense that the "error" e is orthogonal to the finite dimensional subspace of $L^2(\mathbb{S}^2)$ spanned by the harmonics up to

[3]The associated Legendre polynomials are solutions to the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[\left(1-x^2\right)\frac{\mathrm{d}}{\mathrm{d}x}P_l^m(x)\right] + \left[l(l+1) - \frac{m^2}{1-x^2}\right] P_l^m(x) = 0, \tag{5.40}$$

which attains non-zero and non-singular solutions in $[-1, 1]$ if and only if m and l are both integers, with $l \geq m$ [73]. An explicit formula for the polynomials is for example

$$P_l^m(x) = \frac{(-1)^m}{2^l l!}\sqrt{(1-x^2)^m}\frac{\mathrm{d}^{l+m}}{\mathrm{d}x^{l+m}}\left(x^2-1\right)^l. \tag{5.41}$$

and including degree L). This is known as the Galerkin method, and the governing equation for the coefficients is the inner product

$$c_{lm} = \int_{\mathbb{S}^2} \mathrm{d}\theta \mathrm{d}\phi \, f(\theta, \phi) Y_l^m(\theta, \phi). \tag{5.43}$$

The decay rate of the coefficients with increasing l determines how well $\tilde{f}$ approximates f, i.e. the convergence rate of the spherical harmonic expansion [108].

Dealing with the problem of integrating functions across the surface of a sphere, McLaren defined in 1963 a measure of the effectiveness of a quadrature rule by how high order L the rule would integrate *exactly* using K independent arbitrary variables used. In terms of the number of points used, the McLaren efficiency is defined as [109]

$$E = \frac{\text{Number of spherical harmonics up to and including order } L}{\text{Number of variables associated with } N \text{ points}} = \frac{(L+1)^2}{3N}. \tag{5.44}$$

The total number of independent variables is 3 for each point: two angles and a weight. The total number of spherical harmonics up to and including order L is

$$\begin{aligned} \sum_{l=0}^{L} \sum_{m=-l}^{l} 1 &= \sum_{l=0}^{L} (2l+1) = \sum_{l=0}^{L} 2l + \sum_{l=0}^{L} 1 \\ &= L(L+1) + (L+1) = (L+1)(L+1) = (L+1)^2. \end{aligned} \tag{5.45}$$

In general we expect $E \leq 1$, although in rare cases $E > 1$ [72].

Let us now consider the efficacy of the common practice of applying a Newton-Cotes quadrature rule (for example the trapezoidal rule) in ϕ and a Gauss-Legendre scheme for the θ integral. An equally spaced grid of M points ensures exact integration of

$$\int_0^{2\pi} \mathrm{d}\phi \, e^{im\phi}, \tag{5.46}$$

for all $m \leq M$, whereas a gaussian quadrature rule using the Legendre polynomials gives exact integration of the associated Legendre polynomials

$$\int_0^{\pi} \mathrm{d}\theta \, P_l^m(\cos\theta), \tag{5.47}$$

for $l \leq M$ if $(M+1)/2$ points are used [108]. In total, the "product Gaussian quadrature formula" takes the form [72]

$$I \approx \sum_{i=1}^{M} w_i \sum_{j=0}^{\frac{L+1}{2}} w_j f(\theta_i, \phi_j), \tag{5.48}$$

and the efficiency is $E = 2/3$ [109].

Figure 5.3: Example of the product gaussian grid (left) and the Lebedev grid (right). The former employs 20 integration points in the Legendre quadrature and 20 points in the equidistant ϕ grid, for a total of 400 points. The latter is a 25th order grid, with a total of 266 points. Note the cluttering of points at the pole for the product gaussian grid. Adapted from a similar figure in [108].

5.5.3 Lebedev quadrature

Since the product Gaussian quadrature fails to attain $E \approx 1$, it is natural to consider it's flaws and look for an improved scheme. An obvious weakness of the product scheme is the clustering of integration points at the poles. Because the spacing in the polar angle θ is linear and there are a constant number of azimuthal angle ϕ points for every θ_i, the distance along the surface of the unit sphere between adjacent angular points ϕ_j and ϕ_{j+1} approaches zero as $\theta \to 0$ or π. Intuitively, the points close to the poles should not be more important than the points at the equator, since we are fully allowed to rotate the coordinate system 90° along the polar angle—essentially switching the poles and the equator—without affecting the value of the integral.

In fact, this very observation is crucial in order to make progress: Any rotation or inversion (all points reversed through a given plane) that leaves the sphere invariant should also leave the quadrature invariant. This observation is the contents of a theorem due to Sobolev (for an english translation of his seminal 1962 paper, see e.g. [110]). Essentially, Sobolev states that given a qudrature rule to integrate a spherical harmonic monomial[4] f on $\mathbb{S}^2$, $I(f)$, the rule must neccessarily give the same result as the quadrature rule resulting from first applying a rotation or inversion,

$$I(f) = I(\gamma[f]), \tag{5.49}$$

if f is invariant under γ. Here, $\gamma \in G$ and G denotes the discrete symmetry group of the sphere. The theorem further states crucially that in order for I to integrate all spherical harmonic monomials (and thus also all linear combinations of harmonics, such as the approximation $\tilde{f}$ from Eq. (5.42)) it is sufficient to only demand I integrate exactly all the monomoials which are invariant under $\gamma \in G$ [72].

[4] I.e. $f(\theta, \phi) = Y_l^m(\theta, \phi)$ for some single unique l and m.

In the 1970s, Lebedev (who was a Ph.D. student under Sobolev) constructed a class of invariant quadratures based on this idea by working out the invariant spherical harmonics and solving the resulting system of non-linear equations for the points and weights [108, 111]. Lebedev published multiple articles deriving higher and higher order methods, culimnating in quadrature rules of degree over a hundred.

An example of a Lebedev grid is shown in Fig. 5.3, alongside a gaussian product grid. We note the difference in the cluttering of the points towards the poles for the gaussian grid. Lebedev's grid attains efficiency $E = 1$, and as such we expect about equal integration results as when using a gaussian product grid of $3/2$ as many points [108]. This means that the 400 point gaussian quadrature on the left and the 266 point Lebedev quadrature on the right should in practice yield the exact same precision. We note that in using the Lebedev grid, we are implicitly expanding the integrand in terms of spherical harmonics which means the precision of the quadrature depends heavily on the relatively rapid convergence of the spherical harmonics coefficients c_{lm} of Eq. (5.5) [107].

5.5.4 Complete molecular grids, Voronoi and Wigner-Seitz partitioning

In order to expand the previous spherical shell grid to a full integral over the entire molecular volume, we need to pair it with a quadrature rule for the radius. For the case of a single atom we may simply pair the Lebedev grid with a Gauss-Laguerre scheme[5] for the radial integral.

However, the polyatomic case is considerably harder to deal with because of the aforementioned problem of the nuclei representing singularities which ruin the favorably low error scaling and convergence properties enjoyed by gaussian quadrature applied to more well-behaved integrands. Before we dive into the thick of things, we need to first introduce the notion of Voronoi partitioning.

Assume we are given a set of K points $\{\mathbf{r}_i\}_{i=1}^K$. We may partition any volume enclosing these points (e.g. all of $\mathbb{R}^3$) into K separate regions in a unique way such that any point in region j is closer to $\mathbf{r}_j$ than to any other $\mathbf{r}_i$, $i = 1, 2, \ldots, j-1, j+1, \ldots, K$. Even if the idea dates back *at least* as far as the 17th century and

[5]The (generalized) Laguerre polynomials are the solutions to the differential equation

$$x\frac{\mathrm{d}^2y(x)}{\mathrm{d}x^2} + (1+\alpha-x)\frac{\mathrm{d}y(x)}{\mathrm{d}x} + ny(x) = 0, \tag{5.50}$$

with n a non-negative integer and α an arbitrary real constant [73]. An explicit expression for the polynomials themselves can be found by the so-called Rodrigues formula:

$$L_n(x) = x^{-\alpha}\frac{1}{n!}\left(\frac{\mathrm{d}}{\mathrm{d}x} - 1\right)^n x^{n+\alpha}. \tag{5.51}$$

The weight function $W(x)$ for the Laguerre polynomials in the setting of gaussian quadrature takes the form $W(x) = x^\alpha e^{-x}$.

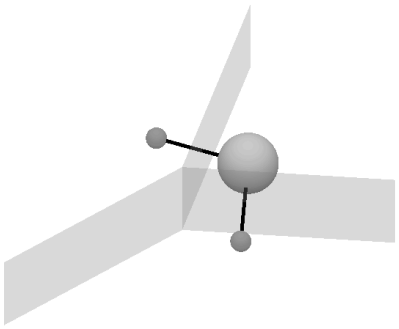

Figure 5.4: Example of a Voronoi partitioning for the water molecule H_2O. Since this is a planar molecule, the Voronoi partitioning is strictly two-dimensional, with trivial vertical components. All surfaces stretch out to infinity, but is truncated in the plot at ± 1 and ± 5 for the vertical and horizontal directions, respectively. The polar covalent O – H bonds are marked with black lines.

Descartes, the procedure and the resulting structure takes its name from Russian-Ukrainian mathematician G. Voronoi: it is called **Voronoi partitioning** or a Voronoi diagram (or sometimes also Dirichlet tesselation, after German mathematician P.G.L. Dirichlet) [112]. For the special case of a regular crystalline lattice of atoms in three-dimensional space is considered the Voronoi partitioning is known as Wigner-Seitz cells [107]. An example of a Voronoi partitioning of a molecular volume is shown in Fig. 5.4, where the boundaries between the Voronoi cells for the H_{20} molecule are drawn as gray walls.

In the following we will describe Boerrigter, Velde, and Baerends' integration scheme for polyatomic molecules which employ Lebedev spherical grids and handle the radial integral by partitioning the molecular volume into Voronoi cells [107]. We will call the resulting quadrature rule for the **Voronoi grid**.

Boerrigter and co-workers note that much of the difficulty in computing the integrals for such multi-atomic molecular systems stem from the singularities inherent in the point-like nuclei. A relatively straight forward way to circumvent these divergencies is to integrate in spherical polar coordinates centered *at the nuclei*. Since the r^2 factor of the Jacobian associated with the coordinate transformation supresses the $1/r$ factor of the Coulomb interaction, the resulting quadrature rule is well-behaved at and around $r \to 0$. The proposed algorithm divides the molecular volume into a set of Voronoi cells for each atom. Next, non-overlapping spheres are constructed by fixing the largest atom-centered spheres that can be contained in each cell. This divides the entire volume into a set of spheres (on which the integration is relatively easy), and an "interstitial" region [107].

The integral over the atomic spheres is done by a Lebedev and Gauss-Legendre product (possibly with the Legendre grid split into multiple sub-regions for heavy atoms, in order to ensure enough emphasis is given to the tight core) [113].

The integral over the interstitial region is split into quadrangular and triangular pyramids, as shown in Fig. 5.5. This essentially means we are integrating over what is left of the Voronoi partition, after we have already dealt with the atomic spheres. The base of the pyramids can in general be any polygon, but it is always possible to split a

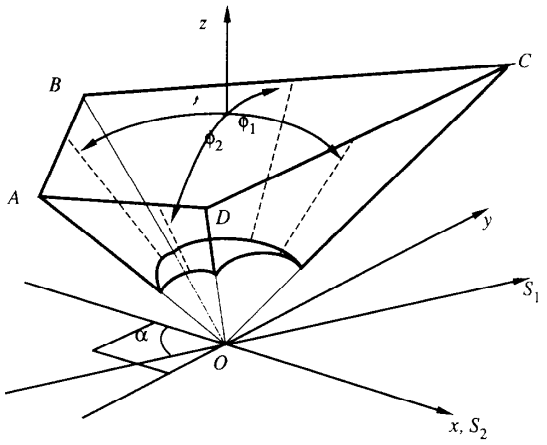

Figure 5.5: Illustration of a triangular interstitial region outside of the atomic sphere. The base of the quadrangular pyramid, $ABCD$, is the edge of the Voronoi cell between the atom at O and the neighbouring atom. The integral is taken over cartesian coordinates in the basal plane and radially from the atomic sphere surface $z_0(x, y)$ to (x, y, z_B) at the base. This is equivalent to the beam of solid angle $\Delta\Omega$ with corresponding ϕ_1 and ϕ_2 associated with x and y. The z coordinate at the base is denoted z_B. Figure taken from [113].

n-vertex polygon into a quadrangle and a $n-2$-vertex polygon and iteratively reduce the number of sides of the polygon while generating more and more quadrangles. Depending on if the starting side number n of the original n-gon was even or odd, the last partition ends up as a quadrangle or a triangle.

In order to map the general truncated (atomic sphere removed) pyramid onto the unit cube, three parameters u, v, and w are defined. A general point on the base is given in terms of u and v as

$$\mathbf{Q}(u, v) = (1-u)(1-v)\mathbf{A} + u(1-v)\mathbf{B} + uv\mathbf{C} + (1-u)v\mathbf{D}, \tag{5.52}$$

with $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ being the vertices of the base [107]. The line S_1 is defined as the common line of the planes ABO and CDO, while S_2 is the common line of ADO and CBO [113]. The angles ϕ_1 and ϕ_2 are aligned $45°$ w.r.t. the x and y-axes.

We will leave out the rest of the technicalities here. The important take-away is that the Voronoi grid always integrates over volumes containing nuclei in terms of spherical polar coordinates centered at said nuclei. This ensures the Lebedev-Legendre quadrature is well-behaved for the atomic spheres. For the complete expressions for the quadrature points and weights for both the atomic spheres and the interstitial regions, see [113].

5.6 Becke grid

The Voronoi grid of Boerrigter and co-workers is unfortunately not very well suited for complete automation. A complication which was largely ignored in the previous section is that in order for optimal integral convergence, further splitting of the radial grids is needed. E.g. a partitioning of the atomic sphere into two or even three separate regions, say $(0, 0.1.)$, $(0.1, 0.3)$, and $(0.3, 1)$ may be needed in order to obtain acceptable results [107]. It is however hard to come up with *general* guide lines for where exactly such splitting should occur, given arbitrary atomic charges and ge-

ometries. We will therefore end our discussion of numerical integration with the grid due to Becke [114].

He suggests a scheme based on the same philosophy, dividing the molecular volume into Voronoi cells. However, instead of constructing "hard atomic spheres" and treating the left-over interstitial regions separately, he introduces a "fuzzy" cellular partitioning, i.e. a Voronoi tessellation with smooth and continous transitions between cells.

Becke starts out with a two-center coordinate system known as *confocal elliptical coordinates*, (λ, μ, ϕ), where the ϕ coordinate denotes the angle about the internuclear axis, while λ and μ are defined as

$$\lambda = \frac{r_1 + r_2}{R_{12}}, \quad \text{and} \quad \mu = \frac{r_1 - r_2}{R_{12}}, \tag{5.53}$$

where r_1, r_2, and R_{12} denote distance to nucleus one, distance to nucleus two, and the internuclear separation, respectively [114]. The confocal coordinate system corresponds to representing positions in terms of confocal (i.e. all sharing focal points) quadrics, generalizations of the conic sections: ellipses, hyperbolas, and parabolas. The μ coordinate represents the inter-nuclear separation, with the surface at $\mu = 0$ being the edge of the Voronoi cells of nuclei 1 and 2, i.e. the surface at which $r_1 = r_2$. The limit $\mu \to -1$ ($\mu \to +1$) correspond to the ray extending from the midpoint, through nucleus 1 (nucleus 2), and out to infinity [73].

For nucleus i, taking the confocal elliptical coordinates relative to all other nuclei j gives a set of coordinate systems $\{(\lambda_{ij}, \mu_{ij}, \phi_{ij})\}_{j=1, j\neq i}^{M}$, with which we can express the Voronoi cells as step functions of μ_{ij}:

$$s(\mu_{ij}) = \begin{cases} 1 & \text{for} \quad -1 \leq \mu_{ij} \leq 0 \\ 0 & \text{for} \quad 0 < \mu_{ij} \leq 1 \end{cases} \tag{5.54}$$

with $P_i(\mathbf{r}) = \prod_{j\neq i} s(\mu_{ij})$ being the *cell function* for nuclei i which takes the value 1 whenever $\mathbf{r}$ is inside the Voronoi cell for nuclei i, and 0 otherwise [114]. Becke now suggests replacing the sharp step function by a less sharp "cutoff function," yielding the aforementioned "fuzzy" Voronoi partitioning.

The cutoff function is not unique (the only neccessary properties being $f(-1) = -1$, $f(+1) = +1$, and vanishing derivatives at both these points), so Becke simply chooses a simple one: $s_n(\mu) = [1 - p^n(\mu)]/2$ with $p(\mu) = 3\mu/2 - \mu^3/2$ and $p^n(\mu)$ being the n times iterated p. The iterations $p^n(\mu)$ yield progressively sharper cutoff profiles $p^n \equiv p \circ p \circ p \circ \cdots \circ p$ (n-times), $p^n(\mathbf{r}) = p\{p(\ldots p[\mathbf{r}]\ldots)\}$, with $\lim_{n\to\infty} s_n$ recovering Eq. (5.54) [115]. Examples of the iteration for various values of n is shown in Fig. 5.5.

Next, for each nucleus assign to each point in the (entire) molecular volume a relative weight $w_m(\mathbf{r})$, such that for any given $\mathbf{r}$, the following holds

$$\sum_{m=1}^{M} w_m(\mathbf{r}) = 1. \tag{5.55}$$

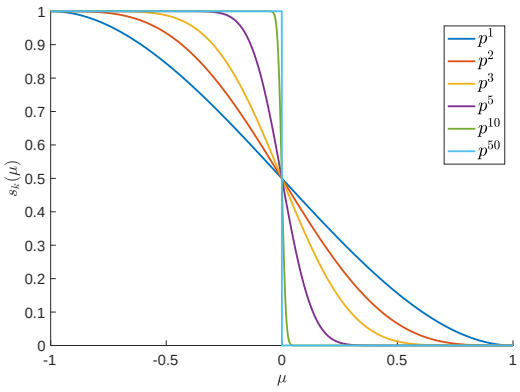

Figure 5.6: The Becke cutoff profile as a function of the confocally elliptical inter-nuclear separation coordinate, μ, for various values of iteration, k. We note that in the limit of very high k, the step function of Eq. (5.54) is reproduced. Adapted from a similar figure in [114].

We demand that $w_m(\mathbf{r})$ be assigned in such a way as to take the value unity in the viscinity of nucleus m, and vanish in the viscinity of every other nucleus. Using the previously described cutoff functions (Becke suggests a value of $n = 3$ iterations of the polynomial p), we can assign $w_m(\mathbf{r})$ as [114]

$$w_n(\mathbf{r}) = \frac{P_n(\mathbf{r})}{\sum_{m=1}^{M} P_m(\mathbf{r})}. \tag{5.56}$$

Having defined the relative weight of each point in space, the only thing that remains to be done is to define around each nucleus a local spherical polar coordinate system (r_m, θ_m, ϕ_m) and we can rewrite the integral any (scalar-valued) real-valued function $F(\mathbf{r})$ as a sum over *single center integrals* $F_m(\mathbf{r})$ as

$$I = \int F(\mathbf{r})\, \mathrm{d}^3\mathbf{r} \tag{5.57}$$

$$= \sum_{m=1}^{M} \int F_m(\mathbf{r})\, \mathrm{d}^3\mathbf{r}. \tag{5.58}$$

Each $F_m(\mathbf{r})$ is the single center integrand defined by the weight function Eq. (5.56),

$$F_m(\mathbf{r}) \equiv w_m(\mathbf{r}) F(\mathbf{r}), \tag{5.59}$$

and we note that since $\sum_{m=1}^{M} w_m(\mathbf{r}) = 1$, the sum over all $F_m(\mathbf{r})$ is equal to $F(\mathbf{r})$ for every point $\mathbf{r}$, and so the two integrals of Eqs. (5.57) and (5.58) are equal.

In short, the scheme proposed by Becke maps the problem of integrating across the whole molecular volume onto a series of much simpler *single center integrals* over the viscinity of each nucleus. Since the weight function of nucleus m vanishes close to any other nucleus, we can just integrate over a simple spherical Lebedev grid (with the r integral handled by a Gauss-Chebychev quadrature of the second kind, as proposed by Becke and co-workers) for each nucleus without having to worry about the singularities at the other nucleonic centers [116]. Since we can create a

spherical grid around each nucleus with cutoff radii dependent on the orbitals of the basis centered around that nucleus, we know that we always include *enough* of the surrounding space to obtain sufficient precision in the integral. This is in contrast to the scheme proposed by Boerrigter and co-workers [107], described in previously in section 5.5.4. In that scheme we are guaranteed only to *directly* hit the volume enclosed by the largest sphere that we can totally enclose within each Voronoi cell, and subsequently have to handle in some intricate and complicated way the "left-over volume" that is inside each Voronoi cell but not inside this sphere.

Chapter 6

Variational Monte Carlo

A large collection of computational methods exist which attempt to solve the Schrödinger equation via the use of stochastic *Monte Carlo* methods. These are collectively known as *Quantum Monte Carlo methods*, and the (arguably) simplest such method is known as **Variational Monte Carlo** (VMC). The VMC scheme attempts to directly evaluate the integral governing the ground state energy expectation value of a quantum mechanical system,

$$E_0 = \langle \Psi | \hat{H} | \Psi \rangle = \frac{\int \mathrm{d}^{3N}\mathbf{R}\, \Psi^*(\mathbf{R}) \hat{H}(\mathbf{R}) \Psi(\mathbf{R})}{\int \mathrm{d}^{3N}\mathbf{R}\, \Psi^*(\mathbf{R}) \Psi(\mathbf{R})}, \tag{6.1}$$

with $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$ being all electronic coordinates. It is fairly obvious from the name that said integral is evaluated using Monte Carlo integration. Any other quantum mechanical quantity of interest can be expressed in terms of the expectation value of an operator à la Eq. (6.1), which means we can essentially formulate all of electronic structure theory in terms of such high-dimensional integrals [68]. Since Monte Carlo methods are well suited to solving such integrals (for which grid-based methods fail spectacularly) the matching of quantum mechanics and Monte Carlo integration has widespread applicability [17].

Of course, we do not a priori know the form of the ground state wave function, $\Psi(\mathbf{R})$, and in electronic structure problems we have to construct an ansatz wave function by filling a (linear combination of) Slater determinant(s) with spin-orbitals from some chosen basis set. However, in contrast with the previously described Hartree-Fock and (Kohn-Sham) Density Functional methods, VMC does not in general require the use of spin-orbitals for which one-, and two-electron Coulombic interaction integrals can be readily calculated [69]. Whereas we are essentially forced into using combinations of gaussian basis functions under the former schemes, the Variational Monte Carlo method offers much more freedom in the choices of orbital basis. Essentially the only requirement on the orbital basis functions is that we need to be able to evaluate the orbitals and their second derivatives realatively efficiently.

The Variational Monte Carlo method is an explicitly correlated one, meaning dynamic electron-electron correlation is taken into account. Unlike the Hartree-Fock

formalism which treats (opposite spin) electron-electron correlation purely in terms of a mean-field approximation, the electrons under the VMC formalism interact *instanteneously*. Electrons are at all times surrounded by "correlation holes" where the probability of finding other electrons vanish. Introducing explicit many-body correlation terms in Hartre-Fock and post Hartree-Fock methods lead to the introduction of molecular integrals which are significantly harder to solve numerically than otherwise ignoring such terms. For this reason, the vast majority of such work is done using Slater detemrinants filled with independent-particle orbitals. However, due to the flexibility of the Monte Carlo integration scheme, we may (and should) relatively easily include correlation terms explicitly in the VMC wave function [117].

In the following, we will present the fundamentals of Monte Carlo integration and its application to the electronic quantum mechanical problem. But first of all we will derive the Metropolis algorithm which we will employ to sample the configuration space of our N-electron system according to the wave function squared $|\Psi(\mathbf{R})|^2$ (the probability density).

6.1 The Metropolis algorithm

The Metropolis algorithm, originally proposed by Metropolis and co-workers [118] and later generalized by Hastings [119] is a method for sampling probability distributions which may be impossible to sample directly[1] [69]. The Metropolis-Hastings algorithm generates a Markov chain of random samples distributed according to the PDF in question.

6.1.1 Markov chains, detailed balance and ergodicity

A Markov chain is a sequence of stochastic variables $\{X_i\}_{i=1}^n$ for which step $i+1$ depends soley on step i, meaning the process has no "memory." In other words, the probability of stepping from state X_k to state X_l depends only on the states k and l, not on *how* it got to X_k, [17, 121]

$$W\left(X_{i+1}; i+1 | X_i; i | X_{i-1}; i-1 | \ldots | X_0; 0\right) = W\left(X_{i+1}; i+1 | X_i; i\right). \tag{6.2}$$

Given the state X_i, the step to the next configuration—the state X_{i+1}—is governed by the stepping probability $W(\,\cdot\,; i+1|X_i; i)$. Even though the stepping probability has no *memory*, the probability of finding the chain in configuration Y at step i is implicitly dependent on the starting configuration, X_0. The *Markov assumption* means we can write the conditional probability of the chain going from X_0, through X_1, X_2,

[1] Direct sampling in this context means e.g. *inverse transform sampling*, which neccessitate inverting the cumulative distribution associated with the probability density function (PDF) [120]. For complicated distributions, this is often either impossible of unfeasible [69].

$\ldots$, X_i, and finally to Y (at step $i+1$) as a simple product [122]

$$P^{(i)}(Y|X_i|X_{i-1}|\ldots|X_0) = W(Y;i+1|X_i;i)W(X_i;i|X_{i-1};i-1)\ldots W(X_1;1|X_0;0). \tag{6.3}$$

In order for the Markov chain to be of much use to us, we need to demand that given *any* starting point, after sufficient time has passed the chain must gradually "forget" about its initial value. In other words, $P^{(i)}(\,\cdot\,|X_0)$ must converge to some *unique stationary* distribution independent of X_0 for large i [121]. Neccessary and sufficient conditions for this to hold will be derived shortly.

Markov chain as a random walk

We can visualize the evolution of the Markov chain as a *walker*, taking steps around in the configuration space according to the stepping probability. The attributes of the walker define completely the state of the system, X, which in the present work is the coordinate configuration of all electrons $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$. Standing at position X in the configuration space, the probability of the walker of stepping to configuration Y in the next step is then given by the stepping probability $W(Y|X)$.

If the configuration space is discrete, i.e. only a finite number K of configurations are possible, then the stepping probability is a $K \times K$ matrix, with the configurations of the system being represented by K-dimensional unit vectors. The states of the walker then become K-dimensional vectors, with the elements representing the probability of the system being in any given configuration. For a system in which the configurations take continous values, the stepping probability is an operator on an infinite dimensional vector space. We will ignore any possible subtle complications introduced by considering the continous integrals as opposed to discrete sums, and refer to W as a matrix also in the continous case.

It is important to note that even though the configuration space consists of quantities with continous ranges, the *sample path* of the Markov chain is still discrete [122]. In other words, stepping from Markov step i to the next step $i+1$ involves some finite (not infinitesimal) change in the configuration.[2] Since the stepping probability is in fact a *probability*, the following equation must hold

$$\sum_{Y\in\mathcal{X}} W(Y;i+1|X;i) = 1, \quad \text{or} \quad \int_{\mathcal{X}} \mathrm{d}Y\, W(Y;i+1|X;i) = 1, \tag{6.4}$$

in the discrete and continous cases, respectively [17]. The set of all possible configurations of the system is here denoted $\mathcal{X}$. Eq. (6.4) essentially states the when the walker is situated at configuration X, a Markov step means the walker either moves or stays put. In addition, non-negativity $W(Y|X) \geq 0$ must hold [69]. This means W is a *stochastic matrix*.

[2] Infinitesimal changes are of course in general also allowed. This ensures for example that there is a non-zero chance of the system remaining in place at the $i \to i+1$ step.

A related but somewhat more useful relation can be found when considering a subsequent step from one of all the possible intermediate configurations Y, to some state Z. Summing (integrating) over the intermediate state yields the Chapman-Kolmogorov equation, [122]

$$W(Z;i+2|X;i) = \sum_{Y\in\mathcal{X}} W(Z;i+2|Y;i+1)W(Y;i+1|X;i), \quad \text{or} \tag{6.5}$$

$$W(Z;i+2|X;i) = \int_{\mathcal{X}} \mathrm{d}Y\, W(Z;i+2|Y;i+1)W(Y;i+1|X;i). \tag{6.6}$$

This is also sometimes known as the Einstein–Smoluchowski–Kolmogorov–Chapman or the Chapman-Einstein-Enskog-Kolmogorov relation [68, 123].[3]

In the following, we drop the temporal i-indices on the stepping probability which we assume to not depend on the *Markov time*. In addition, without any loss of generality, we split it into two parts: a proposal probability and a corresponding acceptance probability. We write thus

$$W(Y;i+1|X;i) = W(Y|X) \equiv T(Y \leftarrow X)A(Y \leftarrow X), \tag{6.7}$$

where T and A are proposal and acceptance probabilities, respectively. When the temporal indices are omitted, a single step is always assumed. It turns out that we can express the condition that resulting probability density converge to some *unique* distirbution in terms of the Chapman-Kolmogorov relation of Eq. (6.6). Since the probability of the walker being in configuration X at time i is $P^{(i)}(X)$, the total probability of being in state Y at time $i+1$ is the sum of probabilities of *being in* X *and stepping into* Y (for all possible state X) plus the probability of *being in* Y *and rejecting a suggested move out to any other state* X [124]:

$$\begin{aligned}
P^{(i+1)}(Y) &= \int_{\mathcal{X}} \mathrm{d}X\, \left[P^{(i)}(X)W(Y|X) + P^{(i)}(Y)\neg W(X|Y)\right] \\
&= \int_{\mathcal{X}} \mathrm{d}X\, \Big[P^{(i)}(X)T(Y \leftarrow X)A(Y \leftarrow X) + \\
&\qquad\qquad P^{(i)}(Y)T(X \leftarrow Y)\left\{1 - A(X \leftarrow Y)\right\}\Big]. \\
&= \int_{\mathcal{X}} \mathrm{d}X\, \Big[P^{(i)}(X)T(Y \leftarrow X)A(Y \leftarrow X) + P^{(i)}(Y)T(X \leftarrow Y) - \\
&\qquad\qquad P^{(i)}(Y)T(X \leftarrow Y)A(X \leftarrow Y)\Big].
\end{aligned} \tag{6.8}$$

Since the integral of $T(X \leftarrow Y)$ over all possible X neccessarily must be unity, the

[3] Or apparently pretty much *any* combination of two or more of the names Chapman, Kolmogorov Einstein, Smoluchowski, and Enskog; depending on who you ask.

middle term is simply $P^{(i)}(Y)$ [68]. This means we can rewrite Eq. (6.8) as

$$P^{(i+1)}(Y) - P^{(i)}(Y) = \int_{\mathcal{X}} \mathrm{d}X \Big[P^{(i)}(X) T(Y \leftarrow X) A(Y \leftarrow X) - P^{(i)}(Y) T(X \leftarrow Y) A(X \leftarrow Y) \Big], \tag{6.9}$$

and the condition that once equilibrium is reached, the Markov chain cannot exit the equilibirium again can now be stated as $P^{(i+1)}(Y) - P^{(i)}(Y) = 0$, i.e.

$$\begin{aligned} P^{(i)}(X) T(Y \leftarrow X) A(Y \leftarrow X) &= P^{(i)}(Y) T(X \leftarrow Y) A(X \leftarrow Y) \\ \frac{P^{(i)}(X)}{P^{(i)}(Y)} &= \frac{T(X \leftarrow Y) A(X \leftarrow Y)}{T(Y \leftarrow X) A(Y \leftarrow X)}. \end{aligned} \tag{6.10}$$

This is known as **detailed balance** [57].

Systems generated by random walks such as this one can be characterized into a few catergories. If, after vising X, the probability of later re-visiting the neighbourhood around X vanishes we classify the system as *null*. If, on the other hand, after vising X, the walker re-visits X every subsequent T steps, we call the system *periodic*. Crucially, a system which is neither null nor periodic is called **ergodic**: revisiting X is allowed but is not done periodically [124]. The ergodicity condition can be stated as: For every pair of states X and Y there exists a non-zero stepping probability (possibly with intermediate steps)

$$W^{(n)}(Y|X) = W(Y|Z_n) W(Z_n|Z_{n-1}) \dots W(Z_1|X) \neq 0, \tag{6.11}$$

where the W-superscript denotes the number of steps taken. In short, ergodicity ensures we can reach any configuration in finite time, regardless of starting point.

It turns out that detailed balance and ergodicity are exactly sufficient[4] for the conditions stated at the end of section 6.1.1 to hold, i.e. the asymptotical distribution after a large number M of Markov steps converges to a unique distribution independently of starting configuration [125].

Connecting stepping probability with the asymptotic probability density

Having shown that under certain conditions, the asymptotic probability density resulting from the Markov chain is unique, it is now time to ask how we can construct the stepping probability in a way which produces the desired $P(X)$. It is exactly this the Metropolis algorithm achieves. The stepping probability $W(Y|X)$ associated with the known probability density $P(X)$ is in general unknown, and often too complicated to even easily write down [68]. However, according to Metropolis and co-workers, we can use a uniform suggestion probability T and accept any suggested

[4] Although not neccessary: detailed balance is more stringent demand than needed [69].

step with the probability

$$A(Y \leftarrow X) = \min\left\{1, \frac{P(Y)}{P(X)}\right\}. \tag{6.12}$$

This ensures the sampled Markov steps adhere to the correct relative probabilities within the distirbution [17]. For completeness, the transition matrix can under the Metropolis-scheme be written as [69]

$$W(Y|X) = \begin{cases} T(Y \leftarrow X)A(Y \leftarrow X) & Y \neq X \\ 1 - \int_{\mathcal{X}} \mathrm{d}Z\, T(Z \leftarrow X)A(Z \leftarrow X) & Y = X \end{cases}. \tag{6.13}$$

The steps of the basic Metropolis algorithm (sometimes referred to as brute-force Metropolis) can be written as

(1) Start in configuration X_0.

(2) Generate a suggested new configuration Y, according to the uniform suggestion probability $T(Y \leftarrow X)$.

(3) Accept the new value with probability $\min\{1, A(Y \leftarrow X)\}$. The acceptance probability is given by $A(Y \leftarrow X) = P(Y)/P(X)$.

(4) Assign $X_1 = Y$ if step (3) was accepted, else assign $X_1 = X_0$.

(5) Repeat steps (2)—(4).

6.1.2 The Metropolis-Hastings algorithm and importance sampling

The uniform suggestion probability proposed by Metropolis and co-workers is just *one way* of satisfying the detailed balance condition [118]. More generally, we need Eq. (6.10) to hold and thus

$$A(Y \leftarrow X) = \min\left\{1, \frac{T(X \leftarrow Y)P(Y)}{T(Y \leftarrow X)P(X)}\right\}. \tag{6.14}$$

It is immediately obvious that if the proposal probability $T(Y \leftarrow X)$ is uniform in the sense that the probability is constant within the hyper-cube in the configration space—any configuration with $|\mathbf{r}_i^{\text{new}} - \mathbf{r}_i^{\text{old}}| \leq \Delta x$ for all i is equiprobable—then the Ts drop out and we are left with the maximized $A(Y \leftarrow X)$ of Eq. (6.12) [69].

In general, we have a lot of freedom in the choice of $T(Y \leftarrow X)$. The only conditions being that it must be stochastic—in the sense of a stochastic matrix, i.e. non-zero and satisfying $\int_{\mathcal{X}} \mathrm{d}Y\, T(Y \leftarrow X) = 1$—and we must be able to easily directly sample it. Instead of using a T uniform in all degrees of freedom, we can instead take advantage of information about the target distribution. A clever choice of such a sampling

probability can take into account the local gradient of $P(X)$, and steer the Markov chain towards areas of higher probability. A good choice turns out to be the Greens function of the short-time approximation Fokker-Planck equation [69].

Fokker-Planck equation

In order to derive the Fokker-Planck equation, we first backtrack a short distance. Recall that the Chapman-Kolmogorov equation for a (in general time-dependent) transition probability takes the form

$$W(Y;t_2|X;t_0) = \int_{\mathcal{X}} \mathrm{d}Z\, W(Y;t_2|Z;t_1)W(Z;t_1|X;t_0). \tag{6.15}$$

Let now $P(X)$ be the probability distribution associated with the transition matrix $W(Y;t_1|X;t_0)$. We assume that the following holds:

$$\lim_{t\to\tau} \frac{1}{(t-\tau)} \int_{\mathcal{X}} \mathrm{d}Y\, (Y-X) \quad W(Y;t|X;\tau) = A(X,\tau), \tag{6.16}$$

$$\lim_{t\to\tau} \frac{1}{(t-\tau)} \int_{\mathcal{X}} \mathrm{d}Y\, (Y-X)^2 \quad W(Y;t|X;\tau) = 2B(X,\tau), \quad \text{and} \tag{6.17}$$

$$\lim_{t\to\tau} \frac{1}{(t-\tau)} \int_{\mathcal{X}} \mathrm{d}Y\, (Y-X)^{3+k}\, W(Y;t|X;\tau) = 0, \tag{6.18}$$

for any non-negative integer $k = 0, 1, 2, \ldots$. These integrals are sometimes called *Kramers-Moyal-expansions*. If we take the distribution to represent e.g. the position of a particle, the first condition constrains the mean velocity of the particle to be $A(X,\tau)$ [121]. Similarily, the second condition describes the influence on the particle's position by some random force. The third condition essentially ensures the motion must be continous, in that $W(Y;t_1|X;t_0)$ must vanish for $Y \neq X$ if $t_1 \neq t_0$ [122]. We will denote these integrals themselves (sans the denominator and limit) by $\langle (Y-X)^n \rangle$.

We will now consider some arbitrary function $g(x)$ which vanishes along with it's first derivative on the edges of our configuration space, $g(X \to \pm\infty) \to 0$ and $g'(X \to \pm\infty) \to 0$. Taking the integral of $g(x)$ weighed by $W(Y;t|X;0)$ over the all possible configurations, and employing the Chapman-Kolmogorov relation, yiels

$$\int_{\mathcal{X}} \mathrm{d}Y\, g(Y)W(Y;t|X;0) = \int_{\mathcal{X}}\int_{\mathcal{X}} \mathrm{d}Y\mathrm{d}Z\, g(Y)W(Y;t|Z;\tau)W(Z;\tau|X;0). \tag{6.19}$$

Exchanging g for it's Taylor series to second order around Z gives next

$$\begin{aligned}(6.19) \approx \int_{\mathcal{X}} \mathrm{d}Z\, g(Z)W(Z;\tau|X;0) + \\ \int_{\mathcal{X}} \mathrm{d}Z\, g'(Z)W(Z;\tau|X;0)\big\langle (Y-Z)\big\rangle + \\ \frac{1}{2}\int_{\mathcal{X}} \mathrm{d}Z\, g''(Z)W(Z;\tau|X;0)\big\langle (Y-Z)^2\big\rangle.\end{aligned} \tag{6.20}$$

Dividing this by $(t-\tau)$ and taking the limit $t \to \tau$ gives finally (after integration by parts and interchanging the limit and the integral)

$$\int_{\mathcal{X}} \mathrm{d}Y \, g(Y) \left[\frac{\partial W}{\partial t} + \frac{\partial AW}{\partial Y} - \frac{\partial^2 BW}{\partial Y^2} \right] = 0, \tag{6.21}$$

where we used the fact that g vanishes at infinity to handle the boundary term arising from the integration by parts [123]. We note that in the limit of vanishing $t - \tau$, the approximation of Eq. (6.20) is *exact* since all terms of order three or higher in the Taylor expansion vanish after the integration.

Since g is completely arbitrary, we see that

$$\frac{\partial W}{\partial t} + \frac{\partial (AW)}{\partial X} - \frac{\partial^2 (BW)}{\partial X^2} = 0 \tag{6.22}$$

must hold. This is known as the **Fokker-Planck equation** or sometimes *the second Kolmogorov equation*. It describes the time evolution of the prabability density of a particle (such as a Markovian walker) subject to a drift force. With $A(X,t) = 0$ and $B(X,t) = B(X)$, the Fokker-Planck equation reduces to a simple diffusion equation.

By taking the *Master equation* as a starting point, we can show that a corresponding equation also holds for the probability density itself, i.e. [126]

$$\frac{\partial P(X,t)}{\partial t} + \frac{\partial \big[A(X,t)P(X,t)\big]}{\partial X} - \frac{\partial^2 \big[B(X,t)P(X,t)\big]}{\partial X^2} = 0. \tag{6.23}$$

Alternatively, note that $P(X,t) = \int_{\mathcal{X}} \mathrm{d}X_0 \, W(X;t|X_0;0)P(X_0,t)$—thus if we assume $P(X_0,0) = \delta(X - X_0)$ then $W(X;t|X_0;0) = P(X,t)$—hence Eq. (6.22) immideately implies Eq. (6.23).

Solving the Fokker-Planck equation

Isotropic Brownian motion adheres to the usual diffusion equation in the same way that a Markovian random walker subject to a drift force satisfies the Fokker-Planck equation. A natural question thus arises: Can we solve the Fokker-Planck equation in the case of a random walker traversing the configuration space of our electronic positions? The answer is yes, and it will help us to improve the efficiency of the Metropolis algorithm.

Let now $P(\mathbf{R},t)$ be a probability distribution function subject to a drift force $\mathbf{F}(\mathbf{R}) = (F_1, F_2, \ldots, F_N)$, where $\mathbf{R} = \{\mathbf{r}_i\}_{i=1}^N$ as usual and $F_i = F_i(x_i)$ denotes the drift term on compontent i of $\mathbf{R}$.[5] This probability density satisfies the Fokker-Planck equation [17]

$$\frac{\partial P(\mathbf{R},t)}{\partial t} = \sum_{i=1}^{dN} D \frac{\partial}{\partial x_i} \left(\frac{\partial}{\partial x_i} - F_i(x_i) \right) P(\mathbf{R},t). \tag{6.24}$$

[5]The i-th component of $\mathbf{F}$ is the drift term corresponding to coordinate $i \bmod d$ of electron $\lfloor i/d \rfloor$ in d dimensions.

The diffusion constant is here denoted by D while lowercase d indicates the number of spatial dimensions. Since we are interested in the probability density which converges to $P(\mathbf{R}, t) = P(\mathbf{R}) = |\Psi(\mathbf{R})|^2$ (assuming for the moment that $\Psi(\mathbf{R})$ is normalized), we look for the solution in the case where the left hand side of Eq. (6.24) vanishes. An obviously sufficient (but possibly not neccessary) condition for this equation to hold is if it holds for each individual term of the sum, such that

$$\frac{\partial^2 P(\mathbf{R})}{\partial x_i^2} = P(\mathbf{R})\frac{\partial F_i(x_i)}{\partial x_i} + F_i(x_i)\frac{\partial P(\mathbf{R})}{\partial x_i}. \tag{6.25}$$

We note that in order for the right hand side to include a second derivative of $P(\mathbf{R})$, we must require

$$\mathbf{F}_i(\mathbf{r}) \sim g[P]\frac{\partial P(\mathbf{R})}{\partial x_i}. \tag{6.26}$$

In addition, in order for cancellation of the P in the first term on the right hand side, $g[P] = 1/P(\mathbf{R})$. It can be shown that the drift vector which results in a stationary probability density $P(\mathbf{R}) = |\Psi(\mathbf{R})|^2$ is

$$\mathbf{F} = 2\frac{1}{\Psi(\mathbf{R})}\nabla\Psi(\mathbf{R}), \tag{6.27}$$

with $\nabla = (\nabla_1, \nabla_2, \ldots, \nabla_N)$ being the vector of gradients w.r.t. each of the electrons [68]. This quantity is sometimes referred to as the *quantum force*, and acts to push the Markovian random walker in the direction of higher probability density. [6]

It turns out that a good choice for the transition probability $T(Y \leftarrow X)$ which incorporates the information in the quantum force $\mathbf{F}$ is the *Green's function* of the Fokker-Planck equation in the limit of small $\|X - Y\|$ [69]. We can rewrite the Fokker-Planck equation in terms of a differential operator $\mathcal{L}$,

$$\frac{\partial P}{\partial t} = \mathcal{L}P, \quad \text{with} \quad \mathcal{L} \equiv D\nabla \cdot (\nabla - \mathbf{F}). \tag{6.28}$$

The Green's function $G(Y, X; \delta t)$ is then the *operator inverse* of $\mathcal{L}$ in the sense that [43]

$$\mathcal{L}G(Y, X; \delta t) = \delta(\underbrace{t_Y - t_X}_{\delta t})\delta(Y - X). \tag{6.29}$$

The finite time step δt is the temporal distance between configurations X and Y.

[6] Please note that even though the quanitity $\mathbf{F}$ is called a quantum *force*, it is strictly speaking not a force. In fact it has dimensions of inverse length. We may instead think of the combination $D\mathbf{F}$ as a *drift velocity*. With D having dimensions of length squared per time, the combination has dimensions of lenght per time. This is in fact the combination which we will encounter (multiplied with a time step δt) shortly in the Green's function of the Fokker-Planck equation.

By inspection of we can straight away write down a representation of the solution, the Green's function is simply given by $G(Y, X; \delta t) = \exp(-\delta t \mathcal{L})$ [17]. Assuming a short time step δt, we can assume also correspondingly short spatial step $\|Y - X\|$ meaning the quantum force will remain essentially unchanged between the two configurations. Under this assumption, we can integrate the Green's function,

$$G(Y, X; \delta t) = \exp\left(D\delta t \left[\nabla^2 - \nabla \cdot \mathbf{F} - \mathbf{F} \cdot \nabla\right]\right), \tag{6.30}$$

over the time interval δt to obtain

$$G(Y, X; \delta t) = \left(\frac{1}{4\pi D\delta t}\right)^{-3N/2} \exp\left(\frac{-\left[Y - X - D\delta t \mathbf{F}(X)\right]^2}{4D\delta t}\right). \tag{6.31}$$

Recall that X here denotes a full configuration of the system, meaning all the electronic coordinates. This means that a more natural labelling of X and Y would be $Y = \mathbf{R}_{\text{new}}$ and $X = \mathbf{R}_{\text{old}}$, however we keep the Xs and Ys to remain in line with the notation of the previous sections.

The Langevin equation

Having found the solution, we are still left with the question of how to generate Fokker-Planck trajectories in practice. This is where we need to introduce the **Langevin equation**

$$\frac{\partial x(t)}{\partial t} = D\mathbf{F}(x(t)) + \eta. \tag{6.32}$$

The Langevin equation which corresponds to our Fokker-Planck equation describes e.g. the movement of a particle under the influence of a rapid and irregularly fluctuating random function of time [122]. The random force η is distributed according to a multidimensional Gaussian with a vanishing mean and variance $2D$.

The Langevin approach of adding random terms to the equations of motion, sometimes called *noise sources*, is fundamentally different but mathematically equivalent to the Fokker-Planck equation [127]. Whereas the former describes the evolution of the degrees of freedom of a system, the latter describes the evolution of the probability density of said degrees of freedom.

Integrating Eq. (6.32) over a short time interval δt gives rise to a time discrete form which we will find useful in generating Markov trajectories for use with the generalized Metropolis-Hastings algorithm: [17]

$$y = x + D\mathbf{F}(x)\delta t + \chi. \tag{6.33}$$

The random variable $\chi = \eta\delta t$ is a rescaled Gaussian, now with variance $2D\delta t$.

Importance sampling

Having introduced both the Fokker-Planck and the Langevin equation we are now ready to extend the original Metropolis algorithm with a non-uniform proposal distribution. The standard approach used to implement the Metropolis-Hastings algorithm with importance sampling in quantum mechanical problems is the following: Use the Langevin equation to propose moves and then accept or reject them according to the solution of the corresponding Fokker-Planck equation [68].

Whereas the Metropolis suggestion step described previously assigned equal probability of stepping to any point inside a $3N$ dimensional hyper-box of side lengths $\|y_i - x_i\| < \Delta x$, the importance sampled Metropolis-Hastings suggestions take the form

$$x_i^{\text{new}} = x_i^{\text{old}} + D\delta t\mathbf{F}(x_i) + \chi. \tag{6.34}$$

The random variable χ is distributed as a Gaussian around zero with variance $2D\Delta t$. Without the velocity drift term, this is the stepping scheme for an isotropic Brownian random walk in configuration space. However, with the added quantum force, the walker is pushed in the direction of higher probability, meaning more significant areas of the configuration space are visited proportionally more often.

Once a step has been suggested, the Metropolis test is performed. The full probability of accepting the step now has a non-vanishing contribution from $T(x_i^{\text{new}} \leftarrow x_i^{\text{old}})$: the ratio of the Green's functions of the Fokker-Planck equation. The acceptance probability is thus—according to Eq. (6.14)—the following [69]

$$A(x_i^{\text{new}} \leftarrow x_i^{\text{old}}) = \min\left\{1, \frac{G(x_i^{\text{new}}, x_i^{\text{old}}; \delta t)|\Psi(\mathbf{R}^{\text{new}})|^2}{G(x_i^{\text{old}}, x_i^{\text{new}}; \delta t)|\Psi(\mathbf{R}^{\text{old}})|^2}\right\}. \tag{6.35}$$

In summary, the Metropolis-Hastings algorithm with importance sampling consists of the following steps

(1) Start in configuration X_0.

(2) Generate a suggested new configuration Y according to the solution of the Langevin equation, Eq. (6.34).

(3) Accept the new value with probability $\min\{1, A(Y \leftarrow X)\}$. The acceptance probability is given by the Green's function ratio of Eq. (6.35).

(4) Assign $X_1 = Y$ if step (3) was accepted, else assign $X_1 = X_0$.

(5) Repeat steps (2)—(4).

6.2 Monte Carlo integration

Monte Carlo integration is a numerical scheme for estimating the value of definite integrals. The method differs from the usual grid-based methods in that the evaluation points are chosen at random. Whereas grid-based methods fail spectacularly as the number of dimensions increase past *a few*[7], the Monte Carlo scheme can somewhat overcome this "curse of dimensionality."

In the most basic form, Monte Carlo integration provides an estimate of the the integral value I by an average of N samplings of the integrand, uniformely within the integration range $[a, b]$, i.e.

$$I \equiv \int_a^b \mathrm{d}x\, f(x) \approx \frac{b-a}{N} \sum_{i=1}^{N} f(X_i). \tag{6.36}$$

The set of values $\{X_i\}_{i=1}^N$ are independent, identically distributed random samples. In general, the distribution need not be uniform. For *any* probability distribution function $P(x)$, we can obtain an estimate of I by

$$I = \int_a^b \mathrm{d}x\, f(x) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{P(X_i)}, \tag{6.37}$$

where the X_is are distributed according to P. Essentially, we are weighing the sum by how likely the value X_i is to be picked, according to the probability distribution $P(x)$. If $P(x_1) = 2P(x_2)$, then we weigh any sampling of x_1 only half as much as a corresponding sampling of the value x_2.

The advantage of the latter approach is immediately obvious: Imagine attempting to compute an approximation to the integral

$$I = \int_0^{100} \mathrm{d}x\, \mathrm{e}^{-x^2} x^2 \tag{6.38}$$

using the uniform sampling technique. Heuristically, at least 95% of sampled points will give an absolutely negligible contribution to the overall value of I. We may however, pick $P(x) = \mathrm{e}^{-x^2}$ and use Eq. (6.37). Using this scheme, every sampled point

[7] As of November 2017, the fastest super computer in the world is situated at the National Supercomputing Center in Wuxi, China: it has a performance of about 100 petaFLOPS $= 10^{17}$ FLOPS [128]. Let us say we wish to perform an integral over some electronic observable for a Neon atom. Under the Born-Oppenheimer approximation, this integral would be 30-dimensional. Using a conventional grid-based method, a conservative estimate for the number of grid points needed in each dimension to get a somewhat decent approximation may be e.g. 30. This means the grid would be 30 by 30 by ... by $30 = 30^{30} \sim 10^{44}$ large. Assuming every evaluation of the integrand consists of a single FLOP (again, a [very] conservative estimate), this means the total problem involves on the order of 10^{44} FLOPs. Running on the aforementioned Chinese super-computer, this would take on the order of 10^{27} s $\sim 10^{19}$ yr or about 10^{10} times the current age of the universe [68]. This is clearly not feasible.

would have a meaningful impact on the average since any x for which the integrand is negligible also now has a *negligible* chance to be sampled. This approach is called **importance sampling** and has two—already obvious—distinct advantages over the uniform or *brute force* method of Eq. (6.36). First, no time is wasted computing integrand values for which the integral is essentially independent because $f(x) \sim 0$. And secondly, for clever choices of $P(x)$, the computational load may be decreased because $g(x) \equiv f(x)/P(x)$ is a less computationally expensive quantity to sample. Furthermore, ideally fluctuations in $g(x)$ are considerably reduced. If we are e.g. able to sample our random points according to $f(x)$ itself, then $g(x) = 1$ and a single sample is sufficient.[8] In general, a distribution $P(x)$ *close* to $f(x)$ will give reduced statistical fluctuations in the integral estimate [17]. The importance sampling scheme also opens up the possibility of extending to an infinite range of integration.

6.2.1 Convergence properties of the Monte Carlo estimators

Taking a short step backwards, let now

$$\langle I^N \rangle = \frac{b-a}{N} \sum_{i=1}^{N} f(X_i) \approx \int_a^b \mathrm{d}x\, f(x) = I \tag{6.39}$$

denote the brute force Monte Carlo estimator of I using N samples. Computing the expectation value of $\langle I^N \rangle$ yields

$$\begin{aligned} E\big[\langle I^N \rangle\big] &= E\left[\frac{b-a}{N} \sum_{i=1}^{N} f(X_i)\right] = \frac{b-a}{N} \sum_{i=1}^{N} E\big[f(X_i)\big] \\ &= \frac{b-a}{N} \sum_{i=1}^{N} \int_a^b \mathrm{d}x\, f(x) P(x) \\ &= (b-a)\frac{1}{b-a} \int_a^b \mathrm{d}x\, f(x) = \int_a^b \mathrm{d}x\, f(x) = I, \end{aligned} \tag{6.40}$$

where we used that $P(x) = 1/(b-a)$ for the uniform probability density [124]. This shows that $\langle I^N \rangle$ is a so-called *unbiased estimator* of I [129]. The same is true of the generalized Monte Carlo estimator with non-uniform distribution $P(x)$, which we

[8] Assuming the integral is taken over the entire domain of P.

will denote $\langle J^N \rangle$:

$$\begin{aligned} E\big[\langle J^N \rangle\big] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{P(X_i)}\right] = \frac{1}{N}\sum_{i=1}^{N} E\left[\frac{f(X_i)}{P(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_a^b \mathrm{d}x\, \frac{f(x)}{P(x)}P(x) \\ &= \int_a^b \mathrm{d}x\, f(x) = I. \end{aligned} \tag{6.41}$$

For *uncorrelated* random variables, $\{Y_i\}_{i=1}^M$, the variance of the sum equals the sum of the variance, i.e. $\sigma^2[\sum_{i=1}^N Y_i] = \sum_{i=1}^N \sigma^2[Y_i]$. Note also the relation $\sigma^2[aY_i] = a^2\sigma^2[Y_i]$ [129] Using this, we may compute the variance of the Monte Carlo estimator as

$$\begin{aligned} \sigma^2\big[\langle J^N \rangle\big] &= \sigma^2\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{P(X_i)}\right] = \frac{1}{N^2}\sum_{i=1}^{N}\sigma^2\left[\frac{f(X_i)}{P(X_i)}\right] \\ &= \frac{1}{N}\sigma^2\left[\frac{f(X_i)}{P(X_i)}\right]. \end{aligned} \tag{6.42}$$

We note that the standard deviation in the mean $\sigma[\langle J^N \rangle] \to 0$ as $\mathcal{O}(\sqrt{N})$. The same is true of the basic Monte Carlo estimator $\langle I^N \rangle$, but $\sigma^2[f(X_i)/P(X_i)]$ is likely much smaller (for a good choice of P) than the corresponding $\sigma^2[f(X_i)]$.

6.2.2 The local energy, E_{L}

Like we did in section 3.1.1, we introduce a spatially dependent measure of the "instantaneous" energy, E_{L}. The *local energy* is defined as

$$E_{\mathrm{L}}(\mathbf{R}) = \frac{\hat{H}\Psi(\mathbf{R})}{\Psi(\mathbf{R})} = \frac{1}{\Psi(\mathbf{R})}\sum_{i=1}^{N}\left[-\frac{\nabla_i^2}{2} - \sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i - \mathbf{r}_A|} + \sum_{j=i+1}^{N}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}\right]\Psi(\mathbf{R}). \tag{6.43}$$

Note that if $\Psi(\mathbf{R})$ is the exact ground state wave function then $E_{\mathrm{L}}(\mathbf{R}) = E_{\mathrm{L}}$ is constant for all electronic configurations because

$$\frac{\hat{H}\Psi(\mathbf{R})}{\Psi(\mathbf{R})} = \frac{E_0\Psi(\mathbf{R})}{\Psi(\mathbf{R})} = E_0. \tag{6.44}$$

The variational energy can be written *in terms of* the local energy by [69]

$$E_{\mathrm{V}} = \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle} = \frac{\int \mathrm{d}\mathbf{R}\,\Psi^*(\mathbf{R})\hat{H}\Psi(\mathbf{R})}{\int \mathrm{d}\mathbf{R}\,\Psi^*(\mathbf{R})\Psi(\mathbf{R})} = \frac{\int \mathrm{d}\mathbf{R}\,\Psi^*(\mathbf{R})\left(\frac{\Psi(\mathbf{R})}{\Psi(\mathbf{R})}\right)\hat{H}\Psi(\mathbf{R})}{\int \mathrm{d}\mathbf{R}\,|\Psi(\mathbf{R})|^2}$$
$$= \frac{\int \mathrm{d}\mathbf{R}\,\Psi^*(\mathbf{R})\Psi(\mathbf{R})\frac{\hat{H}\Psi(\mathbf{R})}{\Psi(\mathbf{R})}}{\int \mathrm{d}\mathbf{R}\,|\Psi(\mathbf{R})|^2} = \int \mathrm{d}\mathbf{R}\,\rho(\mathbf{R})E_{\mathrm{L}}(\mathbf{R}), \tag{6.45}$$

with

$$\rho(\mathbf{R}) \equiv \frac{|\Psi(\mathbf{R})|^2}{\int \mathrm{d}\mathbf{R}\,|\Psi(\mathbf{R})|^2}. \tag{6.46}$$

Consider now this in light of Eq. (6.37), and note that for any trial wave function ansatz Ψ_{T} the following is an unbiased estimator for the true variational energy [17]

$$E\big[\Psi_{\mathrm{T}}\big] = \langle E_{\mathrm{L}}^M\rangle = \frac{1}{M}\sum_{i=1}^{M} E_{\mathrm{L}}(\mathbf{R}_i). \tag{6.47}$$

In the language of the previous section, $f = \rho(\mathbf{R})E_{\mathrm{L}}(\mathbf{R})$, $g = E_{\mathrm{L}}(\mathbf{R})$, and $P = \rho(\mathbf{R})$ which is the normalized probability density. This means that if we can pick samples randomly from the probability density function $\sim |\Psi_{\mathrm{T}}(\mathbf{R})|^2/\int \mathrm{d}\mathbf{R}\,|\Psi_{\mathrm{T}}(\mathbf{R})|^2$, then we can simply evaluate the local energy and take the average. This will give us an estimate for the true variational energy, as well as a measure of the stastical uncertainty from calculating the variance.

The Metropolis-Hastings (MH) algorithm of the previous section enables us to draw samples directly from $\rho(\mathbf{R})$. Note carefully that since MH always works in terms of ratios, $\rho(\mathbf{R})$ need strictly speaking not be normalized. We require simply a quantity proportional to $|\Psi_{\mathrm{T}}(\mathbf{R})|^2$.

A relatively short *burn-in* time is usually employed, wherein the first K samples are taken to be thermalization samples which are discarded. This is done in order to allow the MH Markov chain time to "forget" about it's starting point, and to allow the system as a whole to equilibrate and an energetically favourable region of the configuration space found [121].

6.2.3 Uncertainty estimates and correlated sampling

A stochastic method such as quantum Monte Carlo is useless without an estimate of the precision of the result. The quantity normally used to gauge this is the *estimated*

standard error of the mean, $\text{err}_{\langle E_\text{L}\rangle}$. The standard error is the standard deviation of the mean (for un-correlated samples),

$$\text{err}_{\langle E_\text{L}\rangle} = \sigma_{\langle E_\text{L}\rangle} = \frac{\sigma}{\sqrt{n}}, \tag{6.48}$$

where σ denotes the true standard deviation which we can estimate by the *sample standard deviation*,

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \langle x\rangle)^2} \approx \sigma. \tag{6.49}$$

Note the $N-1$ in the denominator[9]. In order to find a point estimate of σ, we could simply calculate

$$s^2 = \frac{\sum_{i=1}^{N}(x_i - \langle x\rangle)^2}{N-1} = \frac{1}{N-1}\sum_{i=1}^{N}x_i^2 - \frac{1}{N(N-1)}\left(\sum_{i=1}^{N}x_i\right) \tag{6.51}$$

where the sample variance s^2 is an unbiased estimator of σ^2, i.e. [129]

$$s^2 \approx \sigma^2 = \langle E_\text{L}^2\rangle - \langle E_\text{L}\rangle^2. \tag{6.52}$$

This, however, has the under-lying assumption that all the samples are *un-correlated*. If this is not the case—i.e. subsequent samples *are* correlated—then this equation will seriously underestimate the error [124].

In order to account for the auto-correlation of the Markov chain samples, we start instead from the definition of the variance in terms of the covariance, $\text{Var}(X) = \text{Cov}(X,X)$. Setting $\text{err}_{\langle E_\text{L}\rangle} = \sum_{ij}\text{Cov}(X_i,X_j)/N^2$ accounts for the correlation between subsequent samples [68]. The covariance $\text{Cov}(X_i,X_j)$ can we written as

$$\begin{aligned}\text{Cov}(X_i,X_j) &= \Big\langle \big(x_i - \langle x_i\rangle\big)\big(x_j - \langle x_j\rangle\big)\Big\rangle \\ &= \Big\langle x_i x_j - x_i\langle x_j\rangle - \langle x_i\rangle x_j + \langle x_i\rangle\langle x_j\rangle\Big\rangle \\ &= \langle x_i x_j\rangle - \langle x_i\rangle\langle x_j\rangle,\end{aligned} \tag{6.53}$$

[9]The reason for using a $N-1$ denominator—as opposed to the more intuitive N—is that

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \langle x\rangle)^2 \tag{6.50}$$

is an *unbiased* estimator of the variance, but the corresponding expression with the denominator replaced by $(N-1)\to N$ is not. This is sometimes referred to as *Bessel's correction* [129]. Although it is not true that s is an unbiased estimator of the standard deviation (because of the non-linearity of the square root), it is in a sense *less biased* than the N-denominator version.

with

$$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathrm{Cov}(X_i,X_j)=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i-\langle x\rangle)(x_j-\langle x\rangle)=\frac{1}{N}\mathrm{Cov}(x). \quad (6.54)$$

A proper estimate for the standard error of the mean can now be written in terms of the *sample covariance*, $\mathrm{Cov}(x)$, as [68]

$$\mathrm{err}^2_{\langle E_\mathrm{L}\rangle}=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathrm{Cov}(X_i,X_j)=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{1}{N}\mathrm{Cov}(x)=\frac{1}{N}\mathrm{Cov}(x). \quad (6.55)$$

Note carefully that Eq. (6.55) requires simultaneous knowledge of every single sample, and the evaluation is a double sum over the number of MC samples: a variable which routinely runs $N\gtrsim 10^6$. We would very much like to avoid having to perform this calculation. In order to circumvent this costly and inconvenient summation, lets us consider the the standard error of the mean split into two terms as

$$\mathrm{err}^2_{\langle E_\mathrm{L}\rangle}=\frac{1}{N}\mathrm{Var}(x)+\frac{1}{N}\big(\mathrm{Cov}(x)-\mathrm{Var}(x)\big). \quad (6.56)$$

The second term—the correlation term—can be written in terms of it's partial sums as

$$\begin{aligned}\frac{1}{N}\big(\mathrm{Cov}(x)-\mathrm{Var}(x)\big)&=\frac{2}{N}\sum_{k=1}^{N}\sum_{l=k+1}^{N}\big(x_k-\langle x\rangle\big)\big(x_l-\langle x\rangle\big)\\&=2\sum_{d=1}^{N-1}\Bigg[\underbrace{\frac{1}{N-d}\sum_{k=1}^{N-d}\big(x_k-\langle x\rangle\big)\big(x_{k+d}-\langle x\rangle\big)}_{\equiv f_d}\Bigg]. \quad (6.57)\end{aligned}$$

Dividing f_d by the sample variance yields the *autocorrelation function* $\kappa_d=f_d/\mathrm{Var}(x)$ [17]. Defining now the *autocorrelation time* τ,

$$\tau\equiv 1+2\sum_{d=1}^{N-1}\kappa_d, \quad (6.58)$$

we can rewrite $\mathrm{err}_{\langle E_\mathrm{L}\rangle}$ as

$$\begin{aligned}\mathrm{err}^2_{\langle E_\mathrm{L}\rangle}&=\frac{1}{N}\mathrm{Var}(x)+\frac{1}{N}\big(\mathrm{Cov}(x)-\mathrm{Var}(x)\big)\\&=\left(1+2\sum_{d=1}^{N-1}\right)\frac{1}{N}\mathrm{Var}(x)=\frac{\tau}{N}\mathrm{Var}(x). \quad (6.59)\end{aligned}$$

Note that if the samples are indeed uncorrelated, then $\tau=1$ and we recover the expression for the standard error of the mean as $\mathrm{err}_{\langle E_\mathrm{L}\rangle}=\sigma/\sqrt{N}$.

6.2.4 Blocking

In order to account for the correlation in the sampling, we may simply treat *blocks* of samples—with block size $b \geq \tau$ —as the samples. From N total samplings of E_{L}, this gives an *effective* number of samples

$$N_{\mathrm{eff}} = \frac{N}{b}. \tag{6.60}$$

Each effective sample is thus taken to be the average of b samples, and since $b \geq \tau$ we know subsequent *block* samples are uncorrelated. This means we can calculate the standard error of the mean *of the blocks* simply as

$$\begin{aligned}
\mathrm{err}^{\mathrm{blocks}}_{\langle E_{\mathrm{L}} \rangle} &= \frac{1}{N_{\mathrm{blocks}}} \sqrt{\langle E_{\mathrm{L}}^2 \rangle^{\mathrm{block}} - \left(\langle E_{\mathrm{L}} \rangle^{\mathrm{block}} \right)^2} \\
&= \frac{1}{N_{\mathrm{blocks}}} \sqrt{\sum_{i=1}^{N_{\mathrm{blocks}}} \left(\sum_{j=1}^{b} E_{\mathrm{L}}^{ib+j} \right)^2 - \left(\sum_{i=1}^{N_{\mathrm{blocks}}} \sum_{j=1}^{b} E_{\mathrm{L}}^{ib+j} \right)^2} \\
&= \frac{1}{N_{\mathrm{blocks}}} \sqrt{\sum_{i=1}^{N_{\mathrm{blocks}}} \left(\sum_{j=1}^{b} E_{\mathrm{L}}^{ib+j} \right)^2 - \left(\sum_{i=1}^{N} E_{\mathrm{L}}^{i} \right)^2},
\end{aligned} \tag{6.61}$$

where E_{L}^i denotes the i-th sample of the local energy. Calculating τ directly constitutes calculating the autocovariance, but we can estimate it using the calculated standard deviation for different block sizes, b. This procedure is known as **blocking** [130]:

(1) Calculate the standard deviation of the measurement set with block size $b = 1$, i.e. the usual standard deviation.

(2) Calculate the standard deviation using $b = 1, 2, \ldots$, and plot $\sigma(b)$ versus b.

(3) Take the value for which $\sigma(b)$ seems to plateu to be $b^* \approx \tau$.

(4) The blocking estimate of the true standard deviation is then taken to be $\sigma(b^*)$.

Since the standard deviation with block size $b = k$ contains $k/(k+1)$ as many samples as the corresponding deviation with block size $b = k + 1$, the expected change in the standard deviation is on the order of $\sim \sqrt{(k+1)/k}$. However, if the samples are correlated with $\tau < k$, then increasing the block size will sharply increase the calculated standard deviation. As $\tau \gtrsim k$, this effect vanishes and leaves only the $\sqrt{(k+1)/k}$ order change. The resulting plot has a sharp increase for low k and then a characteristic plateu around $k = \tau$.

Chapter 7

Artificial Neural Networks

The following is a *brief* introduction to the theory underlying the construction and training of artificial neural networks (ANNs). For a more comprehensive review of the subject, the reader is encouraged to survey relevant chapters from the recent master theses of Stende and Treider [1, 2]. This introduction follows closely the introduction of Raff and co-workers [131].

Artificial neural networks can be created in numerous, but we will focus exclusively on the most common architecture, namely *multilayer perceptrons* (MLP). The MLP neural networks are built from *layers* of connected *neurons*. In the artificial network, an input value (possibly a vector) is fed into the network model and then propagated through the layers, being processed through each neuron in turn. We will deal only with *feed forward* ANNs, meaning information always flows through the net in one direction only—essentially there are no loops. The entire ANN produces an output value (possibly a vector), which means we can think of it as a complicated function $\mathbb{R}^n \mapsto \mathbb{R}^m$. As we will see, it is possible to write down a closed form expression for this function and it is—crucially—possible to devise an efficient algorithm for calculating the gradient of the entire function w.r.t. any of the internal parameters.

7.1 Artifical neurons

A neuron is simply a model function for propagating information through the network. Inspired by biological neurons, the artificial neuron "fires" if it is stimulated by a sufficiently strong signal. The artificial neuron recieves a vector of input values $\mathbf{p}$. If the neuron is part of the very first hidden layer (this will be expanded upon shortly), the the input is simply the input value(s) to the NN. If one or more layers preceded the current one, $\mathbf{p}$ is a vector of outputs from the neurons in the previous layer.

The neuron is connected to the previous layers' neurons, and the strength of the connection is represented by a vector of weights, $\mathbf{w}$. Let us now consider a neuron which we will label by the index k. The output from neuron i (of the preceding layer),

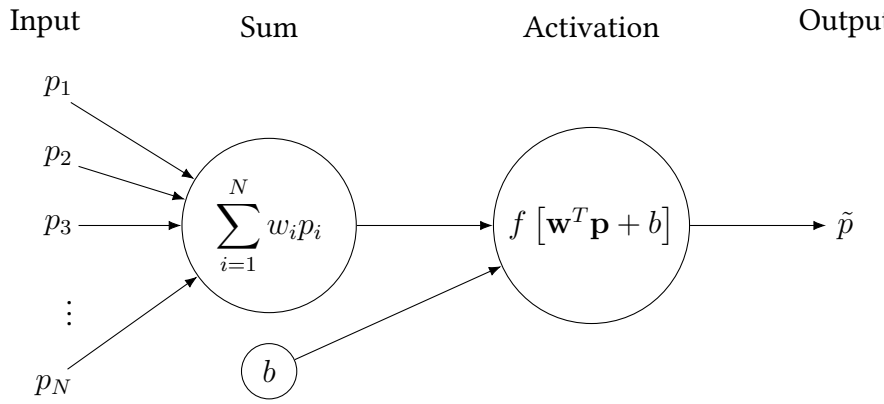

Figure 7.1: A model neuron, a constituent part of the artificial neural network model. The input from the previous layer $\mathbf{p}$ multiplied by corresponding weights $\mathbf{w}$ and summed. Then the bias b is added, and the activation function f is applied to the resulting $\mathbf{w}^T\mathbf{p}+b$. The output $\tilde{p}$ goes on to become input for neurons in the next layer.

p_i, is multiplied by the weight corresponding to the i—k connection, w_i. The combined weight vector multiplied by the input vector gives part of the total activation of the neuron,

$$\sum_{i=1}^{N} w_i p_i = \mathbf{w}^T\mathbf{p}. \tag{7.1}$$

The remaining part is known as the bias, b_k. This is a single real number. There is one for each neuron, and it acts as modifier making the neuron more or less likely to fire independently of the input.

The total input is passed to an activation (or transfer) function, which transforms it in some specified way, yielding the neuron *output* $\hat{p}_k$. This in turn becomes input for the neurons in subsequent layers.

Various different activation functions f are used for different purposes. The function may be linear or non-linear, but should vanish for small inputs and *saturate* for large inputs. For reasons that will become clear shortly, the conditions we enforce on f is continuity, boundedness, as well as non-constantness. We also demand it be monotonically increasing. A popular example, the sigmoid, takes the form

$$f(x) = \frac{1}{1+\mathrm{e}^{-x}}. \tag{7.2}$$

An example of the sigmoid is shown in Fig. 7.2. Numerous alternative transfer functions are in popular use, including the hyperbolic tangent $\tanh$, the inverse tangent $\tan^{-1}$, the rectified and exponential linear units (ReLU and ELU), Gaussians, and identity functions $f(x) = x$.

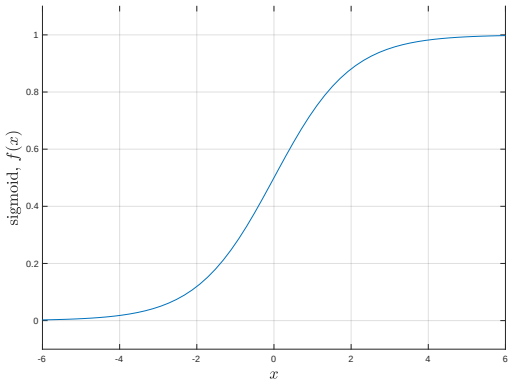

Figure 7.2: Example of a *sigmoid* function, used as a non-linear activation function for artificial neural networks.

In total, the action of a single neuron can be written

$$\text{input } \rightarrow f\left(\mathbf{w}^T\mathbf{p} + b\right) = \tilde{p} \rightarrow \text{ output.} \tag{7.3}$$

A schematic representation of the single neuron connected to the previous and acting as input for the next layers is shown in Fig. 7.1.

7.2 Network layers

The full artificial neural network is built up of layers of neurons. Data is fed sequentially through the network, starting in the input layer (the input values can be thought of as the first layer), through the *hidden* layers, and ending up in the output layer. The propagation needs to happen simultaneously across the network, as layer k needs the fully computed output of layer $k-1$ before the activations can be calculated.

A layer is—put simply—a collection of neurons, all of which are connected to the previous layer's neurons and the next layer's neurons. Let us label the individual neurons in layer k by index i, i.e. n_i^k. The bias of neuron i is then denoted b_i^k, and the weights connecting n_i^{k-1} to n_j^k is called w_{ji}. For each neuron there is a corresponding weight, so the weight vector is denoted $\boldsymbol{w}_i^k$. The combination of all weight vectors for layer k thus makes a matrix, which we will denote by a capital W^k,

$$W^k = \begin{pmatrix} w_{11}^k & w_{12}^k & w_{13}^k & \dots & w_{1N}^k \\ w_{21}^k & w_{22}^k & w_{23}^k & \dots & w_{2N}^k \\ w_{31}^k & w_{32}^k & w_{33}^k & \dots & w_{3N}^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{N1}^k & w_{N2}^k & w_{N3}^k & \dots & w_{NN}^k \end{pmatrix}, \tag{7.4}$$

or more compactly $(W^k)_{ij} = w_{ij}^k$. The collection of all biases for layer k is $\mathbf{b}^k$. In this

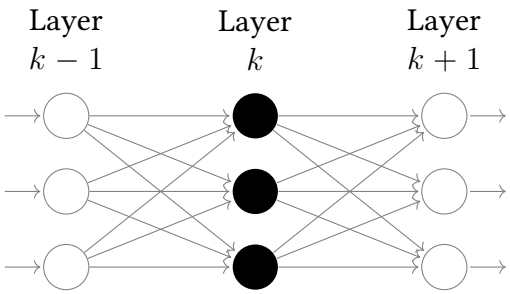

Figure 7.3: Schematic representation of a single ANN layer. Each neuron of the layer indexed k is connected from behind to all neurons in layer $k-1$. The connection weights can be organized into a matrix, W^{k-1}, and the action of layer k can be succinctly stated as $f(W^k\mathbf{p}^{k-1}+\mathbf{b}^k)$ where element-wise operation is assumed for the activation f.

notation, we may write the propagation of the signal from layer $k-1$ to layer k as

$$\begin{aligned}\mathbf{y}^k &= f(W^k\mathbf{y}^{k-1}+\mathbf{b}^k)\\ &= f\left(\begin{bmatrix} w_{11}^k & w_{12}^k & \dots & w_{1N}^k\\ w_{21}^k & w_{22}^k & \dots & w_{2N}^k\\ \vdots & \vdots & \ddots & \vdots\\ w_{N1}^k & w_{N2}^k & \dots & w_{NN}^k\end{bmatrix}\begin{bmatrix} y_1^{k-1}\\ y_2^{k-1}\\ \vdots\\ y_N^{k-1}\end{bmatrix}+\begin{bmatrix} b_1^k\\ b_2^k\\ \vdots\\ y_N^k\end{bmatrix}\right)\end{aligned} \tag{7.5}$$

or in Einstein notation

$$y_i^k = f\big(w_{ij}^k y_j^{k-1}+b_i^k\big). \qquad \text{(no sum over } k \text{ implied)} \tag{7.6}$$

In all of the preceeding three equations, application of f indicates *element wise* functional evaluation.

It is clear from Eq. (7.5) that propagation through an entire layer can be thought of as a matrix-vector product, a vector-vector summation, and a subsequent application of the transfer function f element-wise on the resulting vector.

A schematic representation of a layer consisting of three artificial neurons in a fully connected ANN is shown in Fig. 7.3.

7.3 The full network

A collection of L layers connected to eachother forms a full *network*. Note carefully that the network is nothing more than a (somewhat complex) function. If a single input and a single output value is specified, the action of the NN can be written out in closed form as [1]

$$\hat{y}=\sum_{j=1}^{M} w_{1j}^L f\left(\sum_{k=1}^{M} w_{jk}^{L-1} f\left(\sum_{i=1}^{M} w_{ki}^{L-2} f\left(\dots f\Big(w_{m1}^1 x_1+b_m^1\Big)\dots\right)+b_i^{L-2}\right)+b_k^{L-1}\right)+b_1^L \tag{7.7}$$

Here, we have taken the each layer to consist of M neurons. The scalar x_1 denotes the input value, while $\hat{y}$ is the NN output. From looking at Eq. (7.7), the usefulness of the

model is in no way obvious. But it turns out that for an ANN with at least one hidden layer populated with a finite amount number of neurons is a *universal approximator* [132]. This holds under the aforementioned assumptions on f, c.f. section 7.1. Being a universal approximator means (in this context) that the NN function can be made to be arbitrarily close to any continous Borel-measurable function (essentially *any* function we are likely to encounter) [133].

7.4 Training the ANN

Knowing that ANNs can be universal approximators is not helpful unless we can find a systematic way of obtaining suitable parameters to approximate any given function $g(x)$. This is where *training* comes in. In section 1.2 we defined machine learning as the science of creating computers capable of learning from experience. Teaching a NN to approximate a function is conceptually simple, and involves only three steps:

Assume input x and corresponding *correct* output y is known.

(1) Compute output $\text{NN}(x) = \hat{y}$ of the artificial neural network, and evaluate the *cost* function, typically $C(\hat{y}) \equiv \|y - \hat{y}\|_2$.

(2) Compute the gradient of $C(\hat{y})$ w.r.t. all the parameters of the network, w_{ij}^k and b_j^k.

(3) Adjust the parameters according to the gradients, yielding a better estimate $\hat{y}$ of the true output value y.

(4) Repeat (1)—(4).

The training scheme is known as *supervised learning*, because the NN is continually presented with x, y pairs, i.e. an input and an expected output. The cost (or objective or loss) function determines how happy the network is with it's own performance. A common choice for the cost function is the ℓ^2 norm, essentially the root squared difference,

$$C(\hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2 = \sqrt{\sum_{i=1}^{N_\text{O}} (y_i - \hat{y}_i)^2}. \tag{7.8}$$

In general, the output of the neural network is a vector of values, $\mathbf{y}$, and the cost function is taken across all outputs. In Eq. (7.8), the network produces N_O outputs for each input (which itself may be a vector).

Step (3) is easy to understand, but complex in practice. In order to update the network weights and biases, a measure of the expected change in the total output is

needed. Otherwise, any change would just be done at random[1]. This means we need to compute the set of derivatives

$$g_{ij}^k \equiv \frac{\partial C(\hat{\mathbf{y}})}{\partial w_{ij}^k}, \quad \text{and} \quad h_i^k \equiv \frac{\partial C(\hat{\mathbf{y}})}{\partial b_i^k}. \tag{7.9}$$

The most common algorithm for computing these derivatives is the **backpropagation** algorithm [134]. The method works by first pushing an input through the ANN, and computing the derivatives of the cost function w.r.t. the last layer weights and biases. The network is then traversed backwards, and the gradient w.r.t. all neuron parameters is found by repeated application of the chain rule. An explicit statement of the algorithm is found in any book on neural networks, see e.g. Raff and co-workers [131].

The final step in the training algorithm constitutes updating the weights according to the computed gradient. Possibly the simplest scheme for updating the weights is to just blindly follow the direction of the negative gradient, moving some set step length Δw and Δb. This is known as *gradient descent*, or *steepest descent*. Since the gradient represents the direction in the parameter-hyperspace giving the fastest decrease in $C(\hat{\mathbf{y}})$, this will in principle lead to a minimum. However, modern ANN methods use more sophisticated optimization schemes.

Whereas the gradient descent is a *first order* optimization algorithm—depending only on the first derivative (gradient)—it is possible to devise higher order methods taking advantage of the information contained in e.g. the second derivative (Hessian matrix). Often, the second order schemes require calculation and inversion of the Hessian. This is true of for example Newton's method, which minimizes $C(\hat{\mathbf{y}})$ by finding the roots of

$$\nabla C(\hat{\mathbf{y}}) = 0. \tag{7.10}$$

If the Hessian is too expensive to compute directly, it may be possible to estimate it in a computationally more feasible manner. This leads to a class of optimization algorithms known as Quasi-Newton methods, see e.g. the algorithm of Barzilai and Borwein [135].

A popular approach in modern ANN codes is using gradient descent based methods which automaticall adjust the training rate (the stepping size) for each parameter individually. The Adagrad and Adadelta methods are both examples of such methods, which also attempt to fine-tune the learning rate by considering a decaying backlog of stored squared gradient values [136]. The algorithm we will use in the present work is called Adam (derived from [but not an acronym for] *adaptive moment estimation*), which stores exponentially decaying averages of both gradients and squared gradients in order to find optimal step sizes [137]. For an accessible introduction to the specifics of the Adam optimizer, see e.g. the recent Master thesis of Stende [1].

[1] This is a possible approach, yielding a class of *genetic* optimization algorithms. We will not discuss such schemes in the present work.

Part III

Implementation and results

Chapter 8

Implementation: Hartree-Fock

The following is a description of the implementation of the Hartree-Fock framework described in chapter 4. The main body of the method consists of around $7\,000$ significant[1] lines of C++ code. It consists of about 15 significant classes, with associated sub-classes, of which a generic user is required to interact with only three for basic usage: The managing System class, an appropriate sub-class of the Atom super-class, and either one of RestrictedHartreeFock or UnrestrictedHartreeFock depending on which framework is desired.

The code is object oriented and completely general in that it can compute an approximation to the energy of any molecular configuration possible. Although it will of course be unfeasibly slow in doing so for large systems where the basis size far exceeds 10^2. As noted in section 3.3.4, the mathematically more tractable Cartesian Gaussian basis functions are used in place of the physically more realistic Slater type orbitals. It is in principle tuneable to any desired precision—bounded from below by the Hartree-Fock limit (see section 4.7)—by employing larger and larger basis sets. All basis sets used in the current work are taken from the *Basis Set Exchange* [138]. The specific basis sets used are described in section 8.1.

The code consists of two mostly disjoint parts. The bulk of the program consists of code neccessary to solve—in analytic fashion—one-, and two-electron integrals in terms of Gaussian orbitals. The second and more succinct part deals with setting up and solving the Roothan-Hall (Pople-Nesbet) equations.

The code base has been rigorously tested for first and second row atoms, with Gaussian orbitals up to and including f type, but expanding to heavier atoms and higher angular momentum basis functions is in principle straightforward. A number of basis sets are available, and adding new ones is made easy by the accompanying python script basisFileParser.py. It parses Gaussian basis set files in the Turbomole format, .tm, and outputs C++ code ready for use in the Hartree-Fock program [139].

After calculating the energy, the *Hartree-Fock basis* may be output to file for use

[1] As counted by the cloc program which counts *significant* lines of code, leaving out blank lines, comment lines, etc. [37]

later in e.g. the VMC code (see 9).

Before going on we present an overview of the basis sets employed. We then start off by describing simple usage of the code with some examples and then go on to expand on the implementation of some key classes and methods.

8.1 Basis sets used

In the present work we employ a range of different basis sets for the first and second row elements. The minimal 3-21G as well as the 3-21++G set which adds diffuse functions to hydrogen are taken from Binkley and co-workers [140]. The 6-31G sets—doubling the number of contracted functions for each core electron—are taken from Hehre and co-workers and Dill and co-workers (for Li and B) [91, 92]. Double zeta sets, adding another contracted function for each atomic orbital denoted 6-311G originate from the work of Krishnan and co-workers [141]. The corresponding sets with added diffuse and polarization functions G-311++G**, aswell as the G-311++G(2d,2p) basis set which adds p and d polarization functions to H is also taken from their work. Lastly, the correlation-consistent polarized triple zeta cc-pVTZ and the corresponding *augmented* aug-cc-pVQZ quintuple zeta with added diffuse functions are taken from Dunning and co-workers [93].

8.2 Introductory examples

The simplest usage of the HF code requires four lines of C++ code:

```
System He;
He.addAtom(new Helium("3-21G", vec{0,0,0}));
RestrictedHartreeFock solver(&He);
solver.solve();
```

First the System instance is created. Secondly, a He atom is added at the origin with a minimal 3-21G basis set. The vector vec parameter to the System::addAtom method determines the position of the atom. Subsequently, an RestrictedHartreeFock solver is setup and the last line solves the Roothan-Hall equations using default parameters of tollerance $\varepsilon = 10^{-8} E_h$, and a maximum of 50 iterations. Running the above code gives on the fly output shown in Fig. 8.1, and the final HF energy $E = -2.8356 E_h$.

More complicated molecular structures can easily be set up by simply adding more atoms. The following code sets up an un-restricted Hartree-Fock calculations of the ground state H_2O molecule

```
vec O  {0.000, 0.000, 0.000};
vec H1 {-1.430, 1.108, 0.000};
vec H2 {1.430, 1.108, 0.000};

System H2O;
H2O.addAtom(new Oxygen   ("6-311++G**", O));
```

```
H2O.addAtom(new Hydrogen("6-311++G**", H1));
H2O.addAtom(new Hydrogen("6-311++G**", H2));

UnrestrictedHartreeFock solver(&H2O);
solver.solve();
```

with the output $E = -76.0529 E_h$. Using the diffuse-polarized 6-311++G** basis set, the water molecule problem contains a grand total of 37 contracted basis functions.

8.3 Overview of select classes

8.3.1 Overlap and kinetic integral evaluation

The majority of the code base and the majority of the run-time of the Hartree-Fock program is taken up by integral evaluation code. The dominating factor w.r.t. computational complexity is the evaluation of the four-index J and K integrals. We will begin our discussion of integral evaluation with the OverlapIntegrator class.

Overlap integrals

In order to perform the overlap integrals we will employ the scheme of McMurchie and Davidson, hinted at in section 3.3.5 [85]. Exploiting the properties of the Hermite Gaussians, we can integrate Gaussian products with relative ease.

Recall the notation of the overlap distributions,

$$\Omega_{ij}(x) = g_i^\alpha(x; A_x) g_j^\beta(x; B_x) = K_{AB} x_A^i x_B^j \mathrm{e}^{-p x_p^2}, \tag{8.1}$$

with K_{AB} constant and $p = \alpha + \beta$. By the completeness of the Hermite polynomials, we may expand any polynomial of degree $i + j$ in terms of Hermite polynomials of degree $t \leq i + j$ [82]. This means we can write $\Omega_{ij}(x)$ in terms of $\Lambda_t(x)$ with expansion coefficients E_t^{ij}, i.e. [83]

$$\Omega_{ij} = \sum_{t=0}^{i+j} E_t^{ij} \Lambda_t. \tag{8.2}$$

Consider now the incremented $\Omega_{i+1,j}$, obtained by left-multiplying by an additional factor of x_A. We may use the Gaussian recurrence relation, Eq. (3.39), to relate this to Ω_{ij} by

$$\begin{aligned} \Omega_{i+1,j} &= A_x \Omega_{ij} = (x - A_x)\Omega_{ij} \\ &= (x - P_x)\Omega_{ij} + (P_x - A_x)\Omega_{ij} \\ &= x_p \Omega_{ij} - x_{PA} \Omega_{ij}. \end{aligned} \tag{8.3}$$

```
================== Starting SCF iterations =================
=> Maximum iterations:     50
=> Convergence criterion: 1e-08
=> Total basis size:       2
=> Number of atoms:        1
=> Number of electrons:    2
  ---------------------------------------------------------
  | Helium : 3-21G              ( 0.000,  0.000,  0.000)  |
  ---------------------------------------------------------

============================================================
      Iteration              Energy          Convergence
------------------------------------------------------------
              0         -1.67146855
              1         -2.30499718           0.726635902
              2         -2.58019861           0.340771411
              3         -2.70960718           0.159819875
              4         -2.77284144          0.0746829736
              5         -2.80424894          0.0346626565
              6         -2.81994424          0.0159319385

                            ...

             23         -2.83567975        2.09616335e-07
             24         -2.83567981        1.21250965e-07
             25         -2.83567984        6.93867538e-08
             26         -2.83567986         3.9361673e-08
             27         -2.83567987        2.21682601e-08
             28         -2.83567987         1.2409513e-08
             29         -2.83567987        6.91096458e-09
============================================================

Self consistency SUCCESFULLY reached.

=> Iterations used:                                    29
=> Final convergence test:          6.910964578388246e-09
=> Final electronic energy:            -2.835679869873355
=> Final energy (eV):                  -77.16279085427311
=> Final energy:                       -2.835679869873355
============================================================
```

Figure 8.1: Output of the first example program shown at the start of section 8.2. The right hand side column labeled convergence shows the average of absolute difference between eigenvalues of the Fock matrix between iterations. This is the test used to check for convergence, with ε being the convergence criterion given as input to RestrictedHartreeFock::solve. If no ε is provided, a default value of 10^{-8} is used.

We will now use the multiplication result for Hermite Gaussians shown in section 3.3.5—Eq. (3.46)—to expand the $x_p\Omega_{ij}$ term as

$$\begin{aligned} x_p\Omega_{ij} &= \sum_{t=0}^{i+j} E_t^{ij}\left[t\Lambda_{t-1} + \frac{1}{2p}\Lambda_{t+1}\right] \\ &= \sum_{t=1}^{i+j+1}\left[(t+1)E_{t+1}^{ij} + \frac{1}{2p}E_{t-1}^{ij}\right], \end{aligned} \tag{8.4}$$

yielding finally [84]

$$\Omega_{i+1,j} = \sum_{t=1}^{i+j+1}\left[\frac{1}{2p}E_{t-1}^{ij} + (t+1)E_{t+1}^{ij} + x_{PA}E_t^{ij}\right]\Lambda_t. \tag{8.5}$$

Equating this with the straight-forward expansion $\Omega_{i+1,j} = \sum_{t=1}^{i+j+1} E_t^{ij}\Lambda_t$, we find the recurrence relations for E_t^{ij} as [83]

$$E_0^{00} = K_{AB}, \tag{8.6}$$

$$E_t^{i+1,j} = \frac{1}{2p}E_{t-1}^{ij} + x_{PA}E_t^{ij} + (t+1)E_{t+1}^{ij}, \text{ and} \tag{8.7}$$

$$E_t^{i,j+1} = \frac{1}{2p}E_{t-1}^{ij} + x_{PB}E_t^{ij} + (t+1)E_{t+1}^{ij}. \tag{8.8}$$

Using the conditions $E_t^{ij} = 0$ if $t < 0$ or $t > i + j$, this gives us an algorithm for calculating Hermite expansion coefficients [3].

Having calculated the expansion, finding the overlap integral is trivial:

$$\begin{aligned} \left\langle g_{ijk}^{\alpha}(\mathbf{r}_\alpha;\mathbf{A})\middle|g_{lmn}^{\beta}(\mathbf{r}_\beta;\mathbf{B})\right\rangle &= \int_{-\infty}^{\infty}\mathrm{d}^3\mathbf{r}\, g_{ijk}^{\alpha}(\mathbf{r}_\alpha;\mathbf{A})g_{lmn}^{\beta}(\mathbf{r}_\beta;\mathbf{B}) \\ &= \sum_{t=0}^{i+l}\sum_{u=0}^{j+m}\sum_{v=0}^{k+n} E_t^{il}E_u^{jm}E_v^{kn}\int_{-\infty}^{\infty}\mathrm{d}^3\mathbf{r}\,\Lambda_{tuv}^{\alpha+\beta}(\mathbf{r}_P) \\ S_{il}S_{jm}S_{kn} &= E_0^{il}E_0^{jm}E_0^{kn}\sqrt{\left(\frac{\pi}{\alpha+\beta}\right)^3}. \end{aligned} \tag{8.9}$$

The OverlapIntegrator class computes the integral by Eq. (8.9) as

```
double OverlapIntegrator::computeIntegral(GaussianPrimitive* primitive1,
                                          GaussianPrimitive* primitive2) {

    m_hermiteGaussian.setupCoefficients(primitive1, primitive2);
    const double exponentSum    = m_hermiteGaussian.getExponentSum();
    m_sqrtPiOverP               = sqrt(M_PI / exponentSum);
    const int    xExponent1     = primitive1->xExponent();
    const int    yExponent1     = primitive1->yExponent();
    const int    zExponent1     = primitive1->zExponent();
```

```
    const int     xExponent2      = primitive2->xExponent();
    const int     yExponent2      = primitive2->yExponent();
    const int     zExponent2      = primitive2->zExponent();

    m_Ex = m_hermiteGaussian.getCoefficientX(xExponent1, xExponent2);
    m_Ey = m_hermiteGaussian.getCoefficientY(yExponent1, yExponent2);
    m_Ez = m_hermiteGaussian.getCoefficientZ(zExponent1, zExponent2);
    return m_Ex * m_Ey * m_Ez *
           m_sqrtPiOverP * m_sqrtPiOverP * m_sqrtPiOverP;
}
```

Evaluating the Hermite Gaussian expansion coefficients

The OverlapIntegrator class has a member instance of the HermiteGaussian class which sets up the Hermite factorization. This class essentially has one job: Computing the E_q^{ab} coefficients for any given $g_{ijk}^{\alpha}(\mathbf{r};\mathbf{A})g_{lmn}^{\beta}(\mathbf{r};\mathbf{B})$ product. The Hermite coefficients are stored in an array of arma::cube objects, which themselves are vectors of matrices or rank 3 tensors. In total, this makes the m_coefficients object a rank 4 tensor with indices ordered as (x, i, j, t) and size $(3, i+1, j+1, i+j+1)$.

An excerpt of the HermiteGaussian::computeCoefficients method is shown here:

```
void HermiteGaussian::computeCoefficients() {
    // ...
    double  alpha    = m_exponent1;
    double  beta     = m_exponent2;
    double  p        = alpha + beta;
    double  mu       = alpha * beta / p;
    vec     AB       = m_nucleusPosition1 - m_nucleusPosition2;
    vec     P        = (alpha * m_nucleusPosition1 +
                        beta  * m_nucleusPosition2) / p;
    vec     PA       = P - m_nucleusPosition1;
    vec     PB       = P - m_nucleusPosition2;

    for (int i = 0; i<3; i++) {
        cube& E = m_coefficients[i];
        double AB_ = AB(i);
        double PA_ = PA(i);
        double PB_ = PB(i);

        int iA = 0;
        E(0,0,0) = exp(- mu * AB_ * AB_);
        for (int iB = 0; iB < iB_loopLimits[i]; iB++) {
            for (int t = 0; t < t_loopLimits[i]; t++) {
                if (! (iA == 0 && iB == 0 && t == 0)) {
                    // E(i, j-1, t-1)
                    double previousIBpreviousT = 0;
                    if (isCoefficientNonZero(iA, iB-1, t-1)) {
                        previousIBpreviousT = E(iA, iB-1, t-1);
                    }
                    // E(i, j-1, t)
                    double previousIB = 0;
                    if (isCoefficientNonZero(iA, iB-1, t)) {
                        previousIB = E(iA, iB-1, t);
                    }
                    // E(i,j-1, t+1)
                    double previousIBnextT = 0;
```

```
                    if (isCoefficientNonZero(iA, iB-1, t+1)) {
                        previousIBnextT = E(iA, iB-1, t+1);
                    }
                    E(iA,iB,t) = (1./(2*p))   * previousIBpreviousT   +
                                 PB_          * previousIB            +
                                 (t+1)        * previousIBnextT;
                }
            }
        }

    // Repeat for i
    // ...
}
```

The HermiteGaussian::isCoefficientZero method checks if $t < 0$ or $t > i + j$, in which case $E_t^{ij} = 0$ is returned. Only the building of the $i = 0, j = 0, 1, 2, \ldots$ coefficients is shown; the loop over iA is omitted. The setup of the loopLimits arrays at the beginning of the function is also omitted, but the iA and iB upper limits are set to $i + 1$, $l + 1$, etc. The t, u, and v loops run from zero to $i + l + 1$, $j + m + 1$, and $k + n + 1$.

Computing kinetic integrals

The kinetic integrals,

$$T_{IJ} \equiv -\frac{1}{2}\left\langle g_{ijk}^{\alpha}(\mathbf{r}_\alpha;\mathbf{A})\left|\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right|g_{lmn}^{\beta}(\mathbf{r}_\beta;\mathbf{B})\right\rangle \tag{8.10}$$

are evaluated as linear combinations of overlap integrals, S_{ab}. We denote $g_{ijk} \equiv g_I$ and $g_{lmn} \equiv g_J$ for brevity when defining T_{IJ}. From Eq. (3.40) we know the effect on Cartesian Gaussians of differentiation w.r.t. x, and so we may write the kinetic integral components as

$$\begin{aligned} T_{ij} &= -\frac{1}{2}\left\langle g_i^{\alpha}(x_\alpha;A_x)\left|\frac{\partial^2}{\partial x^2}\right|g_j^{\beta}(x_\beta;B_x)\right\rangle \\ &= -\frac{1}{2}\left\langle g_i^{\alpha}(x_\alpha;A_x)\left|4\beta^2 g_{j+2}^{\beta}-2\beta(2j+1)g_j^{\beta}+j(j-1)g_{j-2}^{\beta}\right\rangle\right., \end{aligned} \tag{8.11}$$

where we have supressed the arguments on $g_j^{\beta}(x_\beta;B_x)$. Since the other two terms are independent of x, the *full* kinetic integral can be written in terms of the overlap integrals as

$$\begin{aligned} T_{IJ} &= -\frac{1}{2}\left\langle g_{ijk}^{\alpha}(\mathbf{r}_\alpha;\mathbf{A})\left|\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right|g_{lmn}^{\beta}(\mathbf{r}_\beta;\mathbf{B})\right\rangle \\ &= -\frac{1}{2}\left[T_{il}S_{jm}S_{kn}+S_{il}T_{jm}S_{kn}+S_{il}S_{jm}T_{kn}\right] \end{aligned} \tag{8.12}$$

with

$$T_{ij} = 4\beta^2 S_{i,j+2} - 2\beta(2j+1)S_{ij} + j(j-1)S_{i,j-2}. \tag{8.13}$$

The implementation of the KineticIntegrator class exploits the fact that if E_t^{ij} is known, then any overlap integral with $i' < i$ and/or $j' < j$ is already computed—simply extract the component of E with the correspondingly lower indices. Since building E_t^{ij} neccessitate the evaluation of all $E_t^{i'j'}$ with $i' < i$ and $j' < j$, these are guaranteed to already be known once the overlap $S_{i,j+2}$ is computed.

An excerpt of KineticIntegrator::computeIntegral is presented in the following. Note that the KineticIntegrator::computeAdjustedOverlapIntegral simply extracts the relevant E_t^{ij} indices and computes the corresponding S_{ij} accordingly, without re-computing the Hermite expansion.

```
double KineticIntegrator::computeIntegral(GaussianPrimitive* primitive1,
                                          GaussianPrimitive* primitive2) {
    // ...
    primitive2->adjustExponentX(2);
    primitive2->adjustExponentY(2);
    primitive2->adjustExponentZ(2);
    m_overlapIntegrator.computeIntegral(primitive1, primitive2);
    primitive2->adjustExponentX(-2);
    primitive2->adjustExponentY(-2);
    primitive2->adjustExponentZ(-2);

    vec& S = m_overlapIntegrals;
    S(0) = m_overlapIntegrator.getIntegralIndicesDimension(ix,jx,0);
    S(1) = m_overlapIntegrator.getIntegralIndicesDimension(iy,jy,1);
    S(2) = m_overlapIntegrator.getIntegralIndicesDimension(iz,jz,2);

    for (int dimension = 0; dimension < 3; dimension++) {
        for (int adjustment = -2; adjustment <= 4; adjustment+=4) {
            computeAdjustedOverlapIntegral(dimension, adjustment);
        }
    }
    for (int dimension = 0; dimension < 3; dimension++) {
        computeT(dimension);
    }

    return -0.5 * (m_T(0) * S(1) * S(2) +
                   S(0) * m_T(1) * S(2) +
                   S(0) * S(1) * m_T(2));
}
```

The KineticIntegrator::computeT simply computes T_{ij} by

```
void KineticIntegrator::computeT(int d) {
    double beta = m_primitive2->exponent();
    int    j    = m_primitive2->getExponentDimension(d);
    m_T(d) = 4*beta*beta * m_adjustedOverlapIntegrals(d,1) -
             2*beta*(2*j+1) * m_overlapIntegrals(dimension) +
             j*(j-1) * m_adjustedOverlapIntegrals(dimension,0);
}
```

8.3.2 Electron-nucleus Coulomb integrals

The electron-nucleus Coulomb integrals are on the form

$$V_{IJ} = \int \mathrm{d}^3\mathbf{r}\, \frac{g_I^\alpha(\mathbf{r};\mathbf{A})g_J^\beta(\mathbf{r};\mathbf{B})}{|\mathbf{r}-\mathbf{r}_C|}, \tag{8.14}$$

where g_I and g_J are primitives centered on nuclei A and B, respectively, and the integral is taken over coordinates relative to (a potentially different) nucleus C. We will denote $|\mathbf{r}-\mathbf{r}_C| \equiv r_C$, and we recognize the overlap distribution Ω_{IJ} in the numerator of the integrand,

$$V_{IJ} = \int \mathrm{d}^3\,\mathbf{r}\frac{\Omega_{IJ}}{r_C}. \tag{8.15}$$

Unfortunately, these integrals do not factor in Cartesian coordinates, but it turns out we may find a closed form solution in terms of the lower incomplete gamma function. We may rewrite the $1/r_C$ factor in terms of the integral over a Gaussian by employing the identiy [73]

$$\int_{-\infty}^{\infty} \mathrm{d}x\, \mathrm{e}^{-\lambda x^2} = \sqrt{\frac{\pi}{\lambda}}. \tag{8.16}$$

Using Eq. (8.16), $1/r_C$ becomes

$$\frac{1}{r_C} = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \mathrm{d}t\, \mathrm{e}^{-r_C^2 t^2}. \tag{8.17}$$

With the integral representation of $1/r_C$, V_{IJ} is a four dimensional integral,

$$V_{IJ} = \frac{K_{AB}}{\sqrt{\pi}} \int \mathrm{d}^3\mathbf{r}\, \left(x_A^i y_A^j z_A^k\right)\left(x_B^l y_B^m z_B^n\right)\mathrm{e}^{-pr_P^2} \int_{-\infty}^{\infty} \mathrm{d}t\, \mathrm{e}^{-r_C^2 t^2}, \tag{8.18}$$

where $p = \alpha + \beta$, $\mathbf{P}$ is the "center of mass" between $\mathbf{A}$ and $\mathbf{B}$, and $K_{AB} = \mathrm{e}^{-\alpha\beta r_{AB}^2/p}$ as usual (c.f. section 3.3.4). As we did with the overlap integrals, we may of course again expand the overlap distribution in terms of Hermite Gaussians

$$V_{IJ} = \frac{1}{\sqrt{\pi}} \int \mathrm{d}^3\mathbf{r} \sum_{tuv} E_t^{il} E_u^{jm} E_v^{kn} \Lambda_{tuv}^{\alpha+\beta}(\mathbf{r}_P) \int_{-\infty}^{\infty} \mathrm{d}t\, \mathrm{e}^{-r_C^2 t^2}. \tag{8.19}$$

One center Coulomb integrals

In order to make progress, we turn to the simplified one center integral of just

$$V_p \equiv \int \mathrm{d}^3\mathbf{r}\, \frac{\mathrm{e}^{-pr_P^2}}{r_C} = \frac{1}{\sqrt{\pi}} \int \mathrm{d}^3\mathbf{r} \int_{-\infty}^{\infty} \mathrm{d}t\, \mathrm{e}^{-pr_P^2}\mathrm{e}^{-t^2 r_C^2}. \tag{8.20}$$

Using the Gaussian product rule and subsequently applying Eq. (8.16) thrice gives [84]

$$V_p = \frac{2}{\sqrt{\pi}} \int_0^\infty \mathrm{d}t \, \exp\left(pR_{CP}^2 \frac{t^2}{p+t^2}\right) \left(\frac{\pi}{p+t^2}\right)^{3/2}. \tag{8.21}$$

The even integral over $(-\infty, \infty)$ was transformed to twice the integral over $[0, \infty)$, and R_{CP} denotes (predictably) the distance $R_{CP} = |\mathbf{C} - \mathbf{P}|$. Performing next the substitution $u^2 \equiv t^2/(p+t^2)$ with transformed integration measure

$$\mathrm{d}t = \frac{1}{p} \left(\frac{t^2}{u^2}\right)^{3/2} \mathrm{d}u, \tag{8.22}$$

and integration limits $[0, 1]$ we obtain

$$V_p = \frac{2\pi}{p} \int_0^1 \mathrm{d}u \, \mathrm{e}^{-pR_{CP}^2 u^2}. \tag{8.23}$$

Introducing the **Boys function**, $F_n(x) = \int_0^1 \mathrm{d}t \, \mathrm{e}^{-xt^2} t^{2n}$, we can rewrite this finally as [83]

$$V_p = \frac{2\pi}{p} F_0(pR_{CP}^2). \tag{8.24}$$

Evaluing the Boys function

The Boys function is related to the lower incomplete gamma function as

$$F_n(x) = \frac{\gamma(n + 1/2, x)}{2x^{n+1/2}} \tag{8.25}$$

where the integral representation of the incomplete gamma function can be written as

$$\Gamma(s, x) = \int_x^\infty \mathrm{d}t \, t^{s-1} \mathrm{e}^{-t} \quad \text{and} \quad \gamma(s, x) = \int_0^x \mathrm{d}t \, t^{s-1} \mathrm{e}^{-t}, \tag{8.26}$$

for the upper (Γ) and lower (γ) regions, respectively [142, 143]. The gamma function proper is obviously just the sum of the lower and upper regions, $\Gamma(\alpha) = \gamma(\alpha, x) + \Gamma(\alpha, x)$. Integrating $\gamma(s, x)$ by parts yields the realation

$$\gamma(s, x) = \int_0^x \mathrm{d}t \, t^{s-1} \mathrm{e}^{-t} = -\mathrm{e}^{-t} t^{s-1} + \int_0^x \mathrm{d}t \, t^{s-2} \mathrm{e}^{-t}, \tag{8.27}$$

from which we can derive the following recurrence relation for $F_n(x)$ [83]

$$F_n(x) = \frac{2x \, F_{n+1}(x) + \mathrm{e}^{-x}}{2n+1}. \tag{8.28}$$

Using Eq. (8.28) we may efficiently find $F_m(x)$ for any $m \leq n$ if we compute once the value $F_n(x)$. This involves only $5(n-m)$ FLOPs, as opposed to the re-evaluation in terms of the gamma function which is significantly more computationally expensive (see e.g. [144]). In the Hartree-Fock framework, this is implemented in the BoysFunction class, and the method

```
double BoysFunction::computeAndApplyDownwardRecurrence(double x, double n) {
    m_recurrenceValues = zeros<vec>(n+1);
    m_recurrenceValues(n) = compute(x, n);

    const double expMinusX = std::exp(-x);
    for (int m=n; m>0; m--) {
        m_recurrenceValues(m-1) = (2*x*m_recurrenceValues(m) + expMinusX) /
                                  (2.0*m-1.0);
    }
    return m_recurrenceValues(0);
}
```

The BoysFunction::compute method simply evalues the Boys function $F_n(x)$ by application of Eq. (8.25) and the boost::math library[2],

```
double BoysFunction::analyticalIncompleteGammaFunction(double x, double n) {
    const double nPlusOneHalf = n+0.5;
    return (x==0) ? 1.0/(n+1) : 1.0/(2*pow(x,nPlusOneHalf)) *
                                boost::math::tgamma_lower(nPlusOneHalf,x);
}
```

Computing Hermite integral expansions

Recall the Hermite Gaussian expansion of the V_{IJ} integrand of Eq. (8.19),

$$V_{IJ} = \frac{1}{\sqrt{\pi}} \int \mathrm{d}^3\mathbf{r} \sum_{tuv} E_t^{il} E_u^{jm} E_v^{kn} \Lambda_{tuv}^{\alpha+\beta}(\mathbf{r}_P) \int_{-\infty}^{\infty} \mathrm{d}t\, \mathrm{e}^{-r_C^2 t^2}. \tag{8.29}$$

Moving the expansion coefficients out of the integral, and inserting the definition of the Hermite Gaussians gives

$$\begin{aligned} V_{IJ} &= \frac{\sum_{tuv} E_t^{il} E_u^{jm} E_v^{kn}}{\sqrt{\pi}} \int \mathrm{d}^3\mathbf{r}\, \frac{\partial^t \partial^u \partial^v}{\partial P_x^t \partial P_y^u \partial P_z^v} \frac{\mathrm{e}^{-p r_P^2}}{r_C} \\ &= \frac{\sum_{tuv} E_t^{il} E_u^{jm} E_v^{kn}}{\sqrt{\pi}} \frac{\partial^t \partial^u \partial^v}{\partial P_x^t \partial P_y^u \partial P_z^v} \int \mathrm{d}^3\mathbf{r}\, \frac{\mathrm{e}^{-p r_P^2}}{r_C} \\ &= \frac{\sum_{tuv} E_t^{il} E_u^{jm} E_v^{kn}}{\sqrt{\pi}} \frac{\partial^t \partial^u \partial^v}{\partial P_x^t \partial P_y^u \partial P_z^v} \left[\frac{2\pi}{p} F_0(p R_{PC}^2) \right], \end{aligned} \tag{8.30}$$

[2] http://www.boost.org/doc/libs/1_65_1/libs/math/doc/html/math_toolkit/sf_gamma/igamma.html

where the result of the one center Coulomb integral—Eq. (8.24)—were inserted and the order of integration and differentiation were swapped in accordance with Leibniz's rule [84, 145].

Defining the **Hermite integrals**

$$R^n_{tuv}(p, R_{PC}) \equiv (-2p)^n \frac{\partial^{t+u+v} F_n(pR^2_{PC})}{\partial P^t_x \partial P^u_y \partial P^v_z}, \tag{8.31}$$

we can rewrite the complete electron-nucleus Coulomb integral in it's final form as [83]

$$V_{IJ} = \frac{2\pi}{p} \sum_{t=0}^{i+l} \sum_{u=0}^{j+m} \sum_{v=0}^{k+n} E^{il}_t E^{jm}_u E^{kn}_v R^0_{tuv}(p, R_{PC}). \tag{8.32}$$

Differentiation of the Boys function leads to the three recurrence relations which we—together with $R^n_{000} = (-2p)F_n$—will use to compute the R^n_{tuv} values:

$$R^n_{t+1,uv} = tR^{n+1}_{t-1,uv} + x_{PC}R^{n+1}_{tuv}, \tag{8.33}$$

$$R^n_{t,u+1,v} = uR^{n+1}_{t,u-1,v} + y_{PC}R^{n+1}_{tuv}, \text{ and} \tag{8.34}$$

$$R^n_{tu,v+1} = vR^{n+1}_{tu,v-1} + z_{PC}R^{n+1}_{tuv}. \tag{8.35}$$

In the source code of the Hartree-Fock program, this is implemented in the class HermiteGaussianIntegral. Once again, the coefficients are stored in a rank 4 tensor called m_coefficients. An excerpt of the method computing R^n_{tuv}s is shown here:

```
void HermiteGaussianIntegral::setupCoefficients(int          t,
                                                int          u,
                                                int          v,
                                                double       p,
                                                arma::vec    PC) {
    // ...
    arma::field<arma::cube>& R = m_coefficients;
    BoysFunction&            F = m_boysFunction;

    double x = p * arma::dot(m_PC, m_PC);
    R(0)(0,0,0) = F.computeAndApplyDownwardRecurrence(x, m_tuv+1);

    double minusTwoPPowerM = 1;
    for (int m = 1; m < m_tuv+1; m++) {
        minusTwoPPowerM *= (-2*p);
        R(m)(0,0,0) = minusTwoPPowerM * F[m];
    }

    for (int tuv = 1; tuv < m_tuv+1;      tuv++)
    for (int n   = 0; n   < m_tuv+1-tuv; n  ++)
    for (int t   = 0; t   < m_t+1;       t  ++)
    for (int u   = 0; u   < m_u+1;       u  ++)
    for (int v   = 0; v   < m_v+1;       v  ++) {
        if (t + u + v != tuv || t + u + v == 0) {
            continue;
        }
        int tuvMax = max(t, max(u, v));
```

```
            double newCoefficient = 0;
            if (tuvMax == t) {
                newCoefficient = (t-1)     * getCoefficient(n+1,t-2,u,v) +
                                 m_PC(0)   * getCoefficient(n+1,t-1,u,v);
            } else if (tuvMax == u) {
                newCoefficient = (u-1)     * getCoefficient(n+1,t,u-2,v) +
                                 m_PC(1)   * getCoefficient(n+1,t,u-1,v);
            } else if (tuvMax == v) {
                newCoefficient = (v-1)     * getCoefficient(n+1,t,u,v-2) +
                                 m_PC(2)   * getCoefficient(n+1,t,u,v-1);
            }
            R(n)(t,u,v) = newCoefficient;
        }
    }
```

The m_tuv variable holds the sum $t+u+v$, and maxIndex represents $m_{\max} \equiv t+u+v+1$. This is the highest order $R^{m_{\max}}_{000}$ we need to compute directly by $R^{n}_{000} = (-2p)F_n$ (by downward recurrence on $F_{m_{\max}}$) before applying the recurrence relations of Eq. (8.33)-(8.35).

Finally, the full electron-nucleus interaction integrals are computed in the Hartree-Fock code as

```
double ElectronNucleusIntegrator::computeIntegral(
                                        GaussianPrimitive* primitive1,
                                        GaussianPrimitive* primitive2) {

    HermiteGaussianIntegral& R = m_hermiteGaussianIntegral;
    HermiteGaussian&         E = m_hermiteGaussian;
    R.setupCoefficients(primitive1, primitive2, m_nucleusPosition);
    E.setupCoefficients(primitive1, primitive2);
    //...
    double integral = 0;
    for (int t = 0; t < tLimit; t++)
    for (int u = 0; u < uLimit; u++)
    for (int v = 0; v < vLimit; v++) {
        double Eproduct = 1;
        Eproduct *= E.getCoefficientDimension(x1,x2,t,0);
        Eproduct *= E.getCoefficientDimension(y1,y2,u,1);
        Eproduct *= E.getCoefficientDimension(z1,z2,v,2);
        integral += Eproduct * R.getCoefficient(0,t,u,v);
    }
    return integral * 2*M_PI / p;
}
```

8.3.3 Electron-electron exchange integrals

The electron-electron exchange integrals are on the form

$$V_{ABCD} = \int \mathrm{d}^3\mathbf{r}_1 \int \mathrm{d}^3\mathbf{r}_2 \, \frac{g^{\alpha}_A(\mathbf{r}_1;\mathbf{A}) g^{\beta}_B(\mathbf{r}_1;\mathbf{B}) g^{\gamma}_C(\mathbf{r}_2;\mathbf{C}) g^{\delta}_D(\mathbf{r}_2;\mathbf{D})}{|\mathbf{r}_1 - \mathbf{r}_2|}. \tag{8.36}$$

As with the Coulomb integrals, we recognize a set of overlap distributions in the numerator which we can expand in terms of Hermite Gaussians as

$$\begin{aligned} V_{ABCD} &= \int \mathrm{d}^3\mathbf{r}_1 \int \mathrm{d}^3\mathbf{r}_2 \frac{\Omega_{AB}^{\alpha+\beta}(\mathbf{r}_1;\mathbf{P})\Omega_{CD}^{\gamma+\delta}(\mathbf{r}_2;\mathbf{Q})}{r_{12}} \\ &= \sum_{tuv} E_{tuv}^{AB} \sum_{\tau\mu\nu} E_{\tau\mu\nu}^{CD} \int \mathrm{d}^3\mathbf{r}_1 \int \mathrm{d}^3\mathbf{r}_2 \frac{\Lambda_{AB}^{\alpha+\beta}(\mathbf{r}_1;\mathbf{P})\Lambda_{CD}^{\gamma+\delta}(\mathbf{r}_2;\mathbf{Q})}{r_{12}}. \end{aligned} \tag{8.37}$$

It turns out we can write the integration over $\mathbf{r}_1$ and $\mathbf{r}_2$ in terms of the Hermite integrals, as

$$V_{ABCD} = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} \sum_{tuv} E_{tuv}^{AB} \sum_{\tau\mu\nu} E_{\tau\mu\nu}^{CD} (-1)^{\tau+\mu+\nu} R_{t+\tau,u+\mu,v+\nu}(\zeta, \mathbf{R}_{PQ}), \tag{8.38}$$

where with $\zeta = pq/(p+q)$, $p = \alpha + \beta$, $q = \gamma + \delta$, $\mathbf{R}_{PQ} = \mathbf{P} - \mathbf{Q}$, and $\mathbf{P}$ ($\mathbf{Q}$) is the "center of mass" between $\mathbf{A}$ and $\mathbf{B}$ ($\mathbf{C}$ and $\mathbf{D}$) (for details, see e.g. Helgaker and Taylor, McMurchie and Davidson, or Boys [80, 83, 85]).

In the source code, the ElectronElectronIntegrator class sets up two Hermite Gaussian expansions—one for the $\mathbf{A}, \mathbf{B}$ overlap and one for the $\mathbf{C}, \mathbf{D}$ overlap—and a single Hermite integral expansion. The implementation is straightforward, the heavy lifting is done in the HermiteGaussian and HermiteGaussianIntegral classes:

```
double ElectronElectronIntegrator::computeIntegral(
                                            GaussianPrimitive* primitive1,
                                            GaussianPrimitive* primitive2,
                                            GaussianPrimitive* primitive3,
                                            GaussianPrimitive* primitive4) {
    HermiteGaussian&          E12 = m_hermiteGaussian12;
    HermiteGaussian&          E34 = m_hermiteGaussian34;
    HermiteGaussianIntegral& R   = m_hermiteGaussianIntegral

    E12.setupCoefficients(primitive1, primitive2);
    E34.setupCoefficients(primitive3, primitive4);
    setupHermiteGaussianIntegral(primitive1, primitive2,
                                 primitive3, primitive4);
    // ...
    double integral = 0;
    for (int t  = 0; t  < tuvLimits[0]; t ++)
    for (int u  = 0; u  < tuvLimits[1]; u ++)
    for (int v  = 0; v  < tuvLimits[2]; v ++)
    for (int t_ = 0; t_ < tuvLimits[3]; t_++)
    for (int u_ = 0; u_ < tuvLimits[4]; u_++)
    for (int v_ = 0; v_ < tuvLimits[5]; v_++) {
        double Eproduct = 1;
        Eproduct *= E12.getCoefficientDimension(x1,x2,t,0);
        Eproduct *= E12.getCoefficientDimension(y1,y2,u,1);
        Eproduct *= E12.getCoefficientDimension(z1,z2,v,2);

        Eproduct *= E34.getCoefficientDimension(x3,x4,t_,0);
        Eproduct *= E34.getCoefficientDimension(y3,y4,u_,1);
        Eproduct *= E34.getCoefficientDimension(z3,z4,v_,2);

        double R = R.getCoefficient(0, t+t_, u+u_, v+v_);
```

```
            double sign = ((t_ + u_ + v_) % 2) == 0 ? 1 : -1;
            integral += Eproduct * R * sign;
        }
        double p1       = primitive1->exponent() + primitive2->exponent();
        double p2       = primitive3->exponent() + primitive4->exponent();
        return m_2sqrtPiToThe5 * integral / (p1*p2*sqrt(p1+p2));
    }
```

The variables xn,yn, and zn represent the exponent of the x, y, and z term in primitive n. The limits of the sum x1+x2+1, y1+y2+1, and so on.

The ContractedIntegrator class

Having presented the integrators for the Gaussian primitives, the integrals over the contracted Gaussians is trivial: we simply take a linear combination of the primitive integrals.

8.3.4 The RestrictedHartreeFockSolver class

For brevity, only the restricted case is presented in the following. As we derive in section 4, the restricted Hartree-Fock formalism expanded in a basis $\{\phi_i\}_i$ orbitals lead to the Roothan-Hall (RH) equations. The RH equations take the form

$$FC = \varepsilon SC, \tag{8.39}$$

where F is the Fock matrix with eigenvalues ε, S is the overlap matrix relative to the orbitals, and C the coefficient matrix representing the expansion of the new Hartree-Fock orbitals in the $\{\phi_i\}_i$ basis. The Fock matrix depends crucially on the coefficient matrix, $F = F(C)$, meaning we must solve the non-linear RH equations by employing some linearization scheme. The universally used scheme is the fixed-point iterative scheme referred to as self-consistent field iterations (SCF), wherein an initial guess is chosen for C and the Fock matrix computed. Secondly, the RH equations are solved and a new (hopefully improved) estimate for C is produced. This is then re-inserted into $F(C)$, giving an updated Fock matrix for which the RH equations are solved once again. This is reapeated until convergence is achieved.

Diagonalization of the Fock matrix

The overlap matrix, S, contains the overlap integrals of all combinations of basis functions. If the basis set is orthonormal $S = \mathbb{1}$ and we may simply ignore it and solve the Roothan-Hall equations at every iteration like an ordinary eigenvalue equation. However, the contracted Gaussian basis is in general not orthonormal and so we must transform Eq. (8.39) into an equation we are able to solve using normal linear algebra tools. In theory we could simply compute S^{-1} and apply it to Eq. (8.39) from the left, leaving us with

$$S^{-1}F\mathbf{C}_k = \varepsilon_k S^{-1}S\mathbf{C}_k = \varepsilon_k C_k, \tag{8.40}$$

for each eigenvector indexed in k. However, $S^{-1}F$ is in general not Hermitian leading to potential difficulties. Instead, we will take a different route.

Instead we will apply a coordinate transformation which renders S an identity matrix, solve the equation, and then transform back to the original basis. Applying a basis change to a matrix constitutes a similarity transform, thus we require a matrix V such that [58]

$$V^\dagger S V = \mathbb{1}. \tag{8.41}$$

The dagger superscript denotes here the Hermitian conjugate, $V^\dagger = (V^*)^T$. The same transformation needs to be applied to the Fock matrix, but F does of course not become diagonal as a result. Let us now left multiply the RH equation by $V^\dagger$, considering for the moment the equation for only a single eigenvector-eigenvalue pair:

$$\begin{aligned} V^\dagger F \mathbf{C}_k &= \varepsilon_k V^\dagger S \mathbf{C}_k \\ V^\dagger F \underbrace{V V^{-1}}_{\mathbb{1}} \mathbf{C}_k &= \varepsilon_k V^\dagger S \underbrace{V V^{-1}}_{\mathbb{1}} \mathbf{C}_k \\ \underbrace{V^\dagger F V}_{F'} \underbrace{V^{-1}\mathbf{C}_k}_{\mathbf{C}'_k} &= \varepsilon_k \underbrace{V^\dagger S V}_{\mathbb{1}} \underbrace{V^{-1}\mathbf{C}_k}_{\mathbf{C}'_k}. \end{aligned} \tag{8.42}$$

This is called a whitening transform. Note carefully that ε is just a number, thus is guaranteed to commute with $V^\dagger$. Defining the transformed eigenvector $\mathbf{C}'_k \equiv V^{-1}\mathbf{C}_k$, and the coordinate transformed Fock matrix $F' \equiv V^\dagger F V$, we find the transformed RH equation (which is now a regular eigenvalue problem)

$$F' \mathbf{C}'_k = \varepsilon_k \mathbf{C}'_k. \tag{8.43}$$

This leaves us still the problem of finding a suitable matrix V. It turns out that S is guaranteed to be positive definite, meaning the set of matrices for which $V^\dagger S V = \mathbb{1}$ holds is infinite. We will choose $V = U s^{-1/2}$, with U being the matrix of eigenvectors of S (the columns) and s holds the inverse square root of the eigenvalues on the diagonal [57]. We note that

$$\begin{aligned} V^\dagger S V &= (U s^{-1/2})^\dagger S U s^{-1/2} \\ &= (s^{-1/2})^\dagger \underbrace{U^\dagger S U}_{s} s^{-1/2} \\ &= s^{-1/2} s s^{-1/2} = \mathbb{1}, \end{aligned} \tag{8.44}$$

since U diagonalizes S in the sense that $U^\dagger S U = s$. Note also that $s^{-1/2}$ is real and diagonal, so $(s^{-1/2})^\dagger = s^{-1/2}$.

The implementation of the diagonalization of S and the creation of the transformation matrix V is done once, at the start of the SCF iterations. This is handled by the HartreeFock super-class, since the method is shared for both the restricted and un-restricted formalisms.

```
void HartreeFock::diagonalizeOverlapMatrix() {
    vec s;
    mat U;
    arma::eig_sym(s, U, m_overlapMatrix);
    m_transformationMatrix = U * arma::diagmat(1.0 / sqrt(s));
}
```

Diagonalizing the Fock matrix

Using the m_transformationMatrix, the Fock matrix can now be diagonalized by the RestrictedHartreeFock class as

```
void RestrictedHartreeFock::diagonalizeFockMatrix() {
    const mat& V        = m_transformationMatrix;
    const mat& F        = m_fockMatrix;
    mat&       Ftilde   = m_fockMatrixTilde;
    mat&       C        = m_coefficientMatrix;
    mat&       Ctilde   = m_coefficientMatrixTilde;

    Ftilde = V.t() * F * V;
    arma::eig_sym(m_epsilon, Ctilde, Ftilde);
    C = V * Ctilde.submat(0,
                          0,
                          m_numberOfBasisFunctions - 1,
                          m_numberOfElectrons / 2   - 1);
}
```

Since we are not interested in the virtual Hartree-Fock orbitals in the present work, we grab only the slice of the m_coefficientMatrix which corresponds to occupied orbitals.

Setting up the Fock matrix and computing the energy

Before we can diagonalize it, we first need to construct the Fock matrix. The elements of the Fock matrix consist of one-body and two-body integrals,

$$\begin{aligned} F_{pq} &= \langle p|\hat{h}|q\rangle + \sum_{r=1}^{L}\sum_{s=1}^{L}\sum_{k=1}^{N/2} C_{rk}C_{sk}\big(2\langle pr|\hat{w}|qs\rangle - \langle pq|\hat{w}|sq\rangle\big) \\ &= \langle p|\hat{h}|q\rangle + \sum_{r=1}^{L}\sum_{s=1}^{L} D_{sr}\big(2\langle pr|\hat{w}|qs\rangle - \langle pq|\hat{w}|sq\rangle\big), \end{aligned} \tag{8.45}$$

where L denotes the basis size, N the number of electrons, D the density matrix, and the one-body and two-body integrals are defined by (c.f. chapter 4)

$$\langle p|\hat{h}|q\rangle = \int \mathrm{d}^3\mathbf{r}\, \phi_p^*(\mathbf{r}) \left[-\frac{\nabla^2}{2} - \sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}-\mathbf{r}_A|}\right]\phi_q(\mathbf{r}), \tag{8.46}$$

and

$$\langle pq|\hat{w}|rs\rangle = \int \mathrm{d}^3\mathbf{r}_1 \int \mathrm{d}^3\mathbf{r}_2\, \phi_q^*(\mathbf{r}_1)\phi_q^*(\mathbf{r}_2)\frac{1}{|\mathbf{r}_1-\mathbf{r}_2|}\phi_q(\mathbf{r}_1)\phi_s(\mathbf{r}_2). \tag{8.47}$$

This is implemented in the Hartree-Fock framework as

```
void RestrictedHartreeFock::computeFockMatrix() {
    for(int p = 0; p < m_numberOfBasisFunctions; p++)
    for(int q = 0; q < m_numberOfBasisFunctions; q++) {
        m_fockMatrix(p,q) = m_oneBodyMatrixElements(p,q);

        for(int r = 0; r < m_numberOfBasisFunctions; r++)
        for(int s = 0; s < m_numberOfBasisFunctions; s++) {
            m_fockMatrix(p,q) += 0.5 *
                                 m_densityMatrix(s,r) *
                                 twoBodyMatrixElementsAntiSymmetric(p,q,r,s);
        }
    }
}
```

with the anti-symmetric matrix elements being $2\langle pq|\hat{w}|rs\rangle - \langle pq|\hat{w}|sr\rangle$.

After setting up and diagonalizing the Fock matrix, we can compute the Hartree-Fock energy, given by

$$E_{\mathrm{HF}} = \sum_{p=1}^{L}\sum_{q=1}^{L} D_{pq}\langle p|\hat{h}|q\rangle + \frac{1}{2}\sum_{pqrs} D_{pq}D_{rs}\left[\langle pr|\hat{w}|qs\rangle - \frac{1}{2}\langle pr|\hat{w}|sq\rangle\right]. \tag{8.48}$$

This is straight forwardly implemented as

```
void RestrictedHartreeFock::computeHartreeFockEnergy() {
    m_hartreeFockEnergy = 0;

    for (int p = 0; p < m_numberOfBasisFunctions; p++)
    for (int q = 0; q < m_numberOfBasisFunctions; q++) {
        m_hartreeFockEnergy += m_densityMatrix(p,q) *
                               m_oneBodyMatrixElements(p,q);

        for (int r = 0; r < m_numberOfBasisFunctions; r++)
        for (int s = 0; s < m_numberOfBasisFunctions; s++) {
            m_hartreeFockEnergy += 0.25 * m_densityMatrix(p,q) *
                                          m_densityMatrix(s,r) *
                                   twoBodyMatrixElementAntiSymmetric(p,q,r,s);
        }
    }
    m_hartreeFockEnergy += m_nucleusNucleusInteractionEnergy;
}
```

Updating the density matrix and possible convergence problems

Similar to other iterative schemes, the Hartree-Fock SCF iterations sometimes suffer from convergence problems. Naive, straight forward, SCF iterations in fact has

very problematic convergence properties, and one usually attempts to *help* it somehow [14]. The most commonly used such fix is called *direct inversion in the iterative subspace* (DIIS) or Pulay mixing and essentially uses the errors from the m previous iterations to extrapolate to a coefficient matrix C which hopefully minimizes said error [146]. A simpler scheme, which we implement in the present work, is called mixing: instead of updating the density fully at each iteration, a weighted average of the old and the newly computed D is taken to be the new density [57]. A mixing factor a is introduced and the updated density matrix is taken equal

$$D = aD_{\text{old}} + (1 - a)D_{\text{new}}, \tag{8.49}$$

where $D_{\text{new}} = 2C_{\text{new}}C_{\text{new}}^{\dagger}$. In the program, this is implemented as

```
void RestrictedHartreeFock::computeDensityMatrix() {
    if (m_smoothing) {
        double a = m_smoothingFactor;
        mat densityMatrixTmp = 2 * m_coefficientMatrix *
                                   m_coefficientMatrix.t();
        m_densityMatrix      =       a   * m_densityMatrix +
                               (1.0 - a) * densityMatrixTmp;
    } else {
        m_densityMatrix = 2 * m_coefficientMatrix * m_coefficientMatrix.t();
    }
}
```

Chapter 9

Implementation: Variational Monte Carlo

The following is a description of the implementation of the VMC framework described in chapter 6. The main body of the method consists of about $4\,000$ significant lines[1] of C++ code. It is object oriented and modular, and written to be as general as possible while still retaining execution speed. It consists of about 12 significant classes, with associated sub-classes, of which a generic user is required to interact with four in order to run simulations: The managing System class, the Atom class for setting up the chemical environment, and appropriate sub-classes of the WaveFunction and Orbital classes to choose which kind of wave function is to be used.

Wherever possible, the modularity makes it possible to in principle directly reuse the classes for different purposes. As an example, the Metropolis class—which handles the accept/reject Metropolis steps and generates a Markov chain of samples drawn from our PDF—can be reused to run e.g. a statistical mechanics simulation of the Ising spin model without changing more than a handful lines of code. The same is true of e.g. the Sampler class which handles sampling the local energy and computing averages, etc.: this class is reusable without changing *a single line of code*.

The developed code can perform multiple different calculations: Atomic or molecular systems in addition to harmonic oscillator quantum dot systems are supported. Various different Slater determinant types are available, such as *direct evaluation* determinant, or the more sophisticated and faster *inverse* determinant machinery. In either case, a relative distance-dependent Jastrow factor can be included. The orbitals which build the Slater can be chosen as either Slater-type orbitals (STO) or Gaussian-type orbitals (GTO). The latter can be taken from a Hartree-Fock basis computed using the code described in chapter 8, automatically parsed from the Hartree-Fock output by the HatreeFockBasisParser class.

However, the only parts of the code which have been rigorously tested and which

[1] As counted by the cloc program which counts *significant* lines of code, leaving out blank lines, comment lines, etc. [37]

will be described in the following are the molecular full inverse Slater machinery with a two-body Jastrow factor, filled with either STOs or GTOs.

We will start off with a few usage examples and then later expand on the implementation of some key classes and methods.

9.1 Introductory examples

The simplest usage of the VMC code requires only a few lines of C++ code:

```
int      Z        = 2;
double   alpha    = 1.843;
double   beta     = 0.347;
vec      position{0,0,0};

System He;
He.setImportanceSampling(true);
He.addCore            (new Atom                (&He, position, Z));
He.setWaveFunction    (new SlaterWithJastrow   (&He, beta));
He.setOrbital         (new HydrogenOrbital     (alpha));
He.runMetropolis      ((int) 1e7);
```

First the System instance is created, and importance sampling is enabled. Then a new Core is added: a charge-2 atom placed at the origin. The WaveFunction is selected as a standard SlaterWithJastrow which is then filled with HydrogenOrbitals. The α and β parameters are the variational input parameters in the hydrogenic radial wave functions and the Jastrow factor, respectively. The values of $\alpha = 1.843$ and $\beta = 0.347$ have been optimized for a single He atom.

Running the above code gives on the fly output shown in Fig. 9.1, and the final energy $E = -2.890 \pm 0.001 E_h$.

More complicated systems can also easily be set up for simulation, e.g. the following code runs a simulation on Ne^+ cation with non-interacting electrons, no Jastrow-factor, and a manually specified importance sampled step length δt:

```
System Ne;
Ne.setImportanceSampling     (true);
Ne.setElectronInteraction    (false);
Ne.setStepLength      (0.025);
Ne.addCore            (new Atom                (&Ne, position, 10, 5, 4));
Ne.setWaveFunction    (new SlaterWithJastrow   (&Ne, -1, false));
Ne.setOrbital         (new HydrogenOrbital     (10.0));
Ne.runMetropolis      ((int) 1e7);
```

The penultimate input parameter of the Atom constructor defines the number of spin-up electrons, while the last parameter gives the number of spin-down electrons. As the total number of electrons is 9, while the nucleus charge is $Z = 10$, this defines a Ne^+ ion. Since the hydrogenic Slater determinant forms the *exact* wave function for non-interacting electrons, the energy $E = -187.5 E_h$ is calculated with vanishing variance.

```
=============== Starting Metropolis Algorithm ==============
=> Number of steps:        1e+07
=> Number of dimensions:   3
=> Number of electrons:    2
=> Step length:            1
=> Number of cores:        1
     ------------------------------------------------------
     | He                    (0.000, 0.000, 0.000)          |
     ------------------------------------------------------

======================================================================
                Total                                Block
 Step       Energy      Std.dev.       Energy      Variance     Acc. rate
----------------------------------------------------------------------
  5e4       -2.8894       0.13448      -2.8764   3.4756e-05        0.9976
  1e5       -2.8825      0.069696      -2.8367   3.7517e-05        0.9976
  1e5       -2.8814      0.047099      -2.9346   6.4217e-05        0.9964
  2e5       -2.8852      0.035672      -2.8164   0.00011602        0.9976
  2e5       -2.8851       0.02869      -2.8837   6.4744e-05         0.994
  3e5       -2.8878      0.024047      -2.9387   0.00012301        0.9952

                                  ...

  9e6       -2.8896     0.0010202      -2.8407   5.2671e-05        0.9944
  9e6       -2.8895     0.0010167      -2.8228   3.3053e-05        0.9972
  9e6       -2.8895      0.001013       -2.897   4.0035e-05        0.9968
  9e6       -2.8896     0.0010092      -2.8771   5.1971e-05         0.994
  9e6       -2.8896      0.001005      -2.8623    3.122e-05         0.998
======================================================================

Metropolis Algorithm finished.

=> Metropolis steps:                              1e+07
=> Final acceptance rate:              0.9967614900486612
=> Final energy average:               -2.889522415923047
=> Final variance:                   1.002073906587686e-06
============================================================
```

Figure 9.1: Output of the first example program show at the start of section 9.1. The first two Total-columns show the full energy computed so far, treating blocks of 2500 Monte Carlo step as single samples. The three right hand side Block-columns show the values computed for the last completed such block. Please note that the block variance shown is treating every single Monte Carlo cycle as an independent sample, and thus massively underestimates the size of the variance. The same is also true of the printed Final variance at the very end of the output.

Cartesian Gaussian orbitals can be requested by calling System::setOrbital with a SlaterTypeOrbital object. The name of a basis file—output from the Hatree-Fock code described in detail in section 8, defining the chemical environment and the basis set itself—is given as input to the constructor of SlaterTypeOrbital. The basis file is the result of previously run Hartree-Fock calculations, and defines the positions of any atoms present. This means there is no need to specify nucleonic positions when SlaterTypeOrbitals are used. An example calculation can be ran by the following code:

```
string   basisFileName = "Be-STO-3G";
System   Be;
Be.setImportanceSampling(true);
Be.setWaveFunction   (new SlaterWithJastrow   (&Be, beta, true));
Be.setOrbital        (new GaussianOrbital     (&Be, basisFileName));
Be.runMetropolis     ((int) 1e7);
```

Here, a STO-3G Gaussian Hartree-Fock basis set is used.

9.2 Overview of select classes

9.2.1 The SlaterWithJastrow class

The work-horse of the VMC program is the WaveFunction class and associated sub-classes. The code can in principle be run with any trial wave function, as long as it can be evaluated, and allows the computation of the Laplacian (and the gradient if importance sampling is desired).

The particular sub-class used in the entirety of the current work is the SlaterWithJastrow class. It represents a product of a Slater determinant $|D(\mathbf{R})|$ and a two-body Jastrow factor $J(\mathbf{R})$,

$$\Psi_{\mathrm{T}}(\mathbf{R}) = |D(\mathbf{R})|J(\mathbf{R}). \tag{9.1}$$

The determinant is populated with orbitals represented by the Orbital class. When using a *restricted* set of orbitals—in the sense that each *spatial* orbital is doubly occupied by one spin-up and one spin-down electron—the full Slater determinant

$$|D(\mathbf{R})| = \frac{1}{\sqrt{N!}} \begin{vmatrix} \tilde{\phi}_1(\mathbf{r}_1) & \tilde{\phi}_2(\mathbf{r}_1) & \tilde{\phi}_3(\mathbf{r}_1) & \dots & \tilde{\phi}_N(\mathbf{r}_1) \\ \tilde{\phi}_1(\mathbf{r}_2) & \tilde{\phi}_2(\mathbf{r}_2) & \tilde{\phi}_3(\mathbf{r}_2) & \dots & \tilde{\phi}_N(\mathbf{r}_2) \\ \tilde{\phi}_1(\mathbf{r}_3) & \tilde{\phi}_2(\mathbf{r}_3) & \tilde{\phi}_3(\mathbf{r}_3) & \dots & \tilde{\phi}_N(\mathbf{r}_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\phi}_1(\mathbf{r}_N) & \tilde{\phi}_2(\mathbf{r}_N) & \tilde{\phi}_3(\mathbf{r}_N) & \dots & \tilde{\phi}_N(\mathbf{r}_N) \end{vmatrix} \tag{9.2}$$

is *singular*. The restricted orbitals $\tilde{\phi}(\mathbf{r})$ are spatially pairwise equal, odd indices carry spin-up while even indices carry spin-down. This means we can write the determi-

nant in terms of *spatial* orbitals $\phi_k(\mathbf{r})$ and $\phi_k(\mathbf{r})$ as

$$|D(\mathbf{R})| = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_{1\uparrow}(\mathbf{r}_1) & \phi_{1\downarrow}(\mathbf{r}_1) & \phi_{2\uparrow}(\mathbf{r}_1) & \phi_{2\downarrow}(\mathbf{r}_1) & \dots & \phi_{N/2\downarrow}(\mathbf{r}_1) \\ \phi_{1\uparrow}(\mathbf{r}_2) & \phi_{1\downarrow}(\mathbf{r}_2) & \phi_{2\uparrow}(\mathbf{r}_2) & \phi_{2\downarrow}(\mathbf{r}_2) & \dots & \phi_{N/2\downarrow}(\mathbf{r}_2) \\ \phi_{1\uparrow}(\mathbf{r}_3) & \phi_{1\downarrow}(\mathbf{r}_3) & \phi_{2\uparrow}(\mathbf{r}_3) & \phi_{2\downarrow}(\mathbf{r}_3) & \dots & \phi_{N/2\downarrow}(\mathbf{r}_3) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{1\uparrow}(\mathbf{r}_N) & \phi_{1\downarrow}(\mathbf{r}_N) & \phi_{2\uparrow}(\mathbf{r}_N) & \phi_{2\downarrow}(\mathbf{r}_N) & \dots & \phi_{N/2\downarrow}(\mathbf{r}_N) \end{vmatrix}, \tag{9.3}$$

where $\phi_{k\uparrow}(\mathbf{r}) = \phi_k(\mathbf{r})\chi(\uparrow) = \tilde{\phi}_{2k-1}(\mathbf{r})$ and $\phi_{k\downarrow}(\mathbf{r}) = \phi_k(\mathbf{r})\chi(\downarrow) = \tilde{\phi}_{2k}(\mathbf{r})$. The χs represent spin-$1/2$ spinors. It can be shown that for a spin-independent operator, such as all the Hamiltonians in the present work, the expectation value

$$E_{\mathrm{T}} = \frac{\langle \Psi_{\mathrm{T}} | \hat{H} | \Psi_{\mathrm{T}} \rangle}{\langle \Psi_{\mathrm{T}} | \Psi_{\mathrm{T}} \rangle}, \tag{9.4}$$

is invariant under a splitting of the determinant. We can split the Slater in two factors: one representing the spatial orbitals for the spin-up electrons, and one representing the corresponding orbitals for the spin-down electrons. Even though the new wave function is no longer*anti-symmetric* w.r.t. interchange of two electrons of opposite spin, the expectation value—all we care about—remains the same, with the added benefit of reducing computational cost [68].

The full split-determinant trial wave function takes the form

$$\Psi_{\mathrm{T}} = |D_\uparrow(\mathbf{R})||D_\downarrow(\mathbf{R})|J(\mathbf{R}), \tag{9.5}$$

where

$$|D_\uparrow(\mathbf{R})| \propto \begin{vmatrix} \phi_{1\uparrow}(\mathbf{r}_1) & \phi_{2\uparrow}(\mathbf{r}_1) & \phi_{3\uparrow}(\mathbf{r}_1) & \dots & \phi_{N/2\uparrow}(\mathbf{r}_1) \\ \phi_{1\uparrow}(\mathbf{r}_2) & \phi_{2\uparrow}(\mathbf{r}_2) & \phi_{3\uparrow}(\mathbf{r}_2) & \dots & \phi_{N/2\uparrow}(\mathbf{r}_2) \\ \phi_{1\uparrow}(\mathbf{r}_3) & \phi_{2\uparrow}(\mathbf{r}_3) & \phi_{3\uparrow}(\mathbf{r}_3) & \dots & \phi_{N/2\uparrow}(\mathbf{r}_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{1\uparrow}(\mathbf{r}_{N/2}) & \phi_{2\uparrow}(\mathbf{r}_{N/2}) & \phi_{3\uparrow}(\mathbf{r}_{N/2}) & \dots & \phi_{N/2\uparrow}(\mathbf{r}_{N/2}) \end{vmatrix}, \tag{9.6}$$

and a corresponding expression for $|D_\downarrow(\mathbf{R})|$. Since we are always working with ratios of wave functions in the Metropolis-Hastings algorithm, the normalization factor of the determinant does not enter our equations and we can safely forget about it from now on.

Evaluating the wave function ratio

In order to perform the Metropolis test step of the Metropolis-Hastings algorithm, we need to be able to calculate the ratio

$$R = \frac{\Psi_{\mathrm{T}}(\mathbf{R}_{\mathrm{new}})}{\Psi_{\mathrm{T}}(\mathbf{R}_{\mathrm{old}})}. \tag{9.7}$$

We may of course directly evaluate the determinant at every Monte Carlo step. After the splitting, we can rewrite R in a more convenient form:

$$R = \left[\frac{|D_\uparrow(\mathbf{R}_{\text{new}})||D_\downarrow(\mathbf{R}_{\text{new}})|}{|D_\uparrow(\mathbf{R}_{\text{old}})||D_\downarrow(\mathbf{R}_{\text{old}})|}\right] \frac{J(\mathbf{R}_{\text{new}})}{J(\mathbf{R}_{\text{old}})} \equiv R_{\text{SD}} R_{\text{J}}. \tag{9.8}$$

From this it is immediately obvious that if the new coordinate set $\mathbf{R}_{\text{new}}$ differs from the old $\mathbf{R}_{\text{old}}$ for *only a single electron* of spin $\uparrow$ ($\downarrow$) then the spin-down (spin-up) determinants falls out of Eq. (9.8). In other words: moving only one electron at the time the ostensibly halves the required computation cost associated with R_{SD}. This of course comes at the cost of correlation—subsequent samples are less correlated if we simultaneously move multiple electrons—a complication which we will return to shortly.

Even after the splitting however, the direct evaluation of our determinants still requires $\mathcal{O}(N^3)$ operations, albeit with a pre-factor $1/8$ compared to the original full determinant. It turns out that we can do better.

Consider the terms of the Slater *matrix*: $D_{ij}(\mathbf{r}) \equiv \phi_j(\mathbf{r}_i)$. The usual Laplace-expansion of the determinant is defined as

$$|D| = \sum_{j=1}^{N} D_{ij} C_{ij}, \tag{9.9}$$

where C_{ij} is the i,j *cofactor* of D, i.e. the determinant of the sub-matrix with row i and column j removed multiplied by $(-1)^{i+j}$ [58]. The determinant of the sub-matrix is called the i,j *minor* of D. By using *Cramer's rule*[2] we can find an explicit expression for the matrix inverse in terms of the determinant as [43]

$$D^{-1} = \frac{\text{adj}D}{|D|}. \tag{9.11}$$

The *adjugate* matrix[3], adj$D$, is simply the transposed matrix of cofactors. In terms of the entries, we can write

$$|D| = \sum_{j=1}^{N} \frac{C_{ji}}{D^{-1}_{ij}} = \sum_{j=1}^{n} D_{ij} C_{ji}. \tag{9.12}$$

[2]Cramer's rule states that for any inertible $n \times n$ matrix A and $\mathbf{b} \in \mathbb{R}^n$, the unique solution $\mathbf{x}$ of the matrix-vector equation $A\mathbf{x} = \mathbf{b}$ has entries

$$x_i = \frac{|A_i(\mathbf{b})|}{|A|}, \quad \text{where} \quad i = 1, 2, \ldots, n, \tag{9.10}$$

where $A_i(\mathbf{b})$ is the matix formed by replacing the i-th column of A by $\mathbf{b}$ [58].

[3]Sometimes, rather confusingly, called the *adjoint*. In more modern terminology, the *adjoint* is reserved for the complex conjugate-transpose, while the transposed cofactor matrix is called the *adjugate* or the *classical adjoint* [68, 147].

As only one electron is moved at each Monte Carlo step, only a single row of the Slater matrix changes at each cycle. Recall that the i, j cofactors are determinants of the sub-matrix resulting from removing column i and row j from D. This means that as row i of D is changed, the i-th column of the adjugate remains unchanged. In short, $C_{ij}(\mathbf{r}_{\text{new}}) = \text{adj}D_{ij}(\mathbf{r}_{\text{old}}) = C_{ji}(\mathbf{r}_{\text{old}})$ [68].

By definition, the Slater matrix and it's inverse must satisfy

$$\sum_{k=1}^{N} D_{ik} D_{kj}^{-1} = \delta_{ij}, \tag{9.13}$$

meaning the denominator drops out of Eq. (9.14) and R_{SD} simplifies immensely to [17, 68, 148]

$$\begin{aligned} \frac{|D(\mathbf{R}_{\text{new}})|}{|D(\mathbf{R}_{\text{old}})|} = R_{\text{SD}} &= \frac{\sum_{j=1}^{N} D_{ij}(\mathbf{r}_{\text{new}}) C_{ji}(\mathbf{r}_{\text{old}})}{\sum_{j=1}^{N} D_{ij}(\mathbf{r}_{\text{old}}) C_{ji}(\mathbf{r}_{\text{old}})} \\ &= \frac{\sum_{j=1}^{N} D_{ij}(\mathbf{r}_{\text{new}}) D_{ji}^{-1}(\mathbf{r}_{\text{old}}) |D(\mathbf{r}_{\text{old}})|}{\sum_{j=1}^{N} D_{ij}(\mathbf{r}_{\text{old}}) D_{ji}^{-1}(\mathbf{r}_{\text{old}}) |D(\mathbf{r}_{\text{old}})|} \qquad (9.14) \\ &= \sum_{j=1}^{N} D_{ij}(\mathbf{r}_{\text{new}}) D_{ji}^{-1}(\mathbf{r}_{\text{old}}). \qquad (9.15) \end{aligned}$$

Note carefully that the inverse need only be re-calculated *if* the new configuration is accepted in the Metropolis test.

Eq. (9.15) is implemented in the VMC code as

```
void SlaterWithJastrow::computeSlaterRatio() {
    Electron* iElectron = m_system->getElectrons().at(m_changedElectron);
    int       i         = iElectron->getSpinIndex();
    int       nElectrons = (m_spinChanged == 1 ? m_numberOfSpinUpElectrons :
                                                 m_numberOfSpinDownElectrons);
    double    xi        = iElectron->getPosition().at(0);
    double    yi        = iElectron->getPosition().at(1);
    double    zi        = iElectron->getPosition().at(2);
    mat&      slater    = (m_spinChanged == 1 ? m_slaterUp : m_slaterDown);

    double sum = 0;
    for (int j = 0; j < nElectrons; j++) {
        sum += m_orbital->evaluate(xi,yi,zi,j,m_spinChanged) * slater(j,i);
    }
    m_Rsd = sum;
}
```

The m_changedElectron is communicated to the WaveFunction by the Metropolis class as it suggets a step, and m_spinChanged—the spin of the moved electron—is subsequently found. Depending on this spin projection, we index into either m_slaterUp or m_slaterDown according to the *spin index* of the moved electron. All electrons have a unique global indentifying index k, but they also have a local place in the spin-up (spin-down) determinant: This is what we denote by the spin-index.

The m_orbital variable holds an instance of the class corresponding to the spin-orbitals which populate the Slater determinant. We give an outline of the Orbital in section 9.2.2.

Updating the inverse

The algorithm of the previous section requires the inverse of the Slater matrix, evaluated at the previous electronic configuration, to be know. In principle we may simply directly evaluate the inverse for every Monte Carlo cycle, but the $\mathcal{O}(N^3)$ computational scaling cost quickly makes this approach unfeasible. We require therefore a more efficient algorithm which makes use of the fact that when we update it, the inverse is already known at the old configuration. Recall that the new Slater matrix differs from the old one only in a single row. In the eloquent words of William H. Press and co-workers in the third edition of *Numerical Recipes* [120]

> “Suppose you have already obtained, by herculean effort, the inverse matrix A^{-1} of a square matrix A. Now you want to make a "small" change in A, for example change (...) one row, or one column. Is there any way of calculating the corresponding A^{-1} without repeating your difficult labors?”
>
> *W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery*

It turns out that such a convenient formula exists. It is known as the *Sherman-Morrison formula*. The original formulation due to Sherman and Morrison deals with the problem of updating the inverse of a matrix, given a change in a single element. However, what is normally referred to as *the Sherman-Morrison formula* (and indeed what we will be referring to by that name) is a straightforward extension of this. The formula states the following: Suppose a square $n \times n$ matrix A and it's inverse A^{-1} is (by heroic effort) known. Assume $\mathbf{u}$ and $\mathbf{v}$ are elements in $\mathbb{R}^n$, then the matrix inverse of $A + \mathbf{u}\mathbf{v}^T$ is given by

$$\left(A + \mathbf{u}\mathbf{v}^T\right)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}}. \tag{9.16}$$

Note that $\mathbf{u}\mathbf{v}^T$ (sometimes dentoed $\mathbf{u} \otimes \mathbf{v}$) is the *outer* product, and in our case $\mathbf{u}$ is the k-th unit vector, with k corresponding to the index of the moved electron. The vector $\mathbf{v}$ contains the change in the orbitals for the moved electron,

$$v_i = \phi_i(\mathbf{r}_k^{\text{new}}) - \phi_i(\mathbf{r}_k^{\text{old}}) \equiv \Delta\phi_i(\mathbf{r}_k). \tag{9.17}$$

In short, $\mathbf{u}\mathbf{v}^T$ is a matrix with vanishing elements in all but a single row—row number k—which contains the change in the Slater matrix due to the moving of a single electron.

The Sherman-Morrison formula can be derived as follows: Finding the matrix inverse of $(A + \mathbf{u}\mathbf{v}^T)$ constitutes finding a vector $\mathbf{x}$ such that $(A + \mathbf{u}\mathbf{v}^T)\mathbf{x} = \mathbf{y}$ is satisfied for some given $\mathbf{y}$. Expanding and defining $s \equiv \mathbf{v}^T\mathbf{x}$, we find [149]

$$\begin{aligned} \left(A + \mathbf{u}\mathbf{v}^T\right)\mathbf{x} &= \mathbf{y} \\ A\mathbf{x} &= \mathbf{y} - \mathbf{u}\mathbf{v}^T\mathbf{x} \\ \mathbf{x} &= A^{-1}\mathbf{y} - A^{-1}\mathbf{u}\underbrace{\mathbf{v}^T\mathbf{x}}_{=s}. \end{aligned} \tag{9.18}$$

Insertion of $\mathbf{x}$ into the the expression for s yields

$$\begin{aligned} s &= \mathbf{v}^T\mathbf{x} \\ &= \mathbf{v}^T A^{-1}\mathbf{y} - \mathbf{v}^T A^{-1}\mathbf{u}s \\ s\left(1 + \mathbf{v}^T A^{-1}\mathbf{u}\right) &= \mathbf{v}^T A^{-1}\mathbf{y} \\ s &= \frac{\mathbf{v}^T A^{-1}\mathbf{y}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}}. \end{aligned} \tag{9.19}$$

Substituting finally Eq. (9.19) into the previous expression for $\mathbf{x}$ gives

$$\begin{aligned} \mathbf{x} &= A^{-1}\mathbf{y} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}\mathbf{y}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}} \\ &= \Bigg(\underbrace{A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}}}_{(A + \mathbf{u}\mathbf{v}^T)^{-1}}\Bigg)\mathbf{y}. \end{aligned} \tag{9.20}$$

In terms of the matrix entires, we have

$$\left(A + \mathbf{u}\mathbf{v}^T\right)^{-1}_{kj} = A^{-1}_{kj} - \frac{A^{-1}_{ki}\left(\mathbf{u}\mathbf{v}^T\right)_{il} A^{-1}_{lj}}{1 + \lambda}, \tag{9.21}$$

where $\lambda \equiv 1 + \mathbf{v}^T A^{-1}\mathbf{u}$. The index i of the recently displaced electron dictate which row of $\mathbf{u}\mathbf{v}^T$ takes non-zero values. Inserting $v_a = \Delta\phi_a(\mathbf{r}_i)$, $A^{-1}_{bc} = D^{-1}_{bc}(\mathbf{r}^{\text{old}})$, and $u_d = \delta_{id}$, Hammond finds [17]

$$D^{-1}_{kj}(\mathbf{r}_{\text{new}}) = \begin{cases} D^{-1}_{kj}(\mathbf{r}_{\text{old}}) - \dfrac{1}{R_{\text{SD}}} D^{-1}_{ji}(\mathbf{r}_{\text{new}}) \displaystyle\sum_{l=1}^{n} D_{il}(\mathbf{r}_{\text{new}}) D^{-1}_{lj}(\mathbf{r}_{\text{old}}) & \text{if} \quad j \neq i \\ \dfrac{1}{R_{\text{SD}}} D^{-1}_{kj}(\mathbf{r}_{\text{old}}) & \text{if} \quad j = i \end{cases} \tag{9.22}$$

where $D^{-1}_{kj}(\mathbf{r}_{\text{new}})$ corresponds to $(A + \mathbf{u}\mathbf{v}^T)^{-1}$ of Eq. (9.20) and

$$R_{\text{SD}} = \frac{|D(\mathbf{R}_{\text{new}})|}{|D(\mathbf{R}_{\text{old}})|} = \sum_{j=1}^{n} D_{ij}(\mathbf{r}_{\text{new}}) D^{-1}_{ji}(\mathbf{r}_{\text{old}}) \tag{9.23}$$

as per Eq. (9.15).

The modified Sherman-Morrison scheme of Eq. (9.22) is implemented in the VMC framework as

```
void SlaterWithJastrow::updateSlaterInverse() {
    Electron*  iElectron  = m_system->getElectrons().at(m_changedElectron);

    const int    i   = iElectron->getSpinIndex();
    const int    sc  = m_spinChanged;
    const double x   = iElectron->getPosition().at(0);
    const double y   = iElectron->getPosition().at(1);
    const double z   = iElectron->getPosition().at(2);

    mat& newS        = (sc == 1 ? m_slaterUp     : m_slaterDown);
    mat& oldS        = (sc == 1 ? m_slaterUpOld  : m_slaterDownOld);
    int  nElectrons  = (sc == 1 ? m_numberOfSpinUpElectrons :
                                  m_numberOfSpinDownElectrons);

    for (int k = 0; k < nElectrons; k++) {
        for (int j = 0; j < nElectrons; j++) {
            if (j != i) {
                double sum = 0;
                for (int l = 0; l < nElectrons; l++) {
                    sum += oldS(l,j) * m_orbital->evaluate(x,y,z,l,sc);
                }
                newS(k,j) = oldS(k,j) - oldS(k,i) * sum / m_Rsd;
            } else {
                newS(k,j) = oldS(k,i) / m_Rsd;
            }
        }
    }
}
```

The Slater matrix evaluated at the old configuration is contained in the variables m_slaterUpOld and m_slaterDownOld. Note carefully that the R_{SD} value stored in m_Rsd here is the *updated* value, which is independent of $D^{-1}(\mathbf{R}_{\mathrm{new}})$ and so can be calculated before we update the inverse in this member function.

At the very start of the Metropolis run, the full Slater matrix needs to be calculated and inverted *once*. This is done (in addition to a whole range of other operations) in SlaterWithJastrow::evaluateWaveFunctionInitial:

```
void SlaterWithJastrow::evaluateWaveFunctionInitial() {
    ...
    const int eUp   = m_numberOfSpinUpElectrons;
    const int eDown = m_numberOfSpinDownElectrons;
    m_slaterUp      = zeros<mat>(eUp,   eUp);
    m_slaterDown    = zeros<mat>(eDown, eDown);

    vector<Electron*> spinUpElectrons   = m_system->getSpinUpElectrons();
    vector<Electron*> spinDownElectrons = m_system->getSpinDownElectrons();

    for (int electron = 0; electron < eUp; electron++) {
        const double x = spinUpElectrons.at(electron)->getPosition().at(0);
        const double y = spinUpElectrons.at(electron)->getPosition().at(1);
        const double z = spinUpElectrons.at(electron)->getPosition().at(2);

        for (int basis = 0; basis < eUp; basis++) {
            m_slaterUp(electron,basis) = m_orbital->evaluate(x,y,z,basis,1);
```

```
        }
    }
    for (int electron = 0; electron < eDown; electron++) {
        const double x = spinDownElectrons.at(electron)->getPosition().at(0);
        const double y = spinDownElectrons.at(electron)->getPosition().at(1);
        const double z = spinDownElectrons.at(electron)->getPosition().at(2);

        for (int basis = 0; basis < eDown; basis++) {
            m_slaterDown(electron,basis) = m_orbital->evaluate(x,y,z,basis,0);
        }
    }

    m_slaterUp   = m_slaterUp.i();
    m_slaterDown = m_slaterDown.i();
    ...
}
```

Efficient evaluation of the gradient, $\nabla_i D(\mathbf{R})$

The calculation of the quantum force involved in the importance sampling of the Metropolis-Hastrings algorithm requires the evaluation of the ratio of the gradient to the wave function itself. Since, by repeated application of the product rule,

$$\begin{aligned}
\frac{\nabla_i \Psi_\mathrm{T}}{\Psi_\mathrm{T}} &= \frac{\nabla_i \left[|D_\downarrow(\mathbf{R})||D_\uparrow(\mathbf{R})|J(\mathbf{R})\right]}{\Psi_\mathrm{T}} \\
&= \frac{J(\mathbf{R})\nabla_i \left[|D_\downarrow(\mathbf{R})||D_\uparrow(\mathbf{R})|\right] + |D_\downarrow(\mathbf{R})||D_\uparrow(\mathbf{R})|\nabla_i J(\mathbf{R})}{\Psi_\mathrm{T}} \\
&= \frac{J(\mathbf{R})|D_\downarrow(\mathbf{R})|\nabla_i|D_\uparrow(\mathbf{R})|}{\Psi_\mathrm{T}} + \frac{J(\mathbf{R})|D_\uparrow(\mathbf{R})|\nabla_i|D_\downarrow(\mathbf{R})|}{\Psi_\mathrm{T}} + \\
&\qquad \frac{|D_\downarrow(\mathbf{R})||D_\uparrow(\mathbf{R})|\nabla_i J(\mathbf{R})}{\Psi_\mathrm{T}} \\
&= \frac{\nabla_i|D_\uparrow(\mathbf{R})|}{|D_\uparrow(\mathbf{R})|} + \frac{\nabla_i|D_\downarrow(\mathbf{R})|}{|D_\downarrow(\mathbf{R})|} + \frac{\nabla_i J(\mathbf{R})}{J(\mathbf{R})},
\end{aligned} \tag{9.24}$$

we require an efficient algorithm for calculating $\nabla_i|D(\mathbf{R})|/|D(\mathbf{R})|$ terms. The indexed del operator ∇_i denotes differentiation w.r.t. the coordinates of electron i. We note that if said electron has spin projection $\sigma_i = \chi(\uparrow)$, then $\nabla_i|D_\downarrow|$ neccessaily vanishes. The same is true of $\nabla_i|D_\uparrow|$ if the spin state is flipped. This means we only ever need to calculate one such term for every Monte Carlo move.

A similar derivation to the one resulting in R_SD gives an expression for $\nabla_i|D(\mathbf{R})|$ in terms of the inverse Slater matrix as [68]

$$\frac{\nabla_i|D(\mathbf{R}_\mathrm{old})|}{|D(\mathbf{R}_\mathrm{old})|} = \sum_{j=1}^{n} \nabla_i D_{ij}(\mathbf{r}_\mathrm{old}) D^{-1}_{ji}(\mathbf{r}_\mathrm{old}) = \sum_{j=1}^{n} \nabla_i \phi_j(\mathbf{r}_i^\mathrm{old}) D^{-1}_{ji}(\mathbf{r}^\mathrm{old}), \tag{9.25}$$

where $D(\mathbf{R})$ denotes either one of $D_\uparrow$ or $D_\downarrow$ (whichever contains electron i). When

the gradient evaluated at the new positions is needed, the equation changes to [17]

$$\frac{\nabla_i |D(\mathbf{R}_{\text{new}})|}{|D(\mathbf{R}_{\text{new}})|} = \frac{|D(\mathbf{R}_{\text{old}})|}{|D(\mathbf{R}_{\text{new}})|} \sum_{j=1}^{n} \nabla_i \phi_j(\mathbf{r}_i^{\text{new}}) D_{ji}^{-1}(\mathbf{r}^{\text{old}})$$
$$= \frac{1}{R_{\text{SD}}} \sum_{j=1}^{n} \nabla_i \phi_j(\mathbf{r}_i^{\text{new}}) D_{ji}^{-1}(\mathbf{r}^{\text{old}}). \tag{9.26}$$

The implementation of Eq. (9.26) is straightforward:

```
void SlaterWithJastrow::updateSlaterGradient(double Rsd, int electron) {
    Electron*   iElectron    = m_system->getElectrons().at(electron);
    int cs                   = iElectron->getSpin();
    int i                    = iElectron->getSpinIndex();
    int nElectron            = (cs == 1 ? m_numberOfSpinUpElectrons :
                                          m_numberOfSpinDownElectrons);
    mat& slaterInverse  = (cs == 1 ? m_slaterUp          : m_slaterDown);
    mat& slaterGradient = (cs == 1 ? m_slaterGradientUp :
                                          m_slaterGradientDown);

    const double x = iElectron->getPosition().at(0);
    const double y = iElectron->getPosition().at(1);
    const double z = iElectron->getPosition().at(2);

    for (int dimension = 0; dimension < m_numberOfDimensions; dimension++) {
        double sum = 0;
        for (int j = 0; j < nElectrons; j++) {
            sum += m_orbital->computeDerivative(x,y,z,j,dimension,cs) *
                   slaterInverse(j,i);
        }
        slaterGradient(i, dimension) = sum / Rsd;
    }
}
```

Please note that instead of using directly the m_changedElectron index and the local m_Rsd value already calculated, Rsd and the electron index is given as parameters to SlaterWithJastrow::updateSlaterGradient. This is done in order to make it possible to compute the entire gradient matrix—the gradient w.r.t. all the electron coordinates in turn—by simply iterating over i and providing Rsd=1. Computing from scratch the entire gradient matrix is done exactly once in the VMC implementation, at the very start of the Metropolis sampling. Subsequent calculations are done by updating only the row corresponding to the moved electrons.

Efficient evaluation of the Laplacian, $\nabla_i^2 D(\mathbf{R})$

Whereas the gradient is needed for the evaluation of the quantum force, the Laplacian is needed in order to calculate the kinetic energy. In the same way we split the total gradient, we may split the total Laplacian into terms corresponding to differentiation of either $D_{\downarrow}(\mathbf{R})$, $D_{\uparrow}(\mathbf{R})$, $J(\mathbf{R})$. By, again, repeated application of the product rule,

we find that

$$\begin{aligned}
\frac{\nabla_i^2\Psi_\mathrm{T}}{\Psi_\mathrm{T}} &= \frac{\nabla_i\cdot\nabla_i\Psi_\mathrm{T}}{\Psi_\mathrm{T}} \\
&= \frac{1}{\Psi_\mathrm{T}}\nabla_i\cdot\Big[J|D_\downarrow|\nabla_i|D_\uparrow| + J|D_\uparrow|\nabla_i|D_\downarrow| + \\
&\qquad\qquad |D_\downarrow||D_\uparrow|\nabla_i J\Big] \\
&= \frac{1}{\Psi_\mathrm{T}}\Big[|D_\downarrow|\nabla_i|D_\uparrow|\cdot\nabla_i J + J\nabla_i|D_\downarrow|\cdot\nabla_i|D_\uparrow| + J|D_\downarrow|\nabla_i^2|D_\uparrow| + \\
&\qquad |D_\uparrow|\nabla_i|D_\downarrow|\cdot\nabla_i J + J\nabla|D_\uparrow|\cdot\nabla_i|D_\downarrow| + J|D_\uparrow|\nabla_i^2|D_\downarrow| + \\
&\qquad |D_\uparrow|\nabla_i J\cdot\nabla_i|D_\downarrow| + |D_\downarrow|\nabla_i|D_\uparrow\cdot|\nabla_i J + |D_\downarrow||D_\uparrow|\nabla_i^2 J\Big] \\
&= \frac{\nabla_i^2|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i^2|D_\downarrow|}{|D_\downarrow|} + \frac{\nabla_i^2 J}{J} + 2\left[\frac{\nabla_i|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i|D_\downarrow|}{|D_\downarrow|}\right]\cdot\frac{\nabla_i J}{J},
\end{aligned} \tag{9.27}$$

where we have supressed the arguments $\mathbf{R}$. As we can see, the computation of the Laplacian involves expressions on the form $\nabla_i^2|D|/|D|$. Analogous to the gradient expression, we can write out the Laplacian in terms of the inverse Slater matrix as [68]

$$\frac{\nabla_i^2|D(\mathbf{R}^\mathrm{new})|}{|D(\mathbf{R}^\mathrm{new})|} = \sum_{j=1}^{N}\nabla_i^2 D_{ij}(\mathbf{r}^\mathrm{new})D_{ji}^{-1}(\mathbf{r}^\mathrm{new}) = \sum_{j=1}^{N}\nabla_i^2\phi_j(\mathbf{r}_i^\mathrm{new})D_{ji}^{-1}(\mathbf{r}^\mathrm{new}). \tag{9.28}$$

Since the Laplacian is only ever computed *after* an accepted Metropolis step, we need no corresponding expression for $\nabla_i^2|D(\mathbf{R}_\mathrm{old})|/|D(\mathbf{R}_\mathrm{old})|$.

Eq. (9.28) is implemented in the VMC framework as

```
void SlaterWithJastrow::computeSlaterLaplacian(int electron) {
    Electron* kElectron  = m_system->getElectrons().at(electron);
    int kSpin       = kElectron->getSpin();
    int nElectrons = (kSpin == 1 ? m_numberOfSpinUpElectrons :
                                   m_numberOfSpinDownElectrons);
    const vector<Electron*>& electrons = (kSpin==1 ?
                                          m_system->getSpinUpElectrons() :
                                          m_system->getSpinDownElectrons());
    const mat& slater = (kSpin==1 ? m_slaterUp : m_slaterDown);
    double& slaterLaplacian = (kSpin==1 ? m_slaterLaplacianUp :
                                          m_slaterLaplacianDown);
    double value = 0;
    for (int i = 0; i < nElectrons; i++) {
        Electron* iElectron = electrons.at(i);
        const double xi = iElectron->getPosition().at(0);
        const double yi = iElectron->getPosition().at(1);
        const double zi = iElectron->getPosition().at(2);

        for (int j = 0; j < nElectrons; j++) {
            double jLaplacian = m_orbital->computeLaplacian(xi,yi,zi,j,kSpin);
            value +=  slater(j,i) * jLaplacian;
```

```
        }
    }
    slaterLaplacian = value;
    m_slaterLaplacian = m_slaterLaplacianUp + m_slaterLaplacianDown;
}
```

The Jastrow ratio

The correlation part of the VMC trial wave function is stored in the matrix m_correlationMatrix, which holds the value of

$$u_{ij} = \frac{a_{ij} r_{ij}}{1 + \beta r_{ij}}, \tag{9.29}$$

where a_{ij} equals $1/2$ ($1/4$) for opposite (parallel) spins and β is a variational parameter (c.f. section 3.2). The inter-electronic distance is denoted $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$. Only the upper diagonal part of the u matrix needs to be filled, since the correlations are obviously symmetric relations, i.e. $u_{ij} = u_{ji}$. The same is true of r_{ij}, the values of which are stored in a matrix called m_interElectronDistances. The computation of the full u matrix is done exactly once, at the very first Metropolis step. This is handled by the SlaterWithJastrow::fillCorrelationMatrix method:

```
void SlaterWithJastrow::fillCorrelationMatrix() {
    mat& u = m_correlationMatrix;
    for (int k = 0; k < m_numberOfElectrons; k++) {
        Electron* kElectron = m_system->getElectrons().at(k);
        for (int i = k+1; i < m_numberOfElectrons; i++) {
            Electron* iElectron = m_system->getElectrons().at(i);
            u(k,i)              = computeJastrowFactor(kElectron,iElectron);
        }
    }
    for (int k = 0; k < m_numberOfElectrons; k++) {
        for (int i = k+1; i < m_numberOfElectrons; i++) {
            u(i,k) = u(k,i);
        }
    }
}
```

The method which evaluates the elements of the matrix u, computeJastrowFactor is implemented in the following

```
inline double SlaterWithJastrow::computeJastrowFactor(Electron*i, Electron*j){
    const double a     = spinCoefficient(i,j);
    const double rik   = m_interElectronDistances(i,j);
    return a * rij / (1.0 + m_beta * rij);
}
```

with the spin coefficient a_{ij} being computed in the spinCoefficient method by a simple return (i->getSpin()== j->getSpin()? 0.25 : 0.5). The m_interElectronDistances, the matrix R, is also fully computed only once.

As a single electron is moved, only one row and column of u and R change. Efficiently updating instead of re-computing is done in the following two SlaterWithJastrow methods:

```
void SlaterWithJastrow::updateElectronDistanceMatrices() {
    Electron* kElectron = m_system->getElectrons().at(m_changedElectron);
    mat&      r         = m_electronPositions;
    mat&      R         = m_interElectronDistances;
    int       k         = m_changedElectron;

    for (int dimension = 0; dimension < 3; dimension++) {
        r(k,dimension) = kElectron->getPosition().at(dimension);
    }
    for (int i = 0; i < k; i++) {
        double x = r(k,0) - r(i,0);
        double y = r(k,1) - r(i,1);
        double z = r(k,2) - r(i,2);
        R(k,i) = sqrt(x*x + y*y + z*z);
        R(i,k) = R(k,i);
    }
    for (int i = k+1; i < m_numberOfElectrons; i++) {
        double x = r(k,0) - r(i,0);
        double y = r(k,1) - r(i,1);
        double z = r(k,2) - r(i,2);
        R(k,i) = sqrt(x*x + y*y + z*z);
        R(i,k) = R(k,i);
    }
    double x = r(k,0);
    double y = r(k,1);
    double z = r(k,2);
    R(k,k) = sqrt(x*x + y*y + z*z);
}
```

and

```
void SlaterWithJastrow::updateCorrelationsMatrix() {
    mat& R = m_interElectronDistances;
    mat& u = m_correlationMatrix;
    mat& a = m_spinMatrix;

    int k = m_changedElectron;
    for (int i = 0; i < k; i++) {
        u(i,k) = a(i,k) * R(i,k) / (1 + m_beta * R(i,k));
        u(k,i) = u(i,k);
    }
    for (int i = k+1; i < m_numberOfElectrons; i++) {
        u(k,i) = a(k,i) * R(k,i) / (1 + m_beta * R(k,i));
        u(i,k) = u(k,i);
    }
}
```

Please note that the k-th diagonal element of the m_interElectronDistance matrix is used to hold the distance of electron k relative to the global origin.

Because of the exponential nature of the Jastrow factor as a whole, extensive can-

celation happens when taking the ratio of $J(\mathbf{R}_{\text{new}})$ to $J(\mathbf{R}_{\text{old}})$, since

$$\begin{aligned}\frac{J(\mathbf{R}_{\text{new}})}{J(\mathbf{R}_{\text{old}})} &= \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} u_{ij}(\mathbf{r}_{ij}^{\text{new}})\right] \Bigg/ \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} u_{ij}(\mathbf{r}_{ij}^{\text{old}})\right] \\ &= \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} u_{ij}(\mathbf{r}_{ij}^{\text{new}}) - \sum_{i=1}^{N}\sum_{j=i+1}^{N} u_{ij}(\mathbf{r}_{ij}^{\text{old}})\right] \\ &= \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} \left\{u_{ij}(\mathbf{r}_{ij}^{\text{new}}) - u_{ij}(\mathbf{r}_{ij}^{\text{old}})\right\}\right], \end{aligned} \tag{9.30}$$

and $u_{ij}(\mathbf{r}_{ij}^{\text{new}}) - u_{ij}(\mathbf{r}_{ij}^{\text{old}})$ obviously vanishes if $\mathbf{r}_i^{\text{new}} = \mathbf{r}_i^{\text{old}}$ and $\mathbf{r}_j^{\text{new}} = \mathbf{r}_j^{\text{old}}$. The only terms which survive the sum are the ones involving m_changedElectron, which we (for the moment) will call k for ease of notation. We find that

$$\begin{aligned}\frac{J(\mathbf{R}_{\text{new}})}{J(\mathbf{R}_{\text{old}})} &= \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} \left\{u_{ij}(\mathbf{r}_{ij}^{\text{new}}) - u_{ij}(\mathbf{r}_{ij}^{\text{old}})\right\}(\delta_{ik} + \delta_{kj})\right] \\ &= \exp\left[\sum_{i=1}^{k-1} \left\{u_{ik}(\mathbf{r}_{ik}^{\text{new}}) - u_{ik}(\mathbf{r}_{ik}^{\text{old}})\right\} + \sum_{j=k+1}^{N} \left\{u_{kj}(\mathbf{r}_{kj}^{\text{new}}) - u_{kj}(\mathbf{r}_{kj}^{\text{old}})\right\}\right]. \end{aligned} \tag{9.31}$$

In the VMC framework this is implemented as

```
void SlaterWithJastrow::computeJastrowRatio() {
    double sum = 0;
    int k = m_changedElectron;
    for (int i = 0; i < k; i++) {
        sum += m_correlationMatrix(i,k) - m_correlationMatrixOld(i,k);
    }
    for (int i = k+1; i < m_numberOfElectrons; i++) {
        sum += m_correlationMatrix(k,i) - m_correlationMatrixOld(k,i);
    }
    m_Rc = exp(sum);
}
```

Efficient calculation of the gradient, $\nabla_i J(\mathbf{R})$

The calculation of the quantum force involves calculating that ratio of the gradient of the wave function. Recall from Eq. (9.24) that

$$\frac{\nabla_i \Psi_{\text{T}}}{\Psi_{\text{T}}} = \frac{\nabla_i |D_\uparrow(\mathbf{R})|}{|D_\uparrow(\mathbf{R})|} + \frac{\nabla_i |D_\downarrow(\mathbf{R})|}{|D_\downarrow(\mathbf{R})|} + \frac{\nabla_i J(\mathbf{R})}{J(\mathbf{R})}. \tag{9.32}$$

We require therefore an efficient algorithm for calculating $\nabla_k J(\mathbf{R})/J(\mathbf{R})$. By the chain rule on the exponential form of $J(\mathbf{R})$, we have trivially that

$$\begin{aligned}\nabla_k J(\mathbf{R}) &= \nabla_k \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} u_{ij}(\mathbf{r}_{ij})\right] = J(\mathbf{R})\nabla_k\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} u_{ij}(\mathbf{r}_{ij})\right] \\ &= J(\mathbf{R})\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N} \nabla_k u_{ij}(\mathbf{r}_{ij})\right]. \end{aligned} \tag{9.33}$$

Furthermore, only the terms containing coordinate indexed k survive the differentiation without vanishing, i.e. u_{kj} or u_{ik}. The first Cartesian coordinate of the gradient w.r.t. the coordinates of electron k correspond to the differentiation $\partial/\partial x_k$,

$$\frac{1}{J(\mathbf{R})}\frac{\partial J(\mathbf{R})}{\partial x_k} = \sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{\partial}{\partial x_k}u_{ij}(\mathbf{r}_{ij}) = \sum_{i=1}^{k-1}\frac{\partial}{\partial x_k}u_{ik}(\mathbf{r}_{ik}) + \sum_{j=k+1}^{N}\frac{\partial}{\partial x_k}u_{kj}(\mathbf{r}_{kj}), \tag{9.34}$$

but with $u_{ij}(\mathbf{r}_{ij})$ depending exclusively on the relative coordinates $|\mathbf{r}_i - \mathbf{r}_j|$ it is more natural to differentiate w.r.t. r_{ij}. By the chain rule we have

$$\frac{\partial}{\partial x_k} = \frac{\partial r_{ik}}{\partial x_k}\frac{\partial}{\partial r_{ik}} = \frac{x_k - x_i}{r_{ik}}\frac{\partial}{\partial r_{ik}} = -\frac{x_i - x_k}{r_{ki}}\frac{\partial}{\partial r_{ki}}, \tag{9.35}$$

meaning Eq. (9.34) simplifies to [68]

$$\frac{1}{J(\mathbf{R})}\frac{\partial J(\mathbf{R})}{\partial x_k} = \sum_{i=1}^{k-1}\frac{x_k - x_i}{r_{ik}}\frac{\partial}{\partial r_{ik}}u_{ik}(\mathbf{r}_{ik}) - \sum_{j=k+1}^{N}\frac{x_j - x_k}{r_{kj}}\frac{\partial}{\partial r_{kj}}u_{kj}(\mathbf{r}_{kj}). \tag{9.36}$$

Identical arguments for the Cartesian y_k and z_k coordinates lead finally to the gradient

$$\frac{\nabla_k J(\mathbf{R})}{J(\mathbf{R})} = \sum_{i=1}^{k-1}\frac{\mathbf{r}_{ik}}{r_{ik}}\frac{\partial u_{ik}}{\partial r_{ik}} - \sum_{j=k+1}^{N}\frac{\mathbf{r}_{kj}}{r_{kj}}\frac{\partial u_{kj}}{\partial r_{ki}}. \tag{9.37}$$

For the simple two-body Jastrow factor used in the present work, the partial derivatives of u_{ij} take the simple form [17]

$$\frac{\partial u_{ij}}{\partial r_{ij}} = \frac{\partial}{\partial r_{ij}}\left(\frac{a_{ij}r_{ij}}{1+\beta r_{ij}}\right) = \frac{a_{ij}}{(1+\beta r_{ij})^2}. \tag{9.38}$$

Updating the terms of the Jastrow gradient is done in the method updateJastrowGradient of the SlaterWithJastrow class:

```
void SlaterWithJastrow::updateJastrowGradient(int k) {
    mat& a = m_spinMatrix;
    mat& R = m_interElectronDistances;

    for (int i = 0; i < k; i++) {
        const double    rik     = R(k,i);
        double          factor  = 1 + m_beta * rik;
        m_jastrowGradient(i, k) = a(k,i) / (factor * factor);
    }
    for (int j = k+1; j < m_numberOfElectrons; j++) {
        const double    rkj     = R(k,j);
        double          factor  = 1 + m_beta * rkj;
        m_jastrowGradient(k, j) = a(j,k) / (factor * factor);
    }
}
```

At the very first Metropolis step a loop over the electrons where updateJastrowGradient is performed—on each one in turn—is performed. This calculates $\nabla_k J(\mathbf{R})$ from scratch for all electrons, and is of course done only once.

Efficient calculation of the Laplacian, $\nabla_i^2 J(\mathbf{R})$

As the gradient was needed for the quantum force calculation, the Laplacian is necessary for computing the kinetic energy. Recall from Eq. (9.27) that

$$\frac{\nabla_i^2 \Psi_\mathrm{T}}{\Psi_\mathrm{T}} = \frac{\nabla_i^2 |D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i^2 |D_\downarrow|}{|D_\downarrow|} + \frac{\nabla_i^2 J}{J} + 2\left[\frac{\nabla_i |D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i |D_\downarrow|}{|D_\downarrow|}\right] \cdot \frac{\nabla_i J}{J}. \tag{9.39}$$

In the following we derive an efficient scheme for calculating the $\nabla_i^2 J(\mathbf{R})/J(\mathbf{R})$ term.

Let us consider the differentiation w.r.t. the first Cartesian coordinate of electron k, i.e. x_k. By Eq. (9.33) we have

$$\begin{aligned}
\frac{\partial^2 J}{\partial x_k^2} &= \frac{\partial}{\partial x_k}\left\{\frac{\partial J(\mathbf{R})}{\partial x_k}\right\} = \frac{\partial}{\partial x_k}\left\{J(\mathbf{R})\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{\partial u_{ij}}{\partial x_k}\right\} \\
&= \frac{\partial J(\mathbf{R})}{\partial x_k}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{\partial u_{ij}}{\partial x_k} + J(\mathbf{R})\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{\partial^2 u_{ij}}{\partial x_k^2} \\
&= \left\{J(\mathbf{R})\sum_{m=1}^{N}\sum_{n=m+1}^{N}\frac{\partial u_{mn}}{\partial x_k}\right\}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{\partial u_{ij}}{\partial x_k} + J(\mathbf{R})\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{\partial^2 u_{ij}}{\partial x_k^2}.
\end{aligned} \tag{9.40}$$

Taking the ratio of $\partial^2 J(\mathbf{R})/\partial x_k^2$ with $J(\mathbf{R})$ will cancel the $J(\mathbf{R})$ factors. The double

derivatives in the last term evaluate to

$$\begin{aligned}\frac{\partial^2 u_{ik}}{\partial x_k^2} &= \frac{\partial}{\partial x_k}\left[\frac{(x_k - x_i)}{r_{ik}}\frac{\partial u_{ik}}{\partial r_{ik}}\right] \\ &= \frac{(x_k - x_i)}{r_{ik}}\frac{\partial}{\partial r_{ik}}\left[\frac{(x_k - x_i)}{r_{ik}}\frac{\partial u_{ik}}{\partial r_{ik}}\right] \\ &= \frac{(x_k - x_i)}{r_{ik}}\left[\frac{\frac{r_{ik}^2}{(x_k - x_i)} - (x_k - x_i)}{r_{ik}^2}\frac{\partial u_{ik}}{\partial r_{ik}} + \frac{(x_k - x_i)}{r_{ik}}\frac{\partial^2 u_{ik}}{\partial r_{ik}^2}\right],\end{aligned} \tag{9.41}$$

after two applications of the chain rule to move the differentiation onto r_{ik} and using that

$$\frac{\partial}{\partial r_{ik}}\left[\frac{(x_k - x_i)}{r_{ik}}\right] = \frac{\frac{\partial}{\partial r_{ik}}(x_k - x_i)}{y} \tag{9.42}$$

with

$$\frac{\partial (x_k - x_i)}{\partial r_{ik}} = \frac{1}{\left[\frac{\partial r_{ik}}{\partial (x_k - x_i)}\right]} = \frac{1}{\left[\frac{(x_k - x_i)}{r_{ik}}\right]} = \frac{r_{ik}}{(x_k - x_i)}. \tag{9.43}$$

Taking the sum over all three Cartesian coordinates, Eq. (9.41) simplifies dramatically to

$$\nabla_k^2 u_{ik} = \left[\frac{d-1}{r_{ik}}\right]\frac{\partial u_{ik}}{\partial r_{ik}} + \frac{\partial^2 u_{ik}}{\partial r_{ik}^2}, \tag{9.44}$$

where d denotes the number of spatial dimensions used. In our case, $d = 3$ always. Note carefully that the same equation holds when differentiating w.r.t. the second index, i.e. $\partial^2 u_{kj}/\partial x_k^2$ does *not* carry a minus sign as was the case for the gradient.

As before, only the $N - 1$ terms in the sum containing a k index survive the differentiation, giving in total

$$\sum_{i=1}^{N}\sum_{j=i+1}^{N}\nabla_k^2 u_{ij} = \sum_{i=1}^{k-1}\left\{\frac{2}{r_{ik}}\frac{\partial u_{ik}}{\partial r_{ik}} + \frac{\partial^2 u_{ik}}{\partial r_{ik}^2}\right\} + \sum_{j=k+1}^{N}\left\{\frac{2}{r_{kj}}\frac{\partial u_{kj}}{\partial r_{kj}} + \frac{\partial^2 u_{kj}}{\partial r_{kj}^2}\right\}. \tag{9.45}$$

The first term of Eq. (9.40) equals—after taking the sum over the three Cartesian coordinates—the inner product of the gradient with itself [68]. Since the evaluation of this term neccessitates the evaluation of the gradient, we perform this computation in conjunction with computing the quantum force. This is also where we handle the *cross term* in the total Laplacian, c.f. Eq. (9.27). The rest of the Jastrow laplacian is calculated in

```
void SlaterWithJastrow::updateJastrowLaplacianTerms(int k) {
    mat& a           = m_spinMatrix;
    mat& R           = m_interElectronDistances;
    mat& laplacianJ  = m_jastrowLaplacianTerms;

    for (int j = 0; j < k; j++) {
        double  factor  = 1 + m_beta * R(k,j);
        laplacianJ(j, k) = -2*a(j,k) * m_beta / (factor*factor*factor);
    }
    for (int j = k+1; j < m_numberOfElectrons; j++) {
        double  factor  = 1 + m_beta * R(k,j);
        laplacianJ(k, j) = -2*a(k,j) * m_beta / (factor*factor*factor);
    }
}
```

and

```
void SlaterWithJastrow::computeJastrowLaplacian() {
    mat& R           = m_interElectronDistances;
    mat& laplacianJ  = m_jastrowLaplacianTerms;
    mat& gradientJ   = m_jastrowGradient;

    double sum = 0;
    for (int k = 0; k < m_numberOfElectrons; k++) {
        for (int i = 0; i < k; i++) {
            sum += 2/R(i,k) * gradientJ(i,k) + laplacianJ(i,k);
        }
        for (int i = k+1; i < m_numberOfElectrons; i++) {
            sum += 2/R(i,k) * gradientJ(k,i) + laplacianJ(k,i);
        }
    }
    m_jastrowLaplacian = sum;
}
```

Here, the double derivative of u_{ij} w.r.t. r_{ij} is computed as

$$\frac{\partial^2}{\partial r_{ij}^2}\left[\frac{a_{ij}r_{ij}}{1+\beta r_{ij}}\right] = -\frac{2a_{ij}\beta}{(1+\beta r_{ij})^3}. \tag{9.46}$$

Calculating the quantum force

Combining methods for calculating the gradients of the Slater determinants and the Jatrow factor, computing the quantum force

$$\mathbf{F} = 2\frac{\nabla\Psi(\mathbf{R})}{\Psi(\mathbf{R})}, \tag{9.47}$$

is relatively straight forward.

```
void SlaterWithJastrow::computeQuantumForce() {
    mat& R = m_interElectronDistances;
    mat& r = m_electronPositions;

    m_energyCrossTerm = 0;
```

```
    for (int k = 0; k < m_numberOfElectrons; k++) {
        const int kSpin      = m_system->getElectrons().at(k)->getSpin();
        const int kSpinIndex = m_system->getElectrons().at(k)->getSpinIndex();

        mat& slaterGradient = (kSpin==1 ? m_slaterGradientUp :
                                          m_slaterGradientDown);

        for (int j = 0; j < 3; j++) {
            double sum = 0;
            for (int i = 0; i < k ; i++) {
                const double xk = r(k,j);
                const double xi = r(i,j);
                sum += (xk - xi) / R(i,k) * m_jastrowGradient(i, k);
            }
            for (int i = k + 1; i < m_numberOfElectrons; i++) {
                const double xk = r(k,j);
                const double xi = r(i,j);
                sum -= (xi - xk) / R(i,k) * m_jastrowGradient(k,i);
            }
            m_quantumForce(k , j) = 2 * slaterGradient(kSpinIndex,j);

            if (m_jastrow) {
                m_quantumForce(k ,j) += 2    * sum;
                m_energyCrossTerm    -= 0.5 * sum*sum +
                                        (slaterGradient(kSpinIndex,j) * sum);
            }
        }
    }
}
```

Note carefully that the m_energyCrossTerm described earlier is computed here, as we have direct access to the fully computed Jastrow gradient, as well as the gradient of the Slater determinant. Also computed is the $\nabla_k J(\mathbf{R}) \cdot \nabla_k J(\mathbf{R})$ factor which was missing in the computeJastrowLaplacian method earlier.

9.2.2 The Orbital class

The orbitals populating the Slater determinants can in principle be any linearly independent square integrable $\mathbb{R}^3 \mapsto \mathbb{C}$ functions. The only neccessary conditions for adding new orbital types in the VMC program is that three methods are present: evaluate, computing the value; computeDerivative, which evaluates the derivative w.r.t. one of the Cartesian coordinates; and computeLaplacian which predictably evaluates the Laplacian at given coordinates.

A global ordering of the orbitals is assumed, and all three neccessary methods provide an index telling which specific orbital is to be evaluated / differentiated. As an example, if index 2 is provided to the HydrogenOrbital::evaluate function, the following code is executed

```
double HydrogenOrbital::evaluate2s(double r) {
   return m_2sNormalization * (1 - m_alpha * 0.5*r) * exp(-m_alpha * 0.5 * r);
}
```

Both the HydrogenOrbital and the SlaterTypeOrbital sub-classes use the standard numbering, 1s, 2s, $2p_x$, $2p_y$, and so on. The global ordering of the GaussianOrbitals is—in a sense—more arbitrary. As the raw Hartree-Fock orbitals are used, we are in no way guaranteed that they are sorted in the sense that the orbital Hartree-Fock energies satisfy $\varepsilon_1 < \varepsilon_2 < \varepsilon_3 < \ldots$

The Gaussian type orbitals employ two classes, PrimitiveGaussian and ContractedGaussian. The orbital class holds the Hartree-Fock basis expansion coefficients, and organizes the evaluation of the former two. Calling the GaussianOrbital::evaluate method produces the following cascade

```
double GaussianOrbital::evaluate(double x,     double y,    double z,
                                 int    index, int    spin) {
    double value = 0;
    for (int i=0; i < m_basisSize; i++) {
        double c = (spin==1 ? m_spinUpCoefficients(i,index) :
                              m_spinDownCoefficients(i,index));
        value += (spin==1 ? m_spinUpCoefficients(i,index) :
                            m_spinDownCoefficients(i,index)) *
                            m_basis.at(i)->evaluate(x,y,z);
    }
    return value;
}
```

which calls

```
double ContractedGaussian::evaluate(double x, double y, double z) {
    double result = 0;
    for (PrimitiveGaussian* primitive : m_primitives) {
        result += (*primitive)(x - m_x, y - m_y, z - m_z);
    }
    m_currentValue = result;
    return result;
}
```

which finally computes the Gaussian function value in

```
double PrimitiveGaussian::operator()(double x, double y, double z) {
    const double r2    = x*x + y*y + z*z;
    const double value = m_constant  *
                         pow(x, m_i) *
                         pow(y, m_j) *
                         pow(z, m_k) *
                         exp(- m_alpha * r2);
    m_currentValue     = value;
    return value;
}
```

Note that the current value of each primitive is saved, to avoid having to re-compute the function value when derivatives are computed shortly thereafter.

9.2.3 The Metropolis class

The Metropolis class sets up and runs the Markov chain in the N-electron configuration space. The user calls the Metropolis::runSteps, which handles the "time" steps. Some automatic estimates are used if no user input is specified, e.g. the minimum *size* atom present determines the step length if no user specified step length is provided.

The proposition of new configurations, aswell as the accept-reject Metropolis test is performed in the Metropolis::step method. In order to draw samples from pseudo-random distributions, the standard C++11 machinery is used: A random *device* and a *generator* is defined, and then samples are drawn according to some *distribution*. An excerpt of the step method is shown in the following:

```
bool Metropolis::step() {
    std::normal_distribution        <double> normalDistribution{0,1};
    std::uniform_real_distribution<double> uniformDouble      {0,1};
    std::uniform_int_distribution  <int>    uniformIntElectron
                                                    {0,
                                                     m_numberOfElectrons-1};
    int    electron = uniformIntElectron (m_randomGenerator);
    double D        = 0.5;
    double xProposedChange = normalDistribution(m_randomGenerator) *
               m_dtSqrt + m_waveFunction->getQuantumForceOld(electron,0) *
               m_dt * D;
    double yProposedChange = normalDistribution(m_randomGenerator) *
               m_dtSqrt + m_waveFunction->getQuantumForceOld(electron,1) *
               m_dt * D;
    double zProposedChange = normalDistribution(m_randomGenerator) *
               m_dtSqrt + m_waveFunction->getQuantumForceOld(electron,2) *
               m_dt * D;

    m_waveFunction->passProposedChangeToWaveFunction(electron, dimension);

    m_system->getElectrons().at(electron)->
                        adjustPosition(xProposedChangeImportanceSampling, 0);
    m_system->getElectrons().at(electron)->
                        adjustPosition(yProposedChangeImportanceSampling, 1);
    m_system->getElectrons().at(electron)->
                        adjustPosition(zProposedChangeImportanceSampling, 2);

    m_waveFunction->updateOldWaveFunctionValue();

    double R = m_waveFunction->computeWaveFunctionRatio(electron) *
               computeGreensFunction();

    if (R > uniformDouble(m_randomGenerator)) {
        m_waveFunction->updateWaveFunctionAfterAcceptedStep();
        return true;
    } else {
        m_waveFunction->updateWaveFunctionAfterRejectedStep();

        m_system->getElectrons().at(electron)->
                        adjustPosition(-xProposedChangeImportanceSampling, 0);
        m_system->getElectrons().at(electron)->
                        adjustPosition(-yProposedChangeImportanceSampling, 1);
        m_system->getElectrons().at(electron)->
                        adjustPosition(-zProposedChangeImportanceSampling, 2);
        return false;
    }
}
```

Note that the possibility of *not* using importance sampling is removed in this excerpt. In the actual Metropolis::step, this possibility is of course preserved.

As a final note on the specific implementation of the VMC framework, we present the evaluation of the Green's function. This is handled by the Metropolis class, which asks the wave function currently in use—m_waveFunction—for quantum force values. Recall from Eq. (6.31) that the Green's function of the short-time limit Fokker-Planck equation is given by

$$G(Y, X; \delta t) = \left(\frac{1}{4\pi D \delta t}\right)^{-3N/2} \exp\left(\frac{-\left[Y - X - D\delta t \mathbf{F}(X)\right]^2}{4D\delta t}\right). \tag{9.48}$$

This is computed as follows

```
double Metropolis::computeGreensFunction() {
    const double D = 0.5;
    WaveFunction* wf = m_waveFunction;
    double greensFunction = 0;
    for (int i = 0; i < m_numberOfElectrons; i++) {
        for (int j = 0; j < 3; j++) {
            greensFunction += 0.5 *
                  (wf->getQuantumForceOld(i,j) + wf->getQuantumForce(i,j)) *
                  (D * m_dt * 0.5 *
                     (wf->getQuantumForceOld(i,j) - wf->getQuantumForce(i,j)) -
                      wf->getPosition(i,j)        + wf->getPositionOld(i,j));
        }
    }
    return exp(greensFunction);
}
```

9.2.4 Variational parameter optimization: β

Chapter 10

Implementation: Artificial Neural Networks

The following is a description of the implementation of the artificial neural network (ANN) framework described in chapter 7. The main body of the implementation consists of around $1\,500$ lines of object oriented Python code. The structure of the neural networks (NN) and the training procedure is implemented by us, but the underlying back-propagation (by automatic differentiation[1]) and parameter optimization is handled by the TensorFlow library [150]. Our code consists of around 10 classes, but a generic user need only interact with a single Python source file. The program is designed to be run from the command line where numerous command line arguments dictate which computation is run and how the output is handled/visualized.

The NN framework is essentially used as a general curve fitting procedure, capable of "parameter free" fitting of (in principle) any real mapping $f : \mathbb{R}^M \mapsto \mathbb{R}^N$. Apart from possibly some examples of pathologically badly-behaved functions, the NN machinery can find a least squares[2] fit to any f. Whereas ordinary curve fitting algorithms require a parametrized ansatz, the ANN approach is completely general.

Given a functional form (with or without added random noise), the developed code is capable of finding an approximation to the noise-less underlying function. It is also possible to provide the program with a file consisting of data points and have the code compute a parametrization of the data points based on one or more inputs.

[1] *Automatic differentiation* denotes the process of analytically evaluating the derivative of an arbitrary computer program w.r.t. any variable in that program. It exploits the fact that any code—regardless of how complicated—at the end of the day only applies a series of elementary operations to a set of variables. Since the derivative of such elementary operations (addition, subtraction, multiplication, sines, exponentiation, etc.) are all known analytically, repeated application of the chain rule can in principle give the closed form analytical derivative of *any* computer code. Please note *very carefully* that this differs fundamentally from an ordinary numerical (finite difference) derivative approximation.

[2] The precise meaning of *least squares* in this context is made clear in chapter 7.

For example, a set of energies originating from *ab initio* QM calculations,

$$E^i_{ab\,initio} = E_{ab\,initio}(r^i_{12}, r^i_{13}, r^i_{23}, \dots, \theta^i_{123}, \theta^i_{134}, \theta^i_{124}, \dots). \tag{10.1}$$

The superscript i signifies the discrete sampling—the energy is only calculated quantum mechanically at a finite set of N configurations—with given inter-nucleus separations r_{ij} and nucleus-nucleus-nucleus angles θ_{ijk}. Feeding the ANN with the discrete nucleonic configurations (distances and angles) and the corresponding *ab initio* energies, the network can *learn* the underlying patterns and provide an continous interpolation

$$E_{\text{NN}} = E_{\text{NN}}(r_{12}, r_{13}, r_{23}, \dots, \theta_{123}, \theta_{134}, \theta_{124}, \dots). \tag{10.2}$$

We will start off our description by presenting examples of the usage of the code, before we delve deeper into the specific implementation.

10.1 Introductory examples

The ANN code is controlled primarily from the command line, and interaction with the source directly is only neccessary for *advanced use*. Querying the program with a `--help` option gives an overview of the usage, i.e.

```
(tensorflow)$ python tfpotential.py --help
```

The `(tensorflow)` denotes an active (possibly virtual) environment which has TensorFlow (TF) and all required libraries in the appropriate Python paths. As a rule, it is generally beneficient to install TensorFlow in a virtual environment (to avoid interfering with the system default Python binaries) using e.g. Anaconda package system [151].

By default—if not otherwise specified—a Lennard-Jones (LJ) functional form is used as an example. The code admits a single positional argument, namely the number of training *epochs* to go through. For example, the following command line statement will run training over 200 epochs on a default LJ data set, and (once finished) visualize the NN output, the training progress, and the approximation error:

```
(tensorflow)$ python tfpotential.py 200 --plotall
```

The structure of the network (number of layers and the amount of neurons per hidden layer) can be specified with the `--size` option. Additionally, training with a data set from e.g. *ab initio* QM calculations can be done by specifying the name of a file containing said data. For long training processes, it is convenient to be able to save the NN state. This enables pausing and resuming the training, and is handled in the code by the `--save` and `--load` key-words. The following example runs 1000 training epochs on a data set from the file `QMData.dat`, saving the network structure and state to facilitate subsequent reloading for more training:

```
(tensorflow)$ python tfpotential.py 1000 --size 3 10 --file QMData.dat --save
```

A network size of 3 hidden layers—each consisting of 10 neurons—is used, and an example of the *on the fly* output of the program is shown in Fig. 10.1.

10.2 Overview of select classes

10.2.1 The NeuralNetwork class

The actual structure and evaluation of the NN is done in the NeuralNetwork class. Here, the weights and biases are initialized and organized into hidden layers. Weights are initialized using the *Xavier* method [152]

```
def initializeWeight(self, shape, layer) :
    nIn    = shape[0]
    nOut   = shape[1]
    if self.hiddenActivation = tf.nn.sigmoid :
        limit = 4 * np.sqrt(6.0 / (nIn + nOut))
    elif self.hiddenActivation == tf.nn.tanh :
        limit =     np.sqrt(6.0 / (nIn + nOut))

    lowerLimit = -limit
    upperLimit =  limit

    name = 'w%d' % (layer)
    with tf.name_scope("Weights") :
        weight = tf.Variable(tf.random_uniform( shape,
                                                lowerLimit,
                                                upperLimit), name=name);
        self.summary(name, weight)
    return weight
```

The TensorFlow variables are classified according to their uses, and can later be visualized using the TensorBoard web visualization tool [153]. An analogous method initializes the biases. Both of the initialization functions are used to create *layers* by

```
def layer(self,
          y_,
          layerNumber,
          activation=None,
          inputLayer=False,
          outputLayer=False) :

    iSize = self.nNodes if (not inputLayer)  else self.inputs
    jSize = self.nNodes if (not outputLayer) else self.outputs
    self.w.append(self.initializeWeight([iSize, jSize], layerNumber))
    self.b.append(self.initializeBias  ([jSize],        layerNumber))
    y_ = tf.add(tf.matmul(y_, self.w[-1]), self.b[-1])
    return y_ if (activation == None) else activation(y_)
```

```
Initializing network:
 => layers     3
 => neurons    10
 => type       sigmoid

Training network:
 => epochs          1000
 => function        QMData.dat
 => data set size   10000
 => batch size      200
 => test set size   1000

================================================================================
                    Cost/                          Test Cost/
Epoch      Cost     DataSize        Test Cost      TestSize
--------------------------------------------------------------------------------
  0      294984.9    2.949849       6296.4092      6.2964092
  1      5946.0332   0.059460332    5746.3628      5.7463628
  2      5909.0933   0.059090933    5743.9268      5.7439268
  3      5906.6206   0.059066206    5741.6689      5.7416689
  4      5903.2309   0.059032309    5736.8623      5.7368623
  5      5899.0264   0.058990264    5733.3242      5.7333242   saved: ckpt-0
  6      5893.5371   0.058935371    5725.332       5.725332
  7      5885.2447   0.058852447    5716.1626      5.7161626
  8      5875.6524   0.058756524    5704.4277      5.7044277
  9      5861.1823   0.058611823    5689.1689      5.6891689
  10     5843.8268   0.058438268    5668.082       5.668082    saved: ckpt-1
  11     5822.3592   0.058223592    5647.6074      5.6476074
  12     5791.2043   0.057912043    5612.1699      5.6121699
  13     5751.5597   0.057515597    5563.3501      5.5633501

                                 ...

  995    13.405828   0.00013405828  17.932564      0.017932564
  996    13.568553   0.00013568553  17.441454      0.017441454
  997    13.552082   0.00013552082  17.314411      0.017314411
  998    13.498887   0.00013498887  17.568069      0.017568069
  999    13.575483   0.00013575483  17.655457      0.017655457
================================================================================
```

Figure 10.1: Output produced by the example run of the NN program shown in section 10.1. Saving of the network state is done *at most* every 5 epochs, but only if the current cost function computed for the test set attains a minimum. If other states with lower values of this cost function have already been saved as a previous checkpoint, the current one is not saved. The output has been lightly edited to make it fit (a column showing the elapsed time per epoch is removed, and some non-UTF8 characters have been replaced with similar UTF8 characters, among other things).

In each layer, the signature "matrix multiply and bias add" is performed, with weight matrices and bias vectors being initialized according to the size of the incoming/outgoing signal.

Layers can then be combined to form the full network model, which we subsequently will evaluate. This is done in the fullNetwork method

```
def fullNetwork(self,
                y_,
                inputs,
                nLayers,
                nNodes,
                outputs,
                hiddenActivation,
                lastActivation) :

    self.w, self.b = [], []
    self.inputs = inputs
    self.nNodes = nNodes
    self.hiddenActivation = hiddenActivation
    self.lastActivation   = lastActivation

    y_ = self.layer(y_, 0, activation=self.hiddenActivation, inputLayer=True)
    for i in xrange(1, nLayers) :
        y_ = self.layer(y_, i,      activation=self.hiddenActivation)

    y_ = self.layer(y_, nLayers,   activation=self.lastActivation)
    y_ = self.layer(y_, nLayers+1, activation=None, outputLayer=True)
    return y_
```

In order to construct a persistent network, and avoid having to pass around several function arguments, a constructNetwork method is used, which creates and saves the network configuration.

```
def constructNetwork( self,
                      inputs,
                      nNodes,
                      nLayers,
                      outputs,
                      networkType=None) :
    self.networkType  = networkType
    self.nLayers      = nLayers
    self.nNodes       = nNodes
    self.inputs       = inputs
    self.outputs      = outputs
    self.x            = tf.placeholder('float', [inputs,  None], name='x')
    self.y            = tf.placeholder('float', [outputs, None], name='y')
    self.parseTypeString(networkType)
    self.network = lambda x : \
                   self.fullNetwork(x,
                                    inputs           = self.inputs,
                                    nLayers          = self.nLayers,
                                    nNodes           = self.nNodes,
                                    outputs          = self.outputs,
                                    hiddenActivation = self.hiddenActivation,
                                    lastActivation   = self.lastActivation)

def __call__(self, inputData) :
    return self.network(inputData)
```

Note that the call-method enables usage of a NeuralNetwork instance as a *callable object*, essentially using the class instance directly as a function.

The activation functions can be any one of the pre-defined TensorFlow activations, including rectified linear (ReLU), exponential linear (ELU) sigmoid, or hyperbolic tangent, among others. In order to avoid unnecessarily constraining the final output, no activation is applied for the last layer.

10.2.2 The NetworkTrainer class

The training of the Network is handled entirely by the NetworkTrainer class. Here, the TensorFlow session is initialized, and a cost function is minimized according to some specified optimization algorithm. Changing the cost function or the optimizer is not currently supported via command line arguments, but can be done by interchanging a single line of code in the source. For all runs in the present work we use the Adam (adaptive moment estimation) optimizer and a ℓ^2 norm difference,

$$\text{Cost}(\hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2, \tag{10.3}$$

is used as the cost function [137]. Recall that for an input vector $\mathbf{x}$, the NN output is denoted $\hat{\mathbf{y}}$ while the true result is $\mathbf{y}$. The ℓ^2 norm cost function is thus simply the square root squared difference between the true result and the network output. The ℓ^2 space—briefly mentioned in section 2.2.1—is the space of *square summable sequences*, essentially a special case of the $\mathcal{L}^2$ space (see e.g. Rynne and Youngson [42]).

The NetworkTrainer constructor defines two *placeholders*, which are later assigned to TF variables. In addition, the network is assembled and cost function and optimizer is set up.

```
def __init__(self, system, saver) :
    self.system    = system
    self.x         = tf.placeholder( 'float',
                                     [None, system.inputs],
                                     name='x')
    self.y         = tf.placeholder( 'float',
                                     [None, system.outputs],
                                     name='y')
    self.prediction = system.network(self.x)
    self.cost       = tf.nn.l2_loss(tf.subtract(self.prediction, self.y))
    self.adam       = tf.train.AdamOptimizer()
    self.optimizer  = self.adam.minimize(self.cost)
    self.save       = system.argumentParser().save
    self.saver      = saver
```

The system parameter is an instance of the TFPotential class, which acts as a driver for the program, glueing the different pieces together. This is the only class the user needs interact with for basic usage of the NN machinery.

Initialization of the TensorFlow session and the optimization takes place in the trainNetwork method. The statements

```
self.sess = tf.Session()
self.sess.run(tf.global_variables_initializer())
```

initializes variables to TF structures, essentially constructing the computational graph. The input variables—usually NumPy (see e.g. Langtangen [154]) arrays—are converted to TensorFlow *tensors*. These are only explicitly evaluated and differentiated (up to the required order, depending on the graph) when the TF run method is called on them. An example of this is the evaluation of the optimizer and cost function in conjunction with the training step. This is done by

```
self.sess = tf.Session()
self.sess.run(tf.global_variables_initializer())
bOpt, bCost = self.sess.run([self.optimizer, self.cost],
                                    feed_dict={ self.x: xBatch,
                                                self.y: yBatch})
```

The feed_dict feeds NumPy arrays into the placeholder variables self.x and self.y. The cost function is then evaluated in the TF graph, alongside self.optimizer. Recall that the latter variable was defined as

```
self.optimizer  = self.adam.minimize(self.cost)
```

and *evaluating* it constitutes an update of the NN weights according to the gradients w.r.t. the cost function (see chapter 7 for a brief description of the back-propagation algorithm).

In total the trainNetwork function takes the following (abridged) form:

```
def trainNetwork(self, numberOfEpochs):
    self.sess = tf.Session()
    self.sess.run(tf.global_variables_initializer())
    # ...
    xEpoch, yEpoch, xTest, yTest = self.system.dataGenerator.generateData \
                                                    (self.system.dataSize,
                                                     self.system.testSize)
    # ...
    for epoch in xrange(numberOfEpochs):
        indices = np.random.choice(dataSize, dataSize, replace=False)
        xEpoch = xEpoch[indices]
        yEpoch = yEpoch[indices]

        self.epochCost = 0
        for i in xrange(dataSize / batchSize):
            startIndex  = i*batchSize
            endIndex    = startIndex + batchSize
            xBatch      = xEpoch[startIndex:endIndex]
            yBatch      = yEpoch[startIndex:endIndex]
            bOpt, bCost = self.sess.run([self.optimizer, self.cost],
                                        feed_dict={ self.x: xBatch,
                                                    self.y: yBatch})
            self.epochCost += bCost

        tCost = -1
```

```
        if epoch % self.system.testInterval == 0 :
            tOpt, tCost = self.sess.run([self.testCost, self.cost],
                                        feed_dict={ self.x: xTest,
                                                    self.y: yTest})
    #...
```

The DataGenerator class is used to provide the training and validation data sets, either from a generating functional form (e.g. a Lennard-Jones form) or from a file containing e.g. *ab initio* QM data.

Online learning and order randomization

The training scheme used in the current work is an *online leaning* method. Online learning describes a way of feeding inputs through the neural network and is usually contrasted with the *batch learning*. In the batch learning scheme, the entire data set is pushed through the NN in one fel swoop at every epoch. The weights and biases are then updated according to the gradient w.r.t. the cost function *of the entire data set*. In order to facilitate data sets which are too big for simultaneous evaluation in the network, online methods feed only a part of the data set through before updating the parameters. If each online batch is size M, and the total data set is N samples, then each epoch consists of evaluating M/N mini-batches with subsequent parameter update.

Crucially, the training data is reordered in a random manner before each training epoch begins. If this is not done—and especially if the training samples are heavily correlated such as e.g. training point i is given by $f(x_i)$ for $x_i = x_0 + \Delta x$—then it is concievable that the model becomes *stuck in place*. As each mini-batch only trains the NN in very constrained range of inputs with heavily homogenous function values, the model has no way of optimizing in the *global* sense, i.e. for the entire data set.

10.2.3 The TFPotential and the DataGenerator classes

Lastly, let us briefly discuss the TFPotential and DataGenerator classes. The former class acts as the interface between the user and the ANN machinery, being the class which is used to start calculations. Calling the tfpotential.py file evaluates

```
if __name__ == "__main__" :
    tfpot = TFPotential()
    tfpot.train(tfpot.argumentParser().epochs)
```

which creates a TFPotential class instance and then calls the train method

```
    def train(self, epochs=-1) :
        numberOfEpochs = self.numberOfEpochs if epochs == -1 else epochs
        self.numberOfEpochs = numberOfEpochs
        self.networkTrainer.trainNetwork(numberOfEpochs)
        self.sess = self.networkTrainer.sess
```

which sets up and facilitates the training. In order to help with setting the correct parameters asked for by the user, the ArgumentParser class is used to handle command line `--argument` and `value` pairs.

The data generation (or *data organization* in the case of QM training data) is handled by the DataGenerator class, who's primary job is to ensure the training data is structured correctly to be fed through the network. Since The default behaviour—with the default LJ functional form—is to create a NumPy `linspace` between two cutoffs, using a number of points specified by the user. Note that since we shuffle the training data around prior to every single training epoch, there is no inherent problem with such a heavily spatially correlated data set.

Chapter 11

Implementation and validation: Density Functional Theory

In order to retain *some* semblance of brevity in the present work, further description of the DFT implementation is omitted. In following, we will rigorously test and use only two different *ab initio*, namely the HF and VMC frameworks. The developed DFT code—much of which derives from the HF program described in the previous chapter—is in a functional state and available on github.com/mortele/HartreeFock under a branch called DFT.

The DFT program is essentially ready for inclusion in the multiscale modelling framework we are developing, but lacks the proper testing done on the remaining two QM methods in this thesis. Thus a natural extension of the present work entails rigorous testing and comparison of the DFT code with the other two algorithms. We expand on this in the closing remarks of chapter IV.

The theory section on DFT is intentionally left in the thesis—as a chapter in part II—and we hope it can provide a helpful introduction to implementation specific details. We find it especially likey that the accessible introduction to numerical grid based integration techniques used extensively in practical DFT calculations will prove useful to any reader wishing to implement their own density functional scheme from scratch.

Chapter 12

Hartree-Fock validation tests

12.1 Dissociation of the hydrogen molecule ion, H_2^+

The conceptually simplest possible diatomic molecule is the hydrogen molecule ion, H_2^+, sometimes also called the dihydrogen cation. Being a positively ionized hydrogen molecule, it consists of a single electron in the Coulomb field of two hydrogen atoms. Dissociation of this molecule involves breaking up the covalent one-electron bond, $H_2^+ \rightleftharpoons H^+ + H$. The *dissociation energy*, D_e, is calculated as the difference between the energy of the bound molecule and the sum of the energies of the constituent parts at infinite separation.

Fixing the intermolecular distance, R, it is relatively straight forward to calculate variational bounds on the dissociation energy using simple trail wave functions (armed with no more than pen, paper, and some patience). Using a single 1s hydrogen orbital centered on each molecule gives a bond length of $R_e = 2.4a_0$ with a corresponding dissociation energy of $D_e = 0.07E_h$, see e.g. [46]. A natural next step in improving on this is to include more orbitals. Hinchliffe notes that adding 2p orbitals centered on the nuclei yields $R_e = 2.005a_0$ and $D_e = 0.09981E_h$. It is a testament to the simplicity of the problem that this is already within 0.1% and 3% of the experimental values of R_e and D_e, respectively [156].

As a test of our Hartree-Fock machinery, let us now see how close we can get using our gaussian basis functions. Since the system has only a single electron, we emply the un-restricted Hartree-Fock method throughout this section. In order to find the bond length we perform a brute force search for a wide range of R values. The resulting potential energy surfaces are shown in Fig. 12.1. First off, we may note from the figure that H_2^+ bonds under the Hartree-Fock approximation for all our basis sets as the energy minima are all $< 0.5E_h$: the energy of a free proton and a neutral hydrogen atom.

The calculated bond lengths and dissociation energies for all the basis sets tested can be seen in Table 12.3. The Slater type trial wave function described previously used only two orbitals for each atom, but we needed to use the 6-311++G** in order

Table 12.1: Geometries used in the validation Hartree-Fock calculations. The first part of the table is adapted from [22], while the latter part takes values from [155]. For descriptions of the bonding geometry, see e.g. [90].

Molecule		**Bond length** $[a_0]$	**Bond angle** [rad]
H_2	(Dihydrogen)	1.400	
CH_4	(Methane)	2.050	1.911 (Tetrahedral)
NH_3	(Ammonia)	1.913	1.862 (Trigonal pyramid)
H_2O	(Water)	1.809	1.824
HF	(Hydrogen fluoride)	1.733	
N_2	(Dinitrogen)	2.074	
CO	(Carbon monoxide)	2.132	
LiF	(Lithium fluoride)	2.955	
LiO	(Lithium monoxide)	3.203	
BeF	(Beryllium monofluoride)	2.572	
BeO	(Beryllium oxide)	2.515	

Table 12.2: Total energies in Hartrees, H_f, calculated with restricted (RHF) and un-restricted (UHF) Hartree-Fock. Two different basis sets are used, namely 3-21G and 6-311++G**. The Hartree-Fock limits are taken from Szabo & Ostlund [22]. Produced using github.com/mortele/HartreeFock commit f01b2f65f0d3a6dd3e0a35260686f6ea65292eb4.

	3-21G		6-311++G**		
Molecule	**RHF**	**UHF**	**RHF**	**UHF**	**HF limit**
H_2	$-1.122\,93$	$-1.122\,93$	$-1.132\,49$	$-1.132\,49$	-1.134
CH_4	-39.9769	-39.9769	-40.2092	-40.2092	-40.225
NH_3	-55.8705	-55.8705	-56.2145	-56.2145	-56.225
H_2O	-75.5854	-75.5854	-76.0529	-76.0529	-76.065
HF	-99.4598	-99.4598	-100.053	-100.053	-100.071
N_2	-108.300	-108.300	-108.972	-108.972	-108.997
CO	-112.093	-112.093	-112.770	-112.770	-112.791

to get a more accurate result. The 6-311++G** for hydrogen has eight primitive orbitals spread over six contracted orbitals. The difference illustrates how much more *natural* the Slater orbitals are for calculating molecular properties. The sole reason we are using gaussian type orbitals is ease of integral calculation. The Slater orbitals are solutions of the hydrogenic Schrödinger equation and an ideal starting point for designing multi-atomic wave function anzatses. The gaussians, on the other hand, can only try to combine multiple primitives in order to emulate the form of the Slater orbitals.

There appears to be two distinct jumps in accuracy: The first one when going from 6-31G (3s1s) to 6-31G** (3s1s1p) and a second one when making the change from 6-31G** (3s1s1p) to 6-311++G** (3s1s1s1s1p). The first one can be understood by the addition of three polarized gaussians which combine to form an orbital of p symmetry. It is clear from this that the H_2^+ ground function has a contribution from a bonding π-orbital[1]that we cover to some extent with the 1p gaussian. The second step up in accuracy comes after introduction of a diffuse s gaussian. It is clear that this covers a part of the σ-symmetric $2\sigma_g$ orbital that is not effectively covered by the other s-symmetric gaussians with larger exponents.

As a validation example, we feel confident that our unrestricted Hartree-Fock code works as it should for small systems. With a basis set consisting of only 15 contracted gaussians (17 total primitives) per atom, we calculate the dissociation energy of H_2^+ to within about 0.3% accuracy. As we will not be very focused on large Hartree-Fock basis sets in the present work, we declare ourselves satisfied with this and move on to testing larger diatomic systems.

12.2 Calculating the energies of the "ten-electron series"

We will now concern ourselves with calculating the total energy for a series of molecular systems totalling ten-electrons. We will also consider H_2, N_2, and CO. Geometry optimization will not be done here, but we will instead use bond lengths given in [22]. The atomic configuration used throughout this section is shown in Table 12.1. For methane (CH_4), a tetrahedral structure is assumed: The four hydrogen atoms placed at the four corners of a tetrahedron with the carbon atom in the center. The bond

[1]Linear diatomic molecular orbitals can be modelled as being comprised of states with a definite value of the axial (along the inter-molecular axis) angular momentum, $\ell = 0, \pm 1, \pm 2, \ldots$. Molecular orbitals with $\ell = 0$ are called σ-orbitals, while $\ell = \pm 1$ are called π_u-orbitals. Orbitals with non-vanishing electron density *between* the nuclei contribute to Coulomb shielding of the atoms from each other and thus promote the inter-atomic bond. These are called bonding orbitals (in contrast to orbitals for which this is *not* the case which are called anti-bonding). A subscript g means the molecular orbital has positive inversion symmetry, i.e. $\hat{P}\phi(r) = \phi(-r) = +\phi(r)$, where $\hat{P}$ denotes the party operator w.r.t. inversion through the inter-molecular center. A subscript u means the orbital has negative inversion symmetry. See e.g. [90]

angle refers to the H—C—H angle between any of the four hyrogen atoms. Similarily, the ammonia molecule (NH_3) assumes a trigonal pyramid structure with the shortest H—N—H angle between any two hydrogen atoms being the bond angle. Reference energies are also taken from [22], where the authors have used the 4-31G and 6-31G* basis sets, in addition to STO-G3.

As the the total energy is the primary quantity available in any *ab initio* calculation such as ours [22], it seems like a good place to start validation for larger systems. Using two different basis sets, 3-21G and 6-311++G**, we calculate the energies for all the ten-electron series molecules in addition to CO, N_2 and H_2. The results are shown in Table 12.2. We note that in every case, our results are better than the corresponding results of [22]using the 6-31G** basis set, as expected; we are using a larger basis. Our results are also consistent with the results of [3], from which we can directly compare results for the 6-311++G** basis set. We also notice that the restricted and unrestricted versions of our code seem to output the exact same values, meaning that forcing electrons pairs to occupy the same molecular orbitals is a valid assumption we can make in these cases.

Also shown in Table 12.2 are the Hartree-Fock limits for all the molecules. With the 6-311++G** basis sets, we are already within 0.02% of the limit for all the systems, except for H_2. For the dihydrogen molecule the relative error w.r.t. the Hartree-Fock limit is 0.13%. Using the cc-pVTZ basis set reduces this down to about 0.08%, and using the even bigger cc-pVQZ yields an error of the same order as for the other molecules, 0.04% at $E = -1.1335 E_h$.

Finding results which correspond perfectly to those of [22] and especially [3] (for the exact same basis sets) makes us even more confident in assuming our Hartree-Fock machinery works as it should.

Table 12.3: Dissociation energies and bond lengths calculated for the hydrogen molecule ion H_2^+ using six different basis sets. The experiemental value is taken from [155]. Produced using github.com/mortele/HartreeFock commit c251e9835d5534f0c957308fec585ec918cb2e94.

Basis set	**Bond length,** R_e $[a_0]$	**Dissociation energy,** D_e $[H_f]$	**Relative error w.r.t expt** D_e **[%]**
3-21G	1.994	0.083 151	18.96
6-31G	1.968	0.084 082	18.05
6-31G**	1.940	0.090 927	11.38
6-311++G**	1.984	0.101 180	1.384
6-311++G(2d,2p)	1.997	0.101 863	0.7183
cc-pVTZ	1.997	0.102 267	0.3245
Expt	2.003	0.1026	

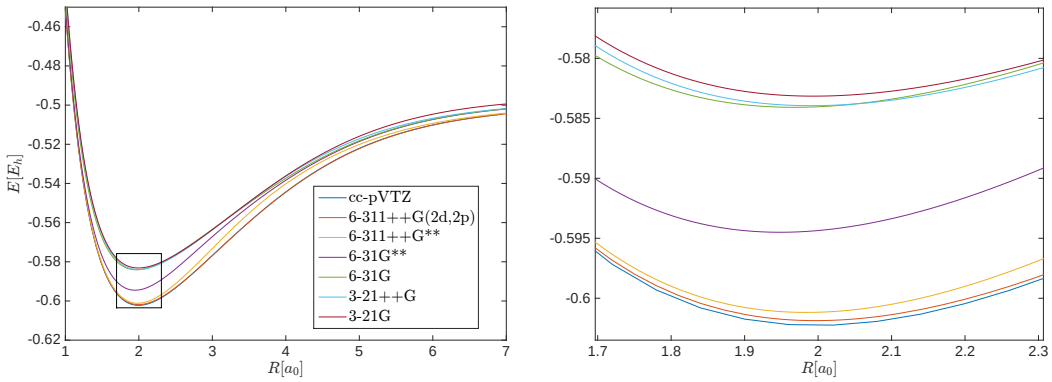

Figure 12.1: Energy as a function of the intermolecular distance, R, for the hydrogen molecule ion H_2^+. Six different hydrogen basis sets were used, yielding various values of the energy minima and equilibrium bond length. Detail around the minima shown on the right. Produced using github.com/mortele/HartreeFock commit c251e9835d5534f0c957308fec585ec918cb2e94.

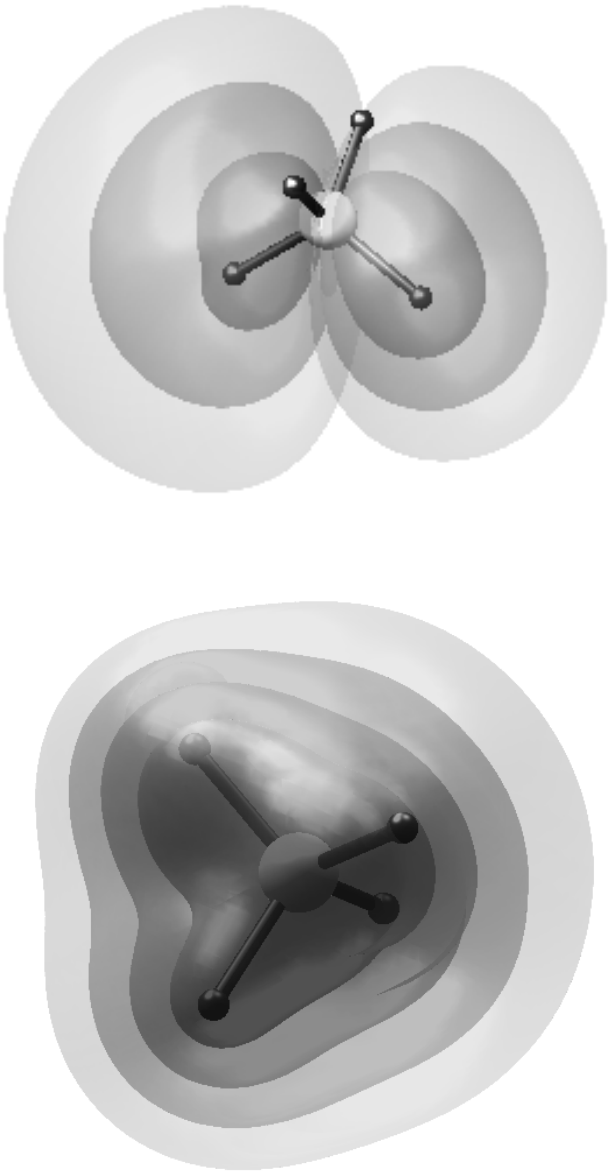

Figure 12.2: Example of a *molecular orbital* in the CH_4 molecule (top) and the electronic density (bottom) as calculated by the Hartree-Fock code using the diffuse-polarized 6–311++G** basis set for all atoms. The C and H atoms are shown as points connected by bars. Multiple isosurfaces of the orbital are shown, with increasing opaqueness denoting higher values. The density was calculated using the density matrix by $\rho(\mathbf{r}) = \sum_{pq} P_{pq}\psi_p(\mathbf{r})\psi_q(\mathbf{r})$. Produced using github.com/mortele/HartreeFock commit a587a2f88184db05d679311058525cebc7ef1ee2.

Chapter 13

Variational Monte Carlo validation tests

13.1 Non-interacting electrons

The simplest possible VMC calculations can be done on non-interacting, hydrogen-like atoms. With N electrons orbiting a single charge-Z nucleus, with no electron-electron interaction, the Hamiltonian takes the form

$$\hat{H} = \sum_{i=1}^{N} \left[-\frac{\nabla_i^2}{2} - \frac{Z}{|\mathbf{r}_i - \mathbf{r}_A|} \right]. \tag{13.1}$$

In this case the Schrödinger equation has a known solution, taking the form of a single Slater determinant filled with hydrogenic orbitals. As this is an actual closed form solution, the local energy becomes independent of the electronic configuration

$$E_{\mathrm{L}}(\mathbf{R}) = \frac{1}{\Psi(\mathbf{R})} \hat{H} \Psi(\mathbf{R}) = \frac{1}{\Psi(\mathbf{R})} E \Psi(\mathbf{R}) = E. \tag{13.2}$$

Since samples are all identical, the variance vanishes exactly. The orbital energies depend on the squared principal quantum number —c.f. the hydrogen atom energies—as

$$E_n^{\text{non-interacting}} = -\frac{Z^2}{2n^2}. \tag{13.3}$$

The total energy of the first and second row closed shell systems thus is

$$E_{\mathrm{He}}^{\text{non-interacting}} = -2\frac{2^2}{21^2} = -4, \tag{13.4}$$

$$E_{\mathrm{Be}}^{\text{non-interacting}} = -2\frac{4^2}{21^2} - 2\frac{4^2}{22^2} = -20, \quad \text{and} \tag{13.5}$$

$$E_{\mathrm{Ne}}^{\text{non-interacting}} = -2\frac{10^2}{21^2} - 8\frac{10^2}{22^2} = -200. \tag{13.6}$$

All three results are reproduced exactly by the VMC implementation, with standard deviations (no blocking) on the order of the machine precision, $\sigma \sim 10^{-15}$.

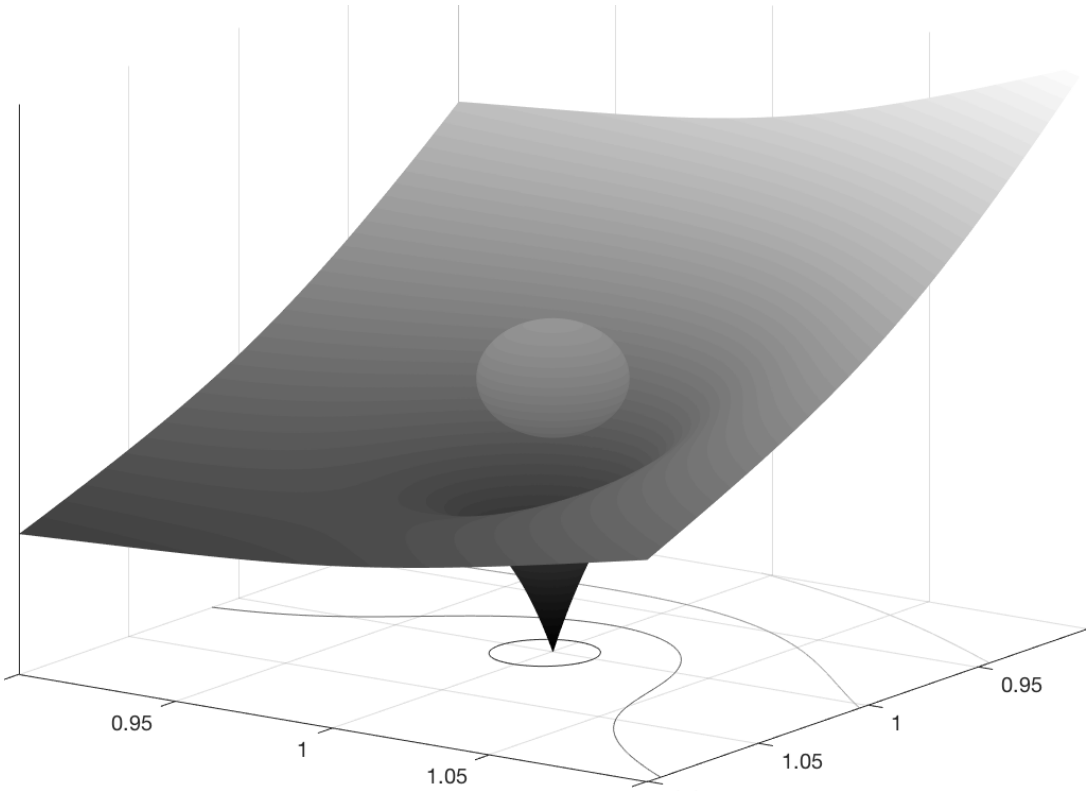

Figure 13.1: Detail of the Be wave function at the point where two electrons meet. Shown is $|\Psi(\mathbf{r}_1;\mathbf{r}_2,\mathbf{r}_3,\mathbf{r}_4)|^2$ for a part of the x_1-y_1-plane of the electronic coordinates of electron one, when the (opposite-spin) electron two is held at $\mathbf{r}_2 = (1,1,0)$. The placement of electron two is indicated by a floating sphere. The remaining two electrons are far separated and held fixed far from the location of this plot. The plot set into the surface beneath indicate the contours of the wave function. The single Be is located at $\mathbf{r}_A = \mathbf{0}$.

13.2 The effect of the Jastrow factor

The Jastrow factor introduces dynamic electron correlation to the wave function. As opposed to the Hartree-Fock scheme—in which all electrons interact only with the combined averaged charge density of the other electrons—dynamic correlation introduces instantaneous repulsion between electrons moving around in the molecular volume. In general, VMC is able to recover roughly 80-90% of the correlation energy—c.f. section 4.7—with highly optimized (multi-parameter) Jastrow factors (and possibly a linear combination of Slater determinants) [157].

The single-parameter, two-body Jastrow factor used in the present work ensures the electron-electron cusp—c.f. section 3.1.2—condition is upheld by reducing the value of the wave function whenever two electrons get close to each other. Same-spin electrons occupying the same spot in space is strictly forbidden by the Pauli

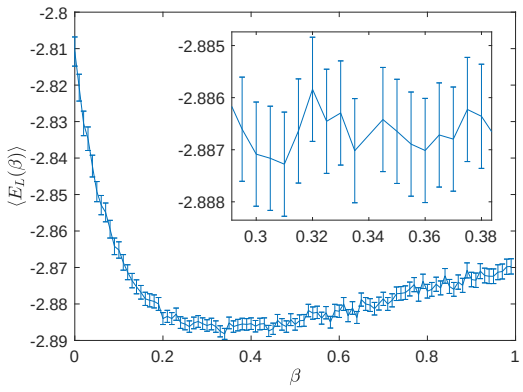

Figure 13.2: Energy expectation value as function of the Jastrow variational parameter, β. Error bars shown are estimated standard deviations obtained by blocking. The minimum found by a gradient descent search is located at $\beta = 0.347$. The inset shows details around the minimum.

princple, and the determinantal form of the wave function ensures that is vanishes exactly as such configurations. This is however not true of opposite-spin electrons. While the Jastrow factor changes the wave function in regions of configuration space where $\mathbf{r}_1$ and $\mathbf{r}_2$ are close, it does not make it vanish. Under certain conditions, the probability density at $\mathbf{r}_1 = \mathbf{r}_2$ is even higher than the surrounding configurations [158]. An example of the Jastrow factor's effect on the electron density is shown in Fig. 13.1. The plot shows the wave function as function of $\mathbf{r}_1$ varying over a plane intersecting $\mathbf{r}_2$, with $\mathbf{r}_2$, $\mathbf{r}_3$, and $\mathbf{r}_4$ held fixed. The Be nucleus is located at $\mathbf{r}_A = \mathbf{0}$, and we note the exponential decay away from the origin is the overall form. However, we also note a distinct *Jastrow hole* at the position of electron two.

Introduction of the Jastrow factor makes the VMC framwork in principle better suited to handle interacting many-electron systems than e.g. HF. We have built in a single variational parameter in $J(\mathbf{R})$, namely β. Recall that

$$J(\mathbf{R}) = \exp\left[\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{a_{ij}r_{ij}}{1+\beta r_{ij}}\right], \tag{13.7}$$

with a_{ij} depending on the spin-projections of electrons i and j. In order to obtain a better parametrization of the many-electron wave function than the single Slater determinant, we need to find the optimal value of β. The naive brute force method of just trying every single β you can think of works at the small scale, but becomes unfeasible as the system size increases. An example of such a search is shown in Fig. 13.2, for He using a STO-6G HF Slater determinant. The statistical error bars shown are standard deviation estimates obtained by blocking.

However, with a Slater already optimized with HF orbitals, we can ideally have a VMC wave function which depends on only *a single* parameter. This makes optimization much easier. Even so, in the present work we employ a simple gradient descent scheme. Between each run of the Metropolis algorithm, the value of β is updated according to

$$\beta_{k+1} = \beta_k - \gamma\nabla\langle E_{\mathrm{L}}\rangle, \tag{13.8}$$

Table 13.1: Example of the gradient descent algorithm applied to the He atom with hydrogenic orbitals. The already optimized $\alpha = 1.843$ was used for all iterations. The tollerance criteria for stopping was a change in β of $\varepsilon \leq 0.001$ which was achieved in 14 iterations, each with a modest 10^6 Monte Carlo cycles. Produced using github.com/mortele/VMC commit a4a2fd7a8698a7fe5a0118b9e78786e118e52d67.

Iteration	**Energy** $[E_h]$	β	**Gradient w.r.t.** β	**Change in** β
0	-2.8872	0.2	$-0.080\,519$	
1	-2.8897	$0.280\,52$	$-0.024\,79$	0.0805
2	-2.8918	$0.305\,31$	$-0.013\,644$	0.0248
3	-2.8887	$0.318\,95$	$-0.008\,953\,5$	0.0136
4	-2.8925	$0.327\,91$	$-0.007\,184\,1$	0.0090
5	-2.8929	$0.335\,09$	$-0.002\,936\,9$	0.0072
6	-2.8922	$0.338\,03$	$-0.001\,789\,3$	0.0029
7	-2.8894	$0.338\,82$	$-0.001\,968\,8$	0.0018
8	-2.8923	$0.340\,79$	$-0.002\,046\,6$	0.0020
9	-2.8890	$0.342\,83$	$-0.001\,075\,6$	0.0020
10	-2.8878	$0.343\,91$	$-0.001\,190\,9$	0.0011
11	-2.8895	$0.345\,10$	$-0.001\,962\,8$	0.0012
12	-2.8914	$0.347\,06$	$-0.001\,573\,9$	0.0020
13	-2.8891	$0.348\,64$	$0.000\,186\,6$	0.0016
14	-2.8866	$0.348\,45$		-0.0002

with the gradient calculated by [68]

$$\frac{\partial \langle E_{\mathrm{L}} \rangle}{\partial \beta} = 2 \left(\left\langle \frac{1}{\Psi[\beta]} \frac{\partial \Psi[\beta]}{\partial \beta} E_{\mathrm{L}}[\beta] \right\rangle - \left\langle \frac{1}{\Psi[\beta]} \frac{\partial \Psi[\beta]}{\partial \beta} \right\rangle \left\langle E_{\mathrm{L}}[\beta] \right\rangle \right). \tag{13.9}$$

The basic gradient descent uses $\gamma = 1$, but this can be extended to various more optimal alternatives[1]. An example of the gradient descent in action can be seen in Table 13.1, where we use the He atom as an example—this time with a Slater determinant occupied by hydrogenic orbitals (with a previously optimized value of the variational exponent α).

Once an optimization run has been done with relatively few Monte Carlo cycles and the energy minimum w.r.t. the variational parameters has been found we run

[1] See e.g. the method of Barzilai and Borwein which attempts to approximate the Hessian without having to actually calculate it [135]. This is an example of a larger class of Quasi-Newton methods for optimization in cases where the Hessian (or even the gradient) is too expensive to compute directly.

Table 13.2: Energies of first and second row closed-shell atomic and homogenous diatomic systems, calculated under VMC. Hydrogenic orbitals are used, with parameters α and β as given below. The given standard deviations are computed using the blocking technique. Reference energies taken from Filippi and Umrigar (Be_2), Buendía and co-workers (Be and Ne), and Moskowitz and Kalos (He and H_2) [157, 159, 160]. Varying numbers of Metropolis cycles used, from $4 \cdot 10^9$ for the lightest He to only $4 \cdot 10^7$ for the heaviest Be_2.

	α	β	**Energy** $[E_h]$	**Standard deviation,** σ	**Relative error w.r.t. reference** $[\%]$
He	1.843	0.347	$-2.890\,18$	$0.000\,075$	0.47
H_2	1.289	0.401	-1.1581	$0.000\,13$	1.43
Be	3.983	0.094	-14.503	0.0019	1.15
Be_2	3.725	0.246	-28.75	0.024	2.23
Ne	10.22	0.091	-127.91	0.0012	0.81

a computationally heavier *single-point* calculation with these parameters. With the optimal α and β, we find e.g. using the Slater type orbitals an energy of $-2.8901 E_h$ with standard deviation (after blocking) $\sigma \sim 10^{-4} E_h$.

13.3 First and second row closed-shell atoms and diatomics

If we want to run VMC on open-shell systems, we need to account for different spin configurations meaning we need to also suggest spin-flip Metropolis steps. This is a complication we want to avoid, so we will focus the testing now on closed-shell systems. First and second row closed-shell atoms include He, Be, and Ne with 2, 4, and 10 electrons, respectively. In addition we will include the homogenous diatomics Be_2 and H_2 in our validation set. The results of the validation runs are shown in Table 13.2.

We note that for He, the VMC approach improves considerably on the HF energy ($-2.8599 E_h$ at the 6-311++G** level). The Slater determinant consists of a single orbital only meaning the variational Monte Carlo single-parameter-orbital offers comparable freedom in functional form as the HF linear combinations. In addtion, the presence of the Jastrow factor improves heavily on the ability to model the electron-electron interaction and thus has quite a large effect on the resulting calculated energy. The same observation is true of the H_2 molecule, for which the HF energy is $-1.1325 E_h$ at the 6-311++G** level. Even though our simple VMC wave function recovers some of the missing correlation energy, we are still quite far off of the ref-

erence $-1.1746E_h$ [160].

For the case of Be, we find that our VMC estimate differs more from the reference energy of Buendía and co-workers of $-14.667E_h$. Even though Be is a closed-shell atom, the first excitation corresponds to a lower energy gap than the corresponding gaps for the noble gasses He and Ne. Essentially, the transition $E_{2s} \rightarrow E_{2p}$ is smaller than the first excitation possible in e.g. Ne, namely $E_{2p} \rightarrow E_{3s}$. Note carefully that even though the non-interacting hydrogen-like atoms have energies independent of the azimuthal quantum number l, this is of course not true of actual atoms for which the electronic interaction breaks the l-degeneracy.

In cases such as Be—where the HOMO-LUMO[2] energy gap is small—we expect the multi-configurational nature of the true wave function to play an important role. Such near-degeneracies are well handled by using e.g. a MCSCF[3] linear combination of determinants [161]. This means that our single Slater determinant is less suited to approximating the true wave function, resulting in less accuracy in calculated energies.

For the larger systems, the *single-parameter* determinant starts to rear it's proverbial ugly head. As the number of orbitals needed increases, the ability of a single variational parameter to approximate well all of them becomes more and more unrealistic. This drawback of our simple variational form begins to overshadow the Jastrow factor asset (as compared to HF) for larger atomic systems At $Z = 10$ the Hartree-Fock energy is more accurate than VMC at the 6-311++G** level (at $-128.527E_h$), only narrowly beating out the minimal 3-21G (at $-127.804E_h$ compared to $E_{\text{VMC}} = -127.91E_h$).

A comment on the overall accuracy compared to litterature results

In general, our variational wave function is much less sophisticated than corresponding ones found in the contemporary litterature. Even the parametrizations used by Moskowitz and Kalos in the early 1980s exhibit more parameters and greater freedom than our functional form [160]. In more modern VMC approaches, many more variational parameters are used. An example is the Jastrow factor of Buendía and co-workers, consisting of two-, and three-body terms with 17 distinct variational parameters [**buendie**]. Their Slater determinant combination is the result of an OEP calculation (*optimized effective potential* method, see e.g. Talman and co-workers [162]) which yields results roughly analogous to HF.

In short, competing with such results with our simple trial wave function is in no way realistic. Obtaining results differing from the litterature by on the order of $\sim 1\%$ is thus interpreted as a sign the machinery is working well.

[2] *Highest occupied molecular orbital* and *lowest unoccupied molecular orbital*, respectively.
[3] Multi-configurational self-consistent field methods, see e.g. [13].

Table 13.3: Energies calculated using the Gaussian fits of the hydrogenic orbitals, denoted HTO-nG (with n $= 1, 2, \ldots, 6$ representing the number of Gaussian primitives used for each orbital) for the He atom with *non-interacting* electrons. The *exact* wave function is the hydrogenic Slater, giving $\sigma_{\text{hydrogenic}} = 0$. Produced using github.com/mortele/VMC commit a4a2fd7a8698a7fe5a0118b9e78786e118e52d67.

Orbital	**Energy** $[E_h]$	**Standard deviation** $[E_h]$	**Relative error w.r.t. HTO** $[\%]$
HTO-1G	-2.8227	0.0034	29.43
HTO-2G	-3.8156	0.0025	4.61
HTO-3G	-3.9636	0.0020	0.91
HTO-4G	-3.9913	0.0016	0.22
HTO-5G	-3.9973	0.0014	0.07
HTO-6G	-3.9991	0.0012	0.02
Hydrogenic	-4.0	0.0	

13.4 Testing the gaussian orbitals

A natural next step in the validation is testing the implementation of the Gaussian type orbitals in the VMC program. In order to isolate only this part for testing, we consider the same non-interacting hydrogen-like atoms as in section 13.1. Fitting[4] Gaussian type orbitals to the hydrogen orbitals in a manner à la the STO-nG wave functions, we compare the ground state energy to the true E_0 for varying n. The results for He can be seen in Table 13.3, where we have dubbed the Gaussian fits HTO-nG.

We note that the implementation appears to work as it should. The next step is to include the 2s orbitals, and calculate the non-interacting energy for Be. This is done in Table 13.4. Evidently, the HTO-1G orbitals are qualitatively wrong, failing to capture the nodal structure of the 2s hydrogenic orbital with only a single primitive. With positive energy, the 1G Be does not admit bound state solutions.

Despite the catastrophic failure at HTO-1G, already with two primitives is the 2s node sufficiently well approximated to result in roughly $\sim 5\%$ relative error. The convergence looks strikingly similar to that of the He atom in Table 13.3.

Corresponding examples of calculations *with* the Jastrow factor and interacting electrons are shown in Tables **??** and **??**, for He and Be respectively.

[4] All curve fitting in the present work is done in MATLAB using the LAD (*least absolute deviations*, as opposed to the more familiar *least squared deviations*) approach and the trust-region algorithm proposed by Moré and co-workers [87–89].

Table 13.4: Energies calculated using the Gaussian fits of the hydrogenic orbitals, denoted HTO-nG (with n $= 1, 2, \ldots, 6$ representing the number of Gaussian primitives used for each orbital) for the Be atom with *non-interacting* electrons. The *exact* wave function is the hydrogenic Slater, giving $\sigma_{\text{hydrogenic}} = 0$. Produced using github.com/mortele/VMC commit a4a2fd7a8698a7fe5a0118b9e78786e118e52d67.

Orbital	**Energy** $[E_h]$	**Standard deviation** $[E_h]$	**Relative error w.r.t. HTO** [%]
HTO-1G	33.509	0.061	267.6
HTO-2G	−18.999	0.019	5.05
HTO-3G	−19.841	0.015	0.80
HTO-4G	−19.964	0.011	0.18
HTO-5G	−19.9804	0.0091	0.10
HTO-6G	−19.9921	0.0074	0.04
Hydrogenic	−20.0	0.0	

Table 13.5: Binding energies for He calculated using slater type orbitals (STO) and n gaussians fitted to the slater orbitals (STO-nG). Only the 1s slater type orbital is used. 10^7 monte carlo cycles were used for all simulations. An effective charge of $\alpha = 1.843$ was used as exponent for the STO, and $\beta = 0.347$ was used as parameter for the Jastrow factor. Produced using github.com/mortele/VMC commit a5a3580b2dc7c4a48594b853c32ad7082b99345c.

Orbital	**Energy** $[E_h]$	**Standard deviation** $[E_h]$	**Relative error w.r.t. STO** [%]
STO-1G	−1.775	0.0031	38.57
STO-2G	−2.675	0.0022	7.43
STO-3G	−2.841	0.0017	1.69
STO-4G	−2.877	0.0013	0.44
STO-5G	−2.886	0.0011	0.13
STO-6G	−2.887	0.0011	0.09
STO	−2.8897	0.000 86	

13.5 Cusp effects and *cusp corrections*

As discussed in section 3.3.4, the Gaussian orbitals do not satisfy the electron-nucleus cusp condition at $r_{iA} \to 0$. For the SCF methods—which depend on the orbitals only in the weak integral sense (see section 4.4.1)—this is not a big problem. When the only quantities entering the equations are integrals over all space, then minute imperfections in one tiny region of configuration space is not critical. When performing VMC integration, however, we are continously sampling the local energy at specific configurations. If E_{L} seemingly diverges for a small portion of these configurations, it poses a very real problem since sampling only a couple such points will cause the Monte Carlo average to become imprecise and make the variance explode. Especially worrying is the fact that this happens at the position of maximum probability, i.e. the only place where $\Psi(\mathbf{R})$ attains a maximum.

As we discussed in chapter 3, any finite linear combination of Gaussian primitives will yield a vanishing derivative at $r_{iA} \to 0$. For increasing numbers of primitives, we can force the contracted Gaussian into a STO-shape, which holds for smaller and smaller r_{iA}. This leads to linear combinations in which some primitives have enormously large exponents. When differentiating twice, these exponents make

$$\frac{\partial^2}{\partial r_{iA}^2}\psi_{\text{STO-nG}} \tag{13.10}$$

oscillate rapidly and with large amplitude close to the nucleus (see Fig. 13.4). A case study is presented in Fig. 13.3.

Let us now define the effective *local single-electron energy*,

$$E_{\mathrm{L}}^{\text{s-e}}(\mathbf{r}) = \frac{1}{\psi(\mathbf{r})}\left[-\frac{\nabla^2}{2} - \frac{Z}{|\mathbf{r}-\mathbf{r}_A|}\right]\psi(\mathbf{r}), \tag{13.11}$$

for the spatial orbital $\psi(\mathbf{r})$ [163]. In Fig. 13.3, we consider $E_{\mathrm{L}}^{\text{s-e}}$ for the 1s orbital of a non-interacting Be atom. Analogous to the actual full local energy, the single-electron local energy is constant in $\mathbf{r}$ when considering the *true* ground state. In this case, the true ground state is simply the 1s STO which is shown together with a STO-5G Gaussian fit. The top-left plot shows very good correspondence between the two, but differentiating reveals the subtle differences. Taking the Laplacian accentuates the spread, and shown in the bottom-left plot is the absolute difference

$$|\psi_{\text{STO-5G}}(\mathbf{r}) - \psi_{\text{STO}}(\mathbf{r})|. \tag{13.12}$$

For small r, the difference blows up. The bottom-right graph shows the proverbial bottom line: The difference between the correct cusp STO and the Gaussians diverges. Since $E_{\mathrm{L}}^{\text{s-e}}[\psi_{\text{STO}}] = E^{\text{s-e}}$ is constant,

$$\left|E_{\mathrm{L}}^{\text{s-e}}[\phi_{\text{STO-5G}}] - E_{\mathrm{L}}^{\text{s-e}}[\phi_{\text{STO}}]\right| = \left|E_{\mathrm{L}}^{\text{s-e}}[\phi_{\text{STO-5G}}] - E^{\text{s-e}}\right|. \tag{13.13}$$

The bottom-right plot is worryingly far from the constant it ideally should be.

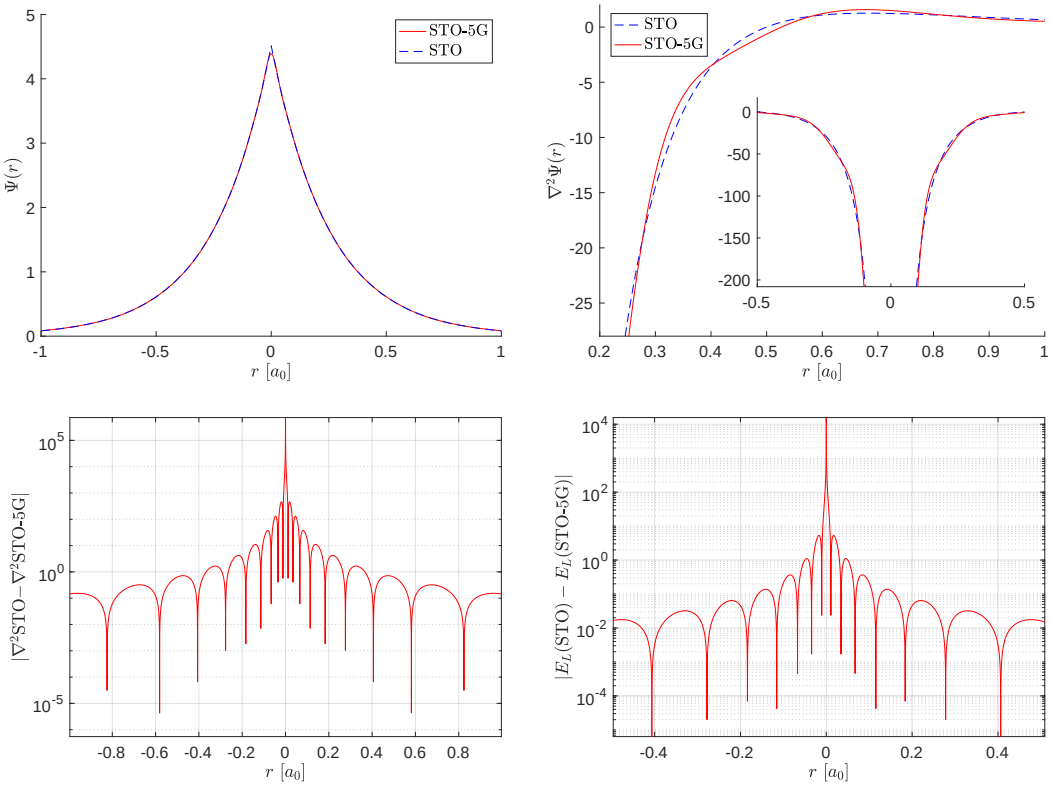

Figure 13.3: Example showing the cusp problems of the Gaussian linear combinations. A STO-5G fit to the 1s STO of a non-interacting Be atom is shown (top-left), along with a comparison of the double derivative of the two (top-right and bottom-left [logarithmic absolute difference]). The bottom-right shows the difference in *local single-electron energy*—as defined in Eq. (13.11)—for the two orbitals.

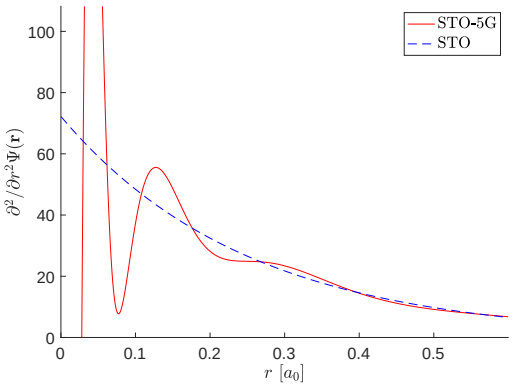

Figure 13.4: Example showing the double derivative of the STO-5G of Fig. 13.3, compared to the double derivative of the corresponding STO.

The case of Ne

For a noble gas—such as Ne—this effect is emphasized by the tight electronic structure. With a small spatial extent of only $r \sim 0.72a_0$, the Ne atom is more tightly bound than e.g. Beryllium at $r \sim 2.11a_0$ [164]. A higher density of eletrons around the nucleus point combined with the sharper exponential decay—neccessitating higher exponent Gaussians for fitting—give us problems in the VMC calculations. Shown in Table 13.7 are results from running Ne with varying STO-nG wave function ansatzes through the Metropolis machinery. Electron-interaction and Jastrow factor are both enabled for these calculations.

The calculated variance is an order of magnitude worse than for the lighter atoms, and the convergence to the STO energy is eratic at best. The results are, however, passable. We are within about $\sim 1\%$ of the reference STO energy, but there is really no reason to use the less stable STO-nG basis sets in the VMC calculations. The reason for employing Gaussian orbitals in the first place was only for ease of integration in SCF methods.

13.5.1 Cusp correction

Recall that these previous results were all ran with basis sets which by design mimic the STO orbitals. The cusp conditions are not met, but the local energy remains well-behaved for reasonably small values of the electron-nucleus distance. This all breaks down—however—when we consider e.g. the Pople family basis sets (see section 3.4). In such cases, handling the cusp problem explicitly post-HF calculations is the only way to obtain reasonable, low-variance results in a reasonable amount of (CPU-)time.

There are multiple ways to perform **cusp correction** of the contracted Gaussian functions. One approach is to use the Jastrow factor, including in it a term proportional to

$$J^{\text{nuclear}}(\mathbf{R}) \sim \exp\left[\sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A r_{iA}}{2(1+\gamma r_{iA})}\right]. \tag{13.14}$$

Table 13.6: Binding energies for Be calculated using slater type orbitals (STO) and n gaussians fitted to the slater orbitals (STO-nG). Only the 1s and 2s slater type orbitals are used. $5 \cdot 10^6$ monte carlo cycles were used for all simulations. An effective charge of $\alpha = 3.983$ was used as exponent for the STO, and $\beta = 0.094$ was used as parameter for the Jastrow factor. Produced using github.com/mortele/VMC commit a5a3580b2dc7c4a48594b853c32ad7082b99345c.

Orbital	**Energy** $[E_h]$	**Standard deviation** $[E_h]$	**Relative error w.r.t. STO** [%]
STO-1G	-10.10	0.023	30.00
STO-2G	-13.53	0.024	6.22
STO-3G	-14.03	0.022	2.76
STO-4G	-14.27	0.013	1.10
STO-5G	-14.41	0.012	0.12
STO-6G	-14.425	0.014	0.02
STO	-14.428	0.0090	

Table 13.7: Energies calculated using the Gaussian fits of the Slater type orbitals, STO-nG (with n $= 1, 2, \ldots, 6$ representing the number of Gaussian primitives used for each orbital) for the Ne atom. A STO calculations is presented for comparison. Note that the α and β parameters were not properly tuned to the variational minimum for this calculations. However, the key point is comparison of STO and STO-nG and in this regard the value of the energy is immaterial—the difference is what matters. Produced using github.com/mortele/VMC commit a4a2fd7a8698a7fe5a0118b9e78786e118e52d67.

Orbital	**Energy** $[E_h]$	**Standard deviation** $[E_h]$	**Relative error w.r.t. STO** [%]
STO-1G	-97.07	0.24	22.35
STO-2G	-115.90	0.23	7.29
STO-3G	-118.83	0.20	4.95
STO-4G	-124.09	0.28	0.74
STO-5G	-123.91	0.13	0.89
STO-6G	-123.63	0.13	1.11
STO	-125.02	0.10	

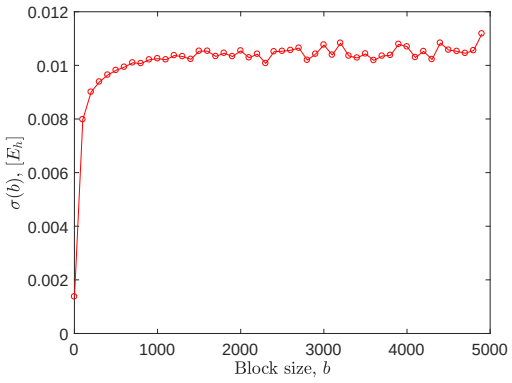

Figure 13.5: Example of the blocking procedure applied to the Be atom, running $N = 10^6$ Monte Carlo cycles. The notation $\sigma(b)$ denotes the standard deviation, calculated with a block size of b. We note the clear plateau starting at around $b = 1500$. Final blocking estimate for the standard deviation is calculated to be $\sigma(1500) = 0.011E_h$.

The γ parameter is another variational parameter which needs to be optimized variationally, analogous to β. See e.g. [161, 165, 166]. A different approach consists of modifying the orbitals of s-type symmetry directly, ensuring they satisfy the electron-nucleus cusp. Most proposed algorithms define a cut-off, inside which the s orbitals are replaced:

$$\psi_s(\mathbf{r}) = \begin{cases} \text{Contracted Gaussian} & \text{for} \quad |\mathbf{r}| > r_{\text{cutoff}} \\ \text{Replacement function} & \text{for} \quad |\mathbf{r}| \leq r_{\text{cutoff}} \end{cases} \tag{13.15}$$

The method of replacement differs, with various researchers using quintic splines (Manolo and co-workers), a fourth order polynomial (Ma and co-workers), or substitution by STOs (Manten and Lüchow), among other approaches [**manolo**, 163, 167]. An illustration of the idea is shown in Fig. 13.4, where replacement within some finite cutoff r_{cutoff} yields a smooth second derivative à la the STO.

Performing any form of cusp correction is unfortunately outside the scope of the present work. This means we are consigned to work with Slater type orbitals for VMC in the main body of this thesis.

13.6 Blocking

A short example of the blocking procedure is shown in the following. We consider the Be atom with an STO Slater-Jastrow wave function. Running 10^6 Monte Carlo cycles yields an energy of $E = -14.4937E_h$ with a naive non-blocking estimate of the standard deviation of $\sigma(1) = 0.00137E_h$. We define in the following $\sigma(b)$ to denote the value of the standard deviation, calculated with a block size b. The calculated blocking deviations are shown in Fig. 13.5, where we note a clear initial increase and a subsequent plateau.

As dictated by Flyvbjerg and Petersen, we interpret the (approximate) point at which the plateau starts to be an estimate of the correlation time τ [130]. In this case, that appears to be around $b = 1500$. The blocking estimate for the *true* standard

deviation thus becomes $\sigma(1500) = 0.011$, and the energy can be presented as

$$E = -14.494E_h \pm 0.011E_h. \tag{13.16}$$

Once the correlation length is known for a system, it is reasonable to assume the same correlation length holds for similar systems. For this reason, the actual blocking procedure needs only be performed a handful of times. When the correlation length is known for a particular system (or one closely related), the calculation of the blocking variance and standard deviations can be done on the fly directly in the C++ code. In order to avoid having to perform blocking too many times, we use a modest *over-estimate* of the correlation length τ in the present work. This over-estimate is then used for numerous more or less similar systems. This comes at the cost of essentially reporting *under-estimates* of the real accuracy of our program.

Chapter 14

Neural Network validation tests

14.1 Single variable curve fit

The natural place to start the testing of our Neural Network potential fit scheme is with a simple function of a single variable, $f : \mathbb{R} \to \mathbb{R}$. As an initial test, the specific functional form is of little importance. However—in the spirit of the present contex—we choose a Lennard-Jones (LJ) parameterization,

$$V_{\text{LJ}}(r) = \frac{1}{r^6} - \frac{1}{r^{12}}. \tag{14.1}$$

For the moment we forget about the normalization, and the scaling of the distances by the usual σ parameter. Using a simple network structure of a single hidden layer consisting of 10 neurons, we train for 10^3 with a data set consisting of 10^6 samples of the LJ potential for $0.9 \leq r \leq 1.6$. The resulting network output and training details are shown in Fig. 14.1. We note the network output and the training data coincide—approximately—perfectly after 1000 epochs of training, with the average squared difference between the validation points and the true potential is on the order of 10^{-7}.

From the bottom graph of Fig. 14.1, it is clear that an overall minimum has not yet been reached: the cost is still steadily decreasing (albeit slowly).

14.2 Approximating noisy data

When performing *ab initio* calculations using VMC the results will always be slightly distorted due to the statistical nature of the method. The statistical errors can be made arbitrarily small by increasing the sampling set, but for a finite number of Monte Carlo samples, it will never be identically zero[1]. Because of this, it is inherently es-

[1] Assuming for the moment that the *true* wave function is not known, in which case any number of cycles would give zero statistical error.

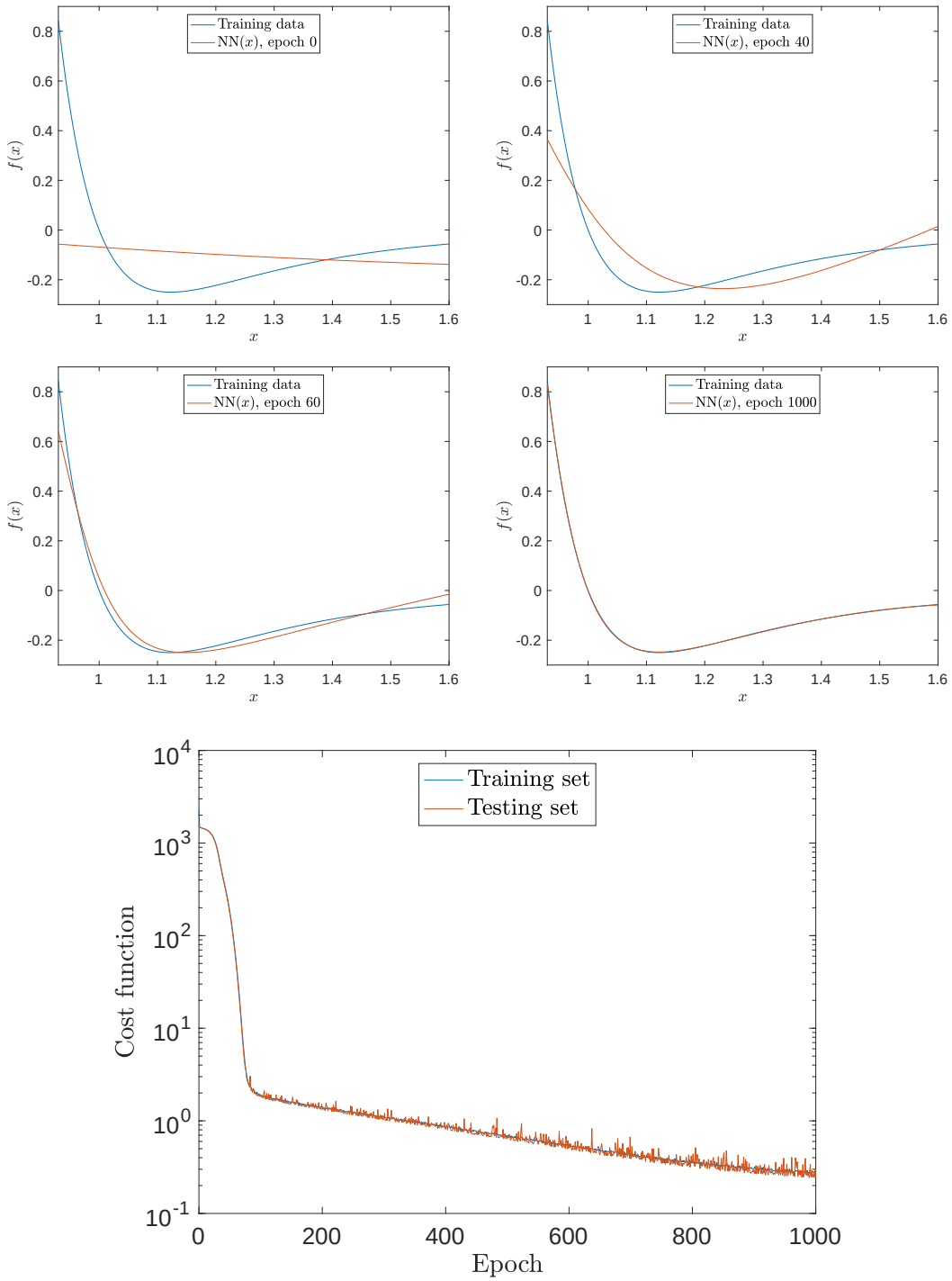

Figure 14.1: Samples showing the training of an ANN on samplings of the Lennard-Jones potential between $0.9 \leq r \leq 1.6$. The evolution of the network output with increasing epochs is shown (top) in addition to the cost as a function of epoch (bottom).

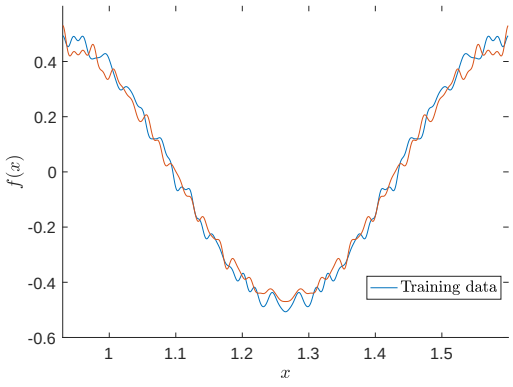

Figure 14.2: Example data set created to contain noise of different frequencies, as described in section 14.2. The red curve is the one used for the training described in the same section.

sential that our ANN potential is able to perform it's job in the presence of noisy data.

Thus, in order to test the suitability of our model in the presence of noise, we generate a sine curve with Gaussian noise. Since we ideally want the NN to be able to handle noise of a wide range of frequencies, we add the random fluctuations of different frequencies. In order to easily generate such a data set, we start from the Fourier coefficients of the data. We want the data to not be *dominated* by noise, so we set $a_2 = 1$, and then take

$$a_k \propto N\left[0, \frac{1}{2k}\right], \quad \text{for} \quad k = 30, 31, \ldots, 99, 100. \tag{14.2}$$

The $N(\mu, \sigma)$ denotes a random Gaussian of mean μ and standard deviation σ. Taking the *real* inverse Fourier transform of the $\boldsymbol{\alpha}$ vector, $f(x) \equiv \mathcal{F}^{-1}[\boldsymbol{\alpha}]$ now yields a sine curve with random noise of differing frequencies. An exmaple of such a data set is shown in Fig. 14.2.

For the training we use a neural network consisting of a single layer of 20 neutrons. A selection of training snapshots are shown in Fig. 14.4. Also shown is the magnitude of the cost function plotted versus the epoch number. In order to validate the training, we use a separate set of data points—which the NN is never trained on—to check the state of the training. Since we are interested in a network solution independent of the noise, only capturing the underlying shape, we use a validation set with the same structure but with *different* noise. For simplicity, we use the data set shown in red in Fig. 14.2. We note that the cost functions relative to both data sets fall off in mostly the same fashion, with an apparent slight difference in the fully trained state.

Crucially, we see no signs of *over-training*, which would cause the NN output to begin to follow the structure of the training data *noise*. Over-training can be seen from the contiuing fall of the cost function relative to the training data, with a simultaneous *increase* in the cost function relative to the validation set. We note that the randomization of the order of the training input, aswell as the online learning scheme efficiently counteracts the over-training phenomenon in the current model.

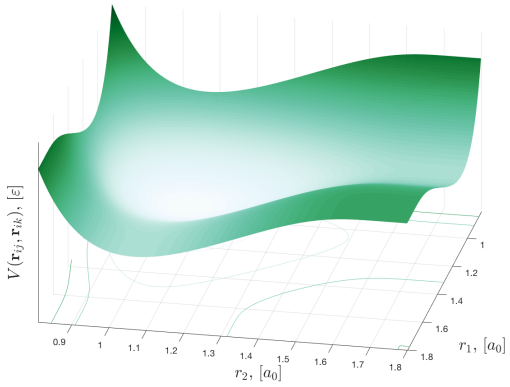

Figure 14.3: The radial part of the SW molecular dynamics potential, used as an example data set for multi-variable potential fitting. The set contains combinations of r_1 and r_2 values for $0.8 < r < a = 1.8$.

14.3 Multi-variable fitting

So far we have tried single-variable functions only. In the following, we explore the performance of our ANN model as a multi-variable curve fitting tool. We choose—entirely arbitrarily—a functional form based on the Stillinger-Weber (SW) molecular dynamics potential, originally developed to model Si interacting atoms [168]. We consider configurations of three Si atoms, indexed by i, j, and k. For simplicity, we use only the radial component, holding the θ_{ijk} angle constant at $\pi/2$. The radial part of the SW potential (in units of the ε parameter) takes the form between

$$V(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) = \phi(r_{ij}) + \phi(r_{ik}) + \phi(r_{jk}), \tag{14.3}$$

with

$$\phi(r) \equiv \begin{cases} A\left(\dfrac{B}{r^4} - \dfrac{1}{r}\right) \exp\left[\dfrac{1}{r-a}\right] & \text{for} \quad r < a \\ 0 & \text{for} \quad r \geq a \end{cases} \tag{14.4}$$

We use the parameters $A = 7.049556277$ and $B = 0.6022245584$ according to Stiller and Webers original suggestions, with the distance scaling $a = 1.8$.

The training data set is shown in Fig. 14.3. We use a grid of r_{ij} and r_{ik} values with $0.8 \leq r \leq a$. For the training, we set up a NN with three layers of 20 neurons each. We allow the training to run a set of 10^6 samples of the potential $V(r_1, r_2)$ for a total of 10^4 epochs. The training is visualized in Fig. 14.5, from which we note that the resulting network output is indistinguishable from the training set of Fig. 14.3. From the cost evolution we conclude that the ANN model we implement is able to also approximate multi-dimensional data sets satisfactorily.

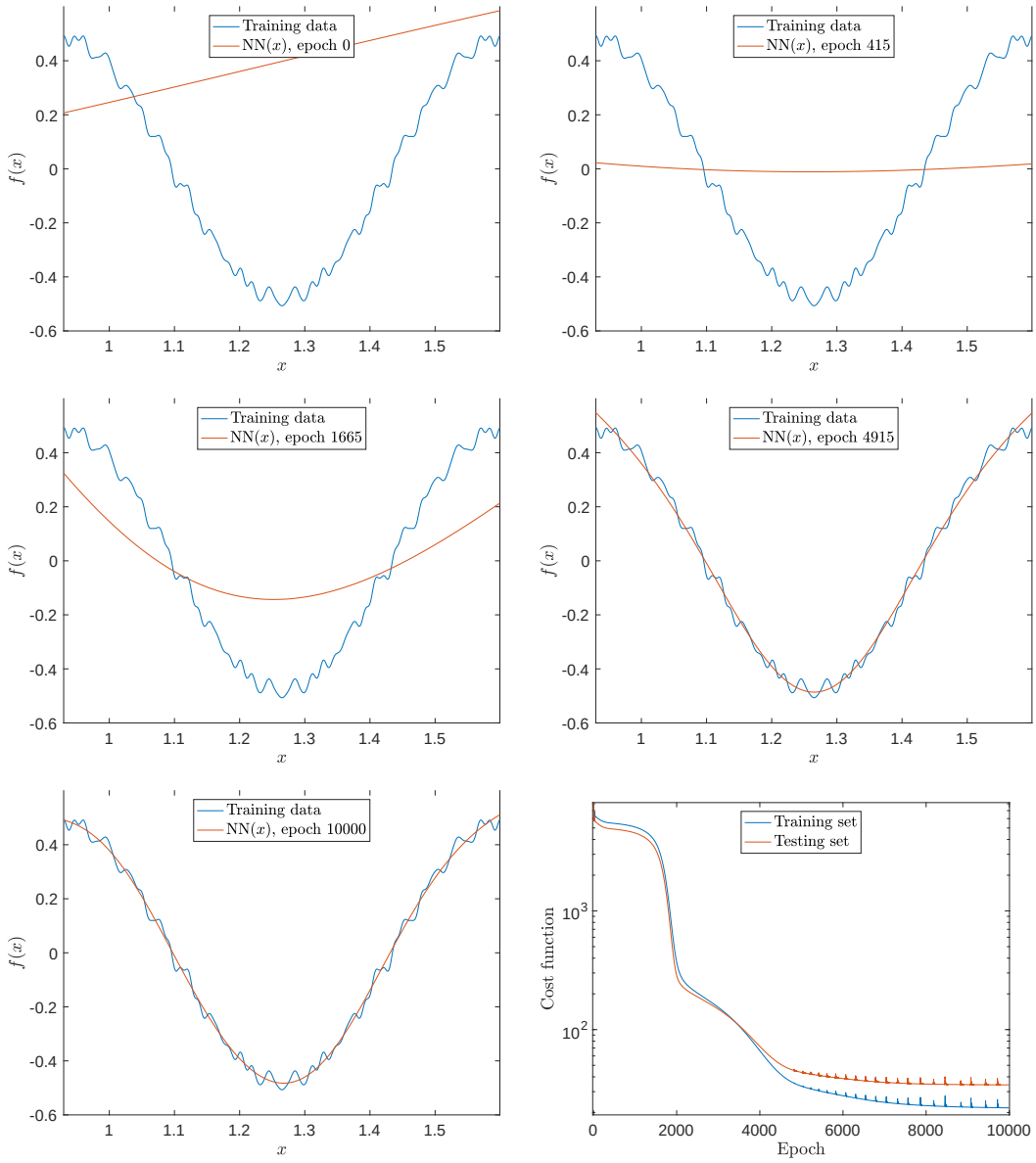

Figure 14.4: Training snapshots for the noisy data test of section 14.2. The training set is shown in blue, while the NN output is shown in red. The epoch number—denoting the length of the training—is inset for each graph. The bottom right plot shows the evolution of the cost function—$\|\mathbf{y} - \hat{\mathbf{y}}\|_2$—as a function of the epoch number for the training data and the separate validation set.

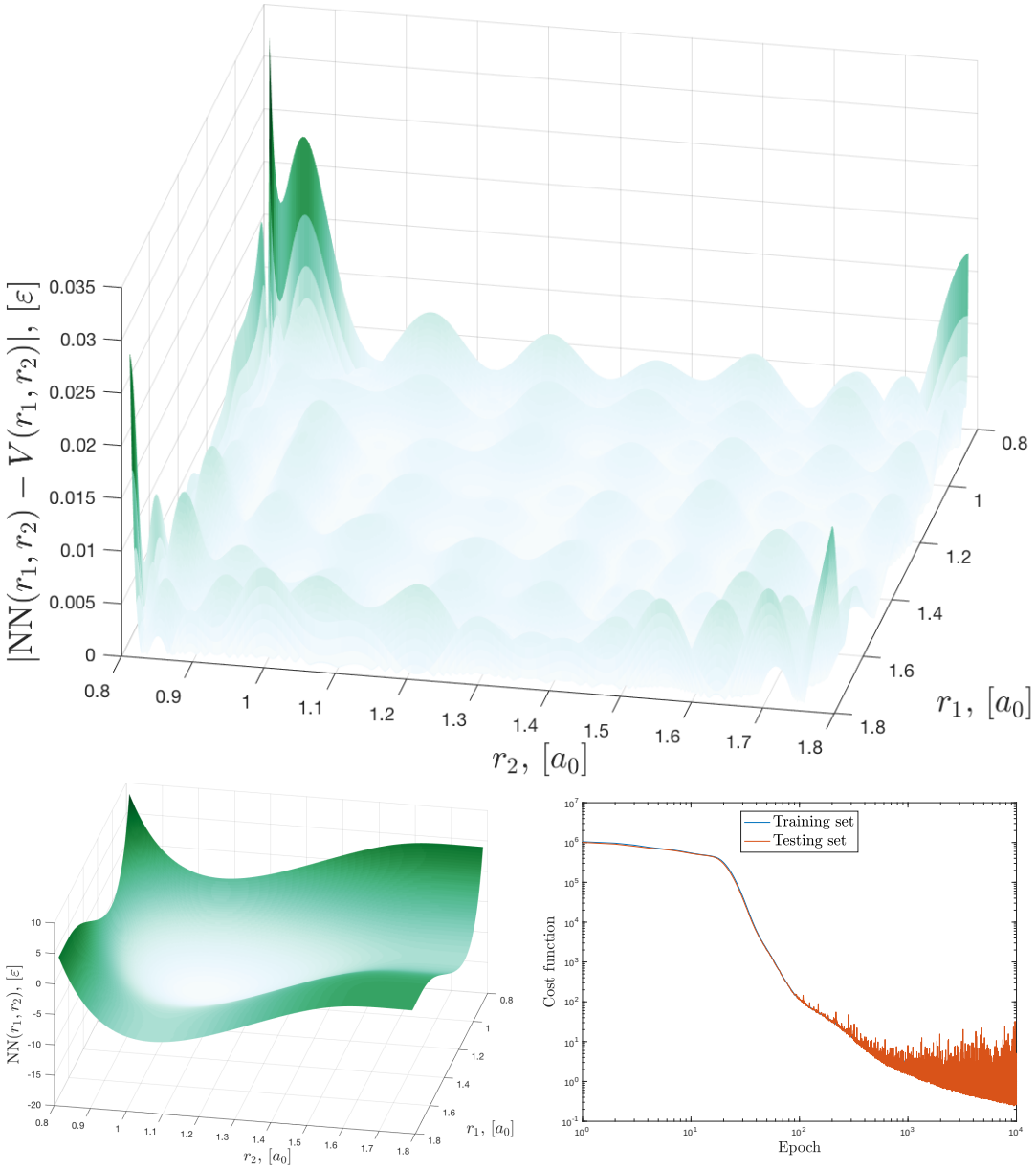

Figure 14.5: Example network trained to approximate the radial SW potential shown in Fig. 14.3. The absolute error w.r.t. the training data (top) and the network output post training (bottom left) is shown, as well as the evolution of the cost as a function of epoch number (bottom right). We note that the functional form of the output from the network is indistinguishable from the training set shown in Fig. 14.3. The approximation error is on the order of $\sim 0.005\varepsilon$ for the interior points, but increases *some* towards the edges of the training set. Note that the validation cost closely follows the training cost, indicating no overtraining is happening.

14.4 Training on *ab initio* data

Out first foray into the use of *ab initio* data will be a case study of the required amount of data. Since QM calculations are exceedingly expensive from a computational perspective, it is interesting to explore how many calculations we *need* to perform in order to have enough data to train a ANN adequately. For this test we employ the H_2^+ data set—presented in section 12—at the 6-311++G** level.

In order to explore the training behaviour of the NN on small data sets, we train different networks using only parts of the complete data set, but retain—crucially—the complete set for validation. In Fig. **??** we present data from training runs using $N = 50, 75, 100, \ldots, 475, 500$ total data points. The removal of training pairs is done by excluding a given number of data points at random:

```
xData = np.asarray(inputFileData).reshape([dataSize, self.inputs])
if N < dataSize :
        # If the requested size is smaller than the total data size, we prune
        # the set at random.
        toRemove = dataSize - N
        toRemove = np.random.choice(    np.arange(dataSize),
                                                                toRemove,
                                                                replace=False)
        xData = np.delete(xData, toRemove)
```

This ensures we are still training on a representative subset of the full data set.

From the visualizations of Fig. 14.6, it is clear that handling small data sizes is something our model handles in stride. While the larger sizes perform better on the whole, the statistical nature of the initialization and training process seems to introduce a significant variance in the resulting network quality. We note that in general, the sizes ≥ 200 all achieve comparable performance w.r.t. the total cost function, with the only outlier being the NN trained on the smallest data set. This fact that none of the cost function graphs appear to have plateaued completely indicates two things: Primarily, 10^4 epochs may not be suffcient training time for small data sets. Secondly, comparing the results of the bottom right hand plot with the top right hand one, shows that a small cost function value is not neccesarily synonymous with being able to reproduce critical parts of the functional form.

The "so far" minimum—shown in the bottom right hand side of Fig. 14.6—is defined at epoch $\#t$ as

$$\min(\text{cost}) \text{ so far} \equiv \min\{\text{cost}_k : k \leq t\}, \tag{14.5}$$

i.e. at step t the minimum so far is the minimum across all steps prior to (and including) t.

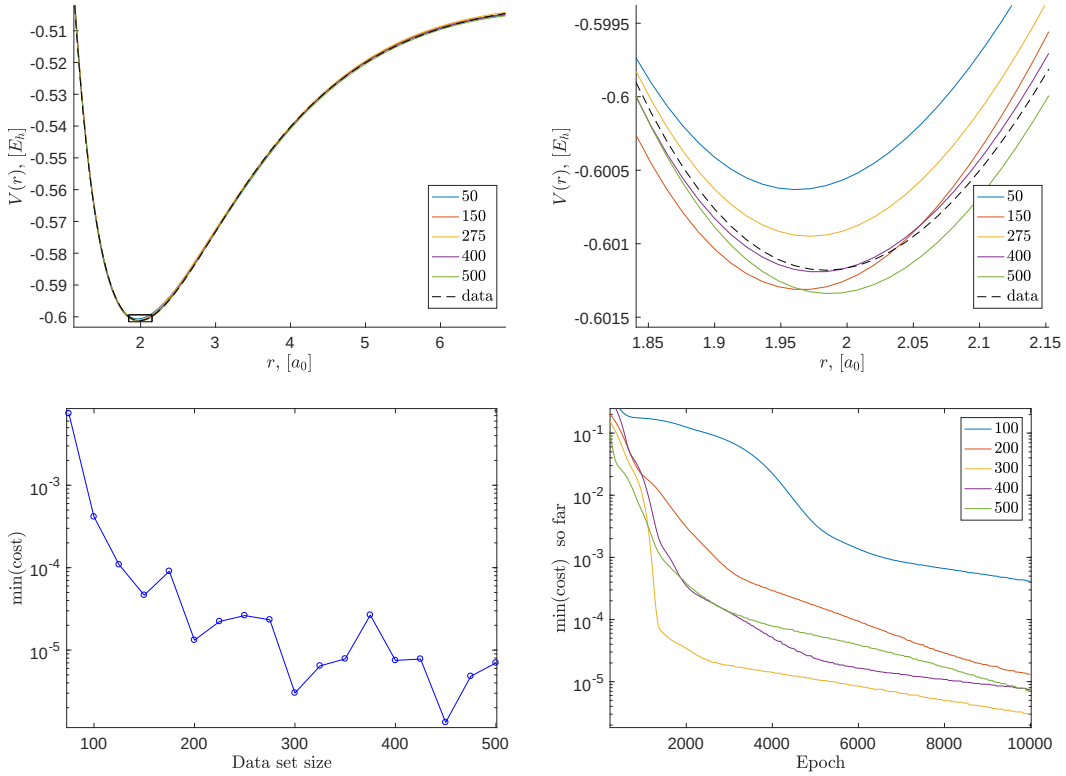

Figure 14.6: Overview of the effect of small sample sizes on the ANN training process. The trained functional form, along with the training data is shown in the top left, with a magnified inset around the minimum shown top right. Further, the final minimum of the cost function for each data set size (bottom left), and the *minimum attained so far* is shown as a function of the epoch time (bottom right).

Part IV

Conclusions and future work

14.5 Conclusions

In this thesis we have demonstrated the feasibility of performing multiscale simulations—by bridging first principles quantum mechanics and machine learning—and implemented a general computational framework in which such simulations can be run. Crucially, our framework is (nearly) parameter-free in the sense that the resulting simulations depend very weakly on any parameter directly put into the

14.6 Prospects for future work

The comprehensive nature of the this thesis means we have only just scratched the surface of possibilities. The possible extensions of the present work fall naturally into two groups: *improving* or *extending*.

Work involved in the former category would entail optimizing and stream-lining the developed QM calculation code in order to facilitate applications on heavier systems. Currently, the Hartree-Fock program is limited to about 100 total basis functions per calculation with *reasonable* speed. The quantum Monte Carlo code—being an inherently slower scheme—is currently limited to only a small handful of first and second row atoms per calculation. Only light profiling and optimization has been performed for the present work, leaving a large body of possible computational improvements open.

In addition to simple optimization, there are numerous ways in which small assumptions and simplifications can be employed to drastically speed up *ab initio* calculations. Techniques for Hartree-Fock involve e.g. integral pre-screening, density fitting, early density contraction, or multipole techniques for handling longer range interactions, among many others. Ultimately, this leads to a *near* linear HF scheme. For VMC, possbilities involve for example using a more sophisticated optimization algorithm, including pseudo-potential replacement of core electrons, or proper inclusion of HF orbitals. For the DFT code, the single bottleneck currently is calculating four center interaction elements for the Coulomb interaction. Unless some fraction of *exact* exchange[2] is needed in the exchange-correlation potential, these integrals can be computed orders of magnitude more efficiently by a Poisson solver technique.

The second category of *extensions* to the present work may include e.g.

More precise ab initio calculations.
The accuracy of the *ab initio* energy parametrization (in terms of nucleonic coordinates) is of crucial importance to the overall predictive power of the multiscale modelling framework described in this thesis. Therefore a natural augmentation of the present work is either applying more sophisticated QM

[2]This is an example of awful but widespread notation. The $\hat{K}$ operator represents the *exact* exchange only under the assumption of a single-Slater Hartree-Fock wave function, and not (as the name might imply) the exact exchange energy functional for the *true* wave function.

calculations, or improving the accuracy of the VMC scheme by parametrizing more complicated wave function ansatzes (three-body and higher order Jastrow factors, multiconfigurational Slater determinant, etc.).

ANN parametrization of effective three-body (and higher order) interactions.
Only the most rudimentary potential energy surface (PES) fitting was done in this thesis. As demostrated in chapter 10, the extension to higher dimensional energy hypersurfaces is in principle straight forward. As most molecular dynamics systems of interest depend critically on effective three(or higher)-body interactions, an obvious extension of the current work is inclusion of such terms in the ANN training and subsequent MD simulations.

Implementing the Behler-Parrinello method.
Taking the previous point to the extreme, we may do away with the effective two-body, three-body, etc. parametrization entirely. Instead, we may compute the *ab initio* energy of the atoms in their chemical environment directly. This involves QM calculations currently inaccessible to the codes developed in the present work (they would be unfeasibly slow), but are in principle possible after some optimization. Since the calculated energies exhibit a high degree of symmetry w.r.t. rotations and interchange of atoms, processing them through so-called symmetry functions is neccessary prior to ANN training. For more information, see [1, 2].

Devise an efficient strategy for sampling the needed molecular configurations.
Sampling the PES on a grid is almost certainly *not* the most efficient scheme for generating training data inputs to the ANNs. A relatively standard way to find which configurations to sample is to run hybrid MD simulations (e.g. Car-Parrinello or Born-Oppenheimer MD) on small systems, and picking appearing geometries in some random fashion. Based on this approach it might be prudent to devise a Markov chain scheme in which a "walker" traverses the configuration space of nucleonic coordinates. This should be able to exploit the fact that a fully converged HF density matrix, or a fully optimized VMC ansatz for a set of nuclei at positions $\mathbf{R}$ is an excellent starting point for the solution at $\mathbf{R} + \Delta\mathbf{R}$ for "small" nucleonic displacements $\Delta\mathbf{R}$.

Appendices

Appendix A

Natural units: Hartree atomic units

When working within a specific branch of physics, it is often useful to deviate from the every-day SI units of measurements and instead use units which are *natural* to the systems under study. Since we are working with "small" systems, the SI *meter*, *second*, *kilogram*, and *coulomb* are of little use to us. Instead we will work in a system of units in which we define the mass of the electron, m_e, to be the scale by which we measure all other masses. This obviously means the numerical value of the electron mass becomes unity, $m_e = 1$. In the same way, we will use Planck's constant, $\hbar$, as the scale by which we measure angular momentum and action, the electron charge, e, will be our scale for electrical charge, and finally Coulomb's constant, k_e, will be our scale of electric permittivity.

The usual way to state this is to set $\hbar = e = m_e = k_e = 1$, and the system of units derived from these four definitions is called Hartree atomic units. We can think of this as the *natural* system of units for the Hydrogen atom system. To better see why this is the case, let us combine these four quantities in such a way as to produce a length.

In terms of the four fundamental dimenions of physics: Length(L), time(T), mass(M), and charge(C), the units of $\hbar$, m_e, e, and k_e are $[\hbar] = \mathrm{ML}^2\mathrm{T}^{-1}$, $[m_e] = \mathrm{M}$, $[e] = \mathrm{C}$, and $[k_e] = \mathrm{ML}^3\mathrm{C}^{-2}\mathrm{T}^{-2}$, respectively. Combining arbitrary powers of these four constants gives

$$\begin{aligned}[\lambda(a,b,c,d)] = \left[k_e^a \hbar^b m_e^c e^d\right] &= \left(\mathrm{M}^a\mathrm{L}^{3a}\mathrm{C}^{-2a}\mathrm{T}^{-2a}\right)\left(\mathrm{M}^b\mathrm{L}^{2b}\mathrm{T}^{-b}\right)\left(\mathrm{M}^c\right)\left(\mathrm{C}^d\right) \\ &= \mathrm{L}^{2a+3b}\mathrm{T}^{-a-2b}\mathrm{M}^{a+b+c}\mathrm{C}^{-2b+d}. \end{aligned} \tag{A.1}$$

There is exactly one way to realize a length from these exponents, i.e. solving the four equations $2a + 3b = 1$, $-a - 2b = 0$, $a + b + c = 0$, and $-2b + d = 0$: $a = -1$, $b = 2$, $c = -1$, and $d = -2$. This means that the natural length scale of our problem is simply (up to a numerical constant)

$$\mathrm{L}_{\text{scale}} = a_0 = k_e^{-1}\hbar^2 m_e^{-1} e^{-2} = \frac{\hbar^2}{k_e m_e e^2} = \frac{4\pi\varepsilon_0\hbar^2}{m_e e^2}, \tag{A.2}$$

which re recognize as simply the *Bohr radius*.

We can go through this same exercise to find a natural *time* scale for our system. There is a unique way to combine the exponents a, b, c, and d in order to realize a time, namely $a = -2$, $b = 3$, $c = -1$, $d = -4$, or

$$\mathrm{T}_{\text{scale}} = k_e^{-2}\hbar^3 m_e^{-1} e^{-4} = \frac{\hbar^3}{k_e^2 m_e e^4} = \frac{\hbar a_0}{k_e e^2}. \tag{A.3}$$

This is the revolution time of an electron in the lowest lying hydrogen state in the Bohr model (apart from a factor of 2π).

From a_0 and $\mathrm{T}_{\text{scale}}$ we can find the natural energy scale,

$$\mathrm{E}_{\text{scale}} = m_e \frac{a_0^2}{a_0^2 \left(\frac{\hbar}{k_e e^2}\right)^2} = \frac{m_e k_e^2 e^4}{\hbar^2} \equiv E_h, \tag{A.4}$$

which we will call a Hartree.

Finally, before we go on we may use the expression for the *fine structure constant* to find the numerical value of c in this system. From

$$\alpha = \frac{k_e e^2}{\hbar c} \Rightarrow c = \frac{k_e e^2}{\hbar \alpha} = \frac{1}{\alpha} \simeq 137, \tag{A.5}$$

after substituting $\hbar = e = k_e = 1$.

Appendix B

Basics of numerical integration

Riemann integral and Riemann integrable functions

Given a function $f(x)$ and a closed finite subset of $\mathbb{R}$, $[a, b]$ with $a < b$, a *Riemann sum* of f is defined as the sum of values attained on n sub-intervals of $[a, b]$, i.e.

$$S_n = \sum_{i=1}^{n} (x_i - x_{i-1})\, f_i. \tag{B.1}$$

The x_is here define the partitionining into sub-intervals $[x_{i-1}, x_i]$ (i.e. $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$), while $f_i \equiv f(\xi_i)$ with ξ_i *some* point in sub-interval i.

A sufficient condition for the *Riemann integral* to exist for the function f is that *any* such sum (any choice of x_i [for which $\max_i |x_i - x_{i-1}| \to 0$] and ξ_i) converge to the same value in the limit $n \to \infty$ [169]. In this case we say

$$\lim_{n\to\infty} S_n = S = \int_a^b f(x)\,\mathrm{d}x, \tag{B.2}$$

and that f is Riemann integrable.

A less strict, but still sufficient condition is to chose $\overline{f_i} = \max\{f(x) : x \in [x_{i-1}, x_i]\}$ and $\underline{f_i} = \min\{f(x) : x \in [x_{i-1}, x_i]\}$ and then only demand that the two sums converge to a common limit [170],

$$\begin{aligned}
\lim_{n\to\infty} \overline{S_n} &= \lim_{n\to\infty} \sum_{i=1}^{n} (x_i - x_{i-1})\overline{f_i} \\
&= \lim_{n\to\infty} \sum_{i=1}^{n} (x_i - x_{i-1})\underline{f_i} \quad = \quad \lim_{n\to\infty} \underline{S_n} \\
&\qquad\qquad\qquad\qquad\qquad\qquad = \int_a^b f(x)\,\mathrm{d}x.
\end{aligned}$$

Although easier than checking *every possible* Riemann sum, checking that the two upper and lower sums converge to a common limit is still a somewhat tedious

procedure for checking integrability. In fact it turns out that a sufficient condition on f is that it is continous and bounded on $[a, b]$ [169]. The latter condition is not neccessary on a finite interval since all continous functions on a closed finite domain are bounded according to the extreme value theorem. However, if we extend the limits of integration to an infinite interval, for example $[0, \infty)$, then the boundedness of f is not guaranteed by the continuity we need to explicitly demand $|f(x)| < \infty$ for all $x \in [a, b]$.

It is easy to see that the converse is *not* true. Any Riemann integrable function is not automatically continous. Take for example the integral over $[0, 1]$ with the step function

$$f(x) = \begin{cases} 1 & \text{if} \quad x > 1/2 \\ 0 & \text{else} \end{cases}. \tag{B.3}$$

Even though the upper and lower Riemann sums both attain the value $1/2$ in the limit $n \to \infty$ and the function is Riemann integrable, it demonstrably is not continous. A more careful analysis shows that a less strict but sufficient condition on f is that it be continous *almost everywhere* on $[a, b]$ (i.e. continous on all of the interval, except possibly on a subset $C \subset [a, b]$ with measure zero) [133]. With this condition, the converse also holds.

Newton-Cotes quadrature

Since the Riemann integral is defined in terms of the limit of a sum, numerical approximations to it arise naturally from any scheme for choosing ξ_i and the partitioning. One of the simplest possible approximations is to take the midpoint value of each sub-interval to be ξ_i with a uniform mesh of equispaced x_is. This constitutes the **midpoint rule** [169],

$$I \approx \sum_{i=1}^{n} f(x_{i-1} + \Delta x/2)(x_i - x_{i-1}) = \Delta x \sum_{i=1}^{n} f(x_{i-1} + \Delta x/2), \tag{B.4}$$

where $\Delta x \equiv (x_i - x_{i-1})$ which is the same for all i.

Instead of the midpoint, we can use the *average* of the left and right endpoints of the subinterval as f_i. Geometrically, this means we are approximating the integral of each sub-interval by the integral over a right trapezoid with base points at $(x_{i-1}, 0)$ and $(x_i, 0)$ and upper point at the function values $(x_{i-1}, f(x_{i-1}))$ and $(x_i, f(x_i))$. The resulting approximation is known as the **trapezoidal rule** [16],

$$I \approx \sum_{i=1}^{n} \frac{f(x_{i-1}) + f(x_i)}{2}(x_i - x_{i-1}) = \Delta x \sum_{i=1}^{n} \frac{f(x_{i-1}) + f(x_i)}{2}. \tag{B.5}$$

Yet another numerical scheme arises from replacing the integrand in each subinterval with an interpolating polynomial of degree two, which by construction coincides with f at the endpoints and the midpoint. This constitutes **Simpson's rule**

[169],

$$I \approx \sum_{i=1}^{n}(x_i - x_{i-1})\left(\frac{f(x_{i-1}}{6} + \frac{4f(x_{i-1} + \Delta x/2)}{6} + \frac{f(x_i}{6}\right)$$
$$= \frac{\Delta x}{6}\sum_{i=1}^{n}\Big(f(x_{i-1}) + 4f(x_{i-1} + \Delta x/2) + f(x_i)\Big). \tag{B.6}$$

All three approximations are examples of Newton-Cotes quadrature rules, which approximate the intergral by replacing the integrand by interpolating polynomials of order k on each of the n sub-intervals. We can build arbitrarily high order methods by constructing higher order interpolating polynomials within each interval. The interpolation procedure is described for example in [171]. We have just seen Newton-Cotes method for orders zero (midpoint rule, zero order polynomial [constant]), one (trapezoidal rule, linear polynomial), and two (Simpson's rule, quadratic polynomial). The next few commonly used methods are the third order *Simpson's 3/8 rule* and the fourth order *Boole's rule*.

Gaussian quadrature

Note that so far we have assumed the sub-intervals to all be the same size. If we drop this requirement, we can construct more advanced rules which exploit some convenient properties of orthogonal polynomials. Gaussian quadrature rules are a set of schema for numerical intergration in which we extract a *weight function* from the integrand

$$\int_a^b f(x)\,\mathrm{d}x = \int_a^b W(x)g(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i g(x_i). \tag{B.7}$$

The weight function is associated with a set of orthogonal polynomials, and the integration points x_i are chosen as the zeros of the polynomial of degree $n-1$. Note carefully that $w_i \neq W(x_i)$. The weights w_i can in general be expressed as [172]

$$w_i = \left(\frac{a_n}{a_{n-1}}\right)\frac{\int_a^b W(x)p_{n-1}(x)^2\,\mathrm{d}x}{p_n'(x_i)p_{n-1}(x_i)} \tag{B.8}$$

where $p_n(x)$ is the orthogonal polynomial of degree n and a_n is the coefficient of the x^n term in $p_n(x)$. In some cases, the weight function is present in the original integral and the extraction constitutes a strict simplification of the function. For example, with Chebyshev polynomials[1] the weight function takes the form $W(x) =$

[1]The Chebyshev polynomials are solutions to the differential equation

$$(1-x^2)\frac{\partial^2 y(x)}{\partial x^2} - x\frac{\partial y(x)}{\partial x} + n^2 y(x) = 0, \tag{B.9}$$

$1/\sqrt{1-x^2}$, so trying to apply Gauss-Chebyshev quadrature to the integrand $(x^{10}+x+2)/\sqrt{1-x^2}$ would yield simply $g(x) = x^{10}+x+2$ and we would just have to evaluate g_i according to the zeroes of the nth Chebyshev polynomial. Each class of polynomials is associated with a specific interval of integration. For Chebyshev, this is $[-1,1]$. So using our previous example, we note that with only 3! integration points (exclamation point for emphasis *and* factorial function) we integrate *exactly*

$$I \equiv \int_{-1}^{1} \underbrace{\frac{x^{10}+x+2}{\sqrt{1-x^2}}}_{\equiv f(x)} \,\mathrm{d}x = \sum_{i=1}^{6} w_i \underbrace{(x^{10}+x+2)}_{g(x)} = \frac{575\pi}{256}. \tag{B.11}$$

In general, if $g(x)$ is a polynomial of degree $2n-1$ for a weight function associated with some class of orthogonal polynomials, then the gaussian quadrature rule associated with the same class of polynomials will integrate the original $f(x)$ (recall that $g(x) = f(x)/W(x)$) *exactly* with only n integration points [16].

Multiple integrals

Both of the aforementioned rules are straight forward to extend to higher dimensional integrals. For the Newton-Cotes rules, we can simply apply the rule again to the sum resulting from the application of the rule, i.e.

$$\begin{aligned} I_{\text{2D}} &= \int_a^b \int_a^b f(x,y)\,\mathrm{d}x\,\mathrm{d}y \approx \int_a^b \sum_{i=1}^{n} \Delta x f(\xi_i, y)\,\mathrm{d}y \\ &\approx \sum_{i=1}^{n}\sum_{j=1}^{n} \Delta x \Delta y f(\xi_i, \zeta_j). \end{aligned} \tag{B.12}$$

Since function evaluations on the endpoints of sub-intervals (sub-areas to be precise) coincide with the endpoints of the neighbouring sub-intervals, a number of points may be evaluated multiple times and thus have a higher *weight* in the final sum. For example, the 1D trapezoidal rule carries weights

$$1/2 \quad 1 \quad 1 \quad 1 \quad \ldots \quad 1 \quad 1 \quad 1 \quad 1/2, \tag{B.13}$$

with n a non-negative integer. In general, the solution can be written as [73]

$$T_n(x) = \sum_{k=0}^{\lfloor 1/2 \rfloor} \binom{n}{2k} (x^2-1) x^{n-2k}. \tag{B.10}$$

since

$$\begin{aligned}\frac{\Delta x}{2}\sum_{i=1}^{n}\Big(f(x_{i-1}+f(x_i)\Big) &= \frac{\Delta x}{2}\left[f(x_0)+2\sum_{i=1}^{n-1}\Big(f(x_i)+f(x_{i+1})\Big)\right]\\ &= \frac{\Delta x}{2}\left[f(x_0)+2\left(\sum_{i=1}^{n-1}f(x_i)\right)+f(x_n)\right]\\ &= \frac{\Delta x}{2}\left[f(x_0)+2f(x_1)+2f(x_2)+\cdots+2f(x_{n-2})+2f(x_{n-1})+f(x_n)\right].\end{aligned} \tag{B.14}$$

In a similar way, the 2D trapezoidal rule has the weights

$$\begin{array}{ccccccccc} 1/2 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1/2 \\ 1 & 2 & 2 & 2 & \dots & 2 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 & \dots & 2 & 2 & 2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 1 & 2 & 2 & 2 & \dots & 2 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 & \dots & 2 & 2 & 2 & 1 \\ 1/2 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1/2 \end{array}, \tag{B.15}$$

while the 2D Simpson's rule attains the weights (apart from a factor $1/6$)

$$\begin{array}{ccccccccccc} 1 & 4 & 2 & 4 & 2 & \dots & 2 & 4 & 2 & 4 & 1 \\ 4 & 16 & 8 & 16 & 8 & \dots & 8 & 16 & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & \dots & 4 & 8 & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & \dots & 8 & 16 & 8 & 16 & 4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 4 & 16 & 8 & 16 & 8 & \dots & 8 & 16 & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & \dots & 4 & 8 & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & \dots & 8 & 16 & 8 & 16 & 4 \\ 1 & 1 & 2 & 4 & 2 & \dots & 2 & 4 & 2 & 4 & 1 \end{array}. \tag{B.16}$$

A similar scheme yields multi-dimensional Gaussian quadrature rules, where the total weights become products of the 1D weights.

Appendix C

Functionals and functional variations

Recall that a *function* is a mapping from some algebraic scalar field $\mathbb{F}$ to another (possibly different) field $\mathbb{F}'$, i.e. $g : \mathbb{F} \to \mathbb{F}'$. In physics, we are usually interested mainly in the cases where $\mathbb{F}$ and $\mathbb{F}'$ are the real or complex numbers, $\mathbb{R}$ or $\mathbb{C}$. An example is the complex exponential, $x \mapsto e^{ix}$, which takes complex values but the argument is real, so $g : \mathbb{R} \to \mathbb{C}$ in this case.

A *functional*, on the other hand, is a mapping from some function space, $\mathcal{F}$, to a scalar field $\mathbb{F}$, i.e. $f : \mathcal{F} \to F$. We will take the space of functions to be the underlying Hilbert space of our quantum mechanical system, $\mathcal{H}$, and the field to be the complex numbers, C. The functional thus assigns to each $f \in \mathcal{H}$ a complex number.

As a familiar example of such a construction, let us consider the definite integral. For the moment, let us take the function space to be continous real functions of a single real arguments in the range $[0, 1]$, $C([0, 1])$. We call this *functional* I, such that $I : C([0, 1]) \to \mathbb{R}$.

$$I[f] = \int_0^1 \mathrm{d}x f(x) \tag{C.1}$$

thus assigns a real number to any continous function on $[0, 1]$. For example, $I[e^x] = e - 1 \approx 1.7183$ or $I[\sqrt{x}] = 2/3 \approx 0.6667$.

When working in a separable Hilbert space as we always do in quantum mechanics, we may always express any function $f \in \mathcal{H}$ in terms of some basis $\{ chi_n \}_{n=1}^{\infty}$, (recall the Parseval relation from section ((QM math)))

$$|f\rangle = \left(\sum_{n=1}^{\infty} |\chi_n\rangle\langle\chi_n| \right) |f\rangle = \sum_{n=1}^{\infty} \underbrace{\langle\chi_n|f\rangle}_{c_n} |\chi_n\rangle = \sum_{n=1}^{\infty} c_n |\chi_n\rangle, \tag{C.2}$$

meaning we can think of a functional $F[f]$ as a *function* of the vector of coefficients relative to the basis set, $\mathbf{c} = (c_1, c_2, \dots)$ [14].

Short mathematical interlude

Let $B(X, Y)$ denote the set of all continous linear transformations from normed vector spaces X and Y (over the algebraic scalar field $\mathbb{F}$[1]). For example we may consider $X = \mathbb{R}^n$ and $Y = \mathbb{R}^m$, i.e. the set of real vectors of length n and m, respectively. The set of continous linear transformations from X to Y, $B(X, Y)$, thus consists of real valued matrices of dimensions $m \times n$, so we may write $B(\mathbb{R}^n, \mathbb{R}^m) = \mathbb{R}^{m\times n}$.

A *Banach space* is a normed vector space which is complete under the metric associated with the norm. Since any norm, $\|\cdot\|$ induces as metric by $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$, and the inner product $\langle\cdot|\cdot\rangle$ induces a norm by $\|\cdot\| = \sqrt{\langle\cdot|\cdot\rangle}$ we can define a *Hilbert space* as a Banach space which is complete w.r.t. this specific metric [133, 170].

The space of linear transformations from X to $\mathbb{F}$, with X being some normed vector space, is called the *dual space* of X, sometimes denoted X^*. We note that a linear transformation from X to $\mathbb{F}$ is exactly a linear *functional*, and so functionals "live in the dual" of the vector space itself. It turns out that if X is normed, the dual is always a Banach space [42]. Since we are inherently working with Hilbert spaces in quantum mechanics, it is natural to ask: What can we say in general about the vector space of functionals on a Hilbert space $\mathcal{H}$?

In the following, we take $f \in \mathcal{H}^*$ to be a linear functional on $\mathcal{H}$ and $x \in \mathcal{H}$ to be a function in the Hilbert space itself. It can be shown that for any such x there exists a *unique* $y \in \mathcal{H}$ such that $f[x] = f_y[x] = \langle x|y\rangle$, where the functional $f_y[\cdot] \equiv \langle\cdot|y\rangle$ [42, 133]. This is known as the Riesz representation theorem or sometimes the Riesz-Fréchet theorem. Essentially, this means that we can associate the dual space of $\mathcal{H}$ with the space itself since there is a correspondance between the functionals and the elements of the space itself. This is more succintly stated as Hilbert spaces are *self-dual*, and it is this property that justifies the use of Dirac bra-ket notation since we are guaranteed that any ket has a unique corresponding bra which is its Hermitian conjugate.

Functional differentials and derivatives

The differential of a functional $F[f]$ is the part of the difference $F[f + \delta f] - F[f]$ that depends linearly on δf, where δf is an infinitesimal variation of the argument function f [60]. Since we need to account for the continous variation of F over the infinitesimal range $[f, f + \delta f]$ we take the integral

$$\delta F[f] = \int \frac{\delta F[f]}{\delta f(x)} \delta f(x)\, \mathrm{d}x, \tag{C.3}$$

[1]Meaning X and Y are closed under scalar multiplication with elements $c \in \mathbb{F}$.

where we have defined the *functional derivative* of F w.r.t. f at the point x as

$$F'[f] \equiv \frac{\delta F[f]}{\delta f(x)}. \tag{C.4}$$

If the underlying space is a Banach space, meaning the dual space is also a Banach space (c.f. section C), we can write the functional differential in a way that is familiar [173]:

$$\delta F[f] = \lim_{\varepsilon \to 0} \frac{F[f + \varepsilon \delta f(x)] - F[f]}{\varepsilon} = \int \frac{\delta F[f]}{\delta f(x)} \delta f(x)\, \mathrm{d}x. \tag{C.5}$$

In the following, assume $g[f]$ is a functional of the function f. It turns out that the functional derivative behaves a lot like ordinary derivatives, [100]

$$\begin{aligned}
\frac{\delta}{\delta f(x)}\Big(aF[f] + bG[f]\Big) &= a\frac{\delta F[f]}{\delta f(x)} + b\frac{\delta G[f]}{\delta f(x)} && \text{(linearity)} \\
\frac{\delta}{\delta f(x)}\Big(F[f]\, G[f]\Big) &= G[f]\frac{\delta F[f]}{\delta f(x)} + F[f]\frac{\delta G[f]}{\delta f(x)} && \text{(product rule)} \\
\frac{\delta}{\delta f(x)}\Big(F[g]\Big) &= \int \frac{\delta F[g]}{\delta g(x')}\frac{\delta g(x')}{\delta f(x)}\, \mathrm{d}x && \text{(chain rule)}
\end{aligned}$$

We can also define higher-order functional derivaties, for example the equivalent of the ordinary double derivative

$$\frac{\delta^2 F[f]}{\delta f(x)\delta f(x')} = \frac{\delta}{\delta f(x)}\left(\frac{\delta F[f]}{\delta f(x')}\right). \tag{C.6}$$

We may use this to compute the Taylor expansion of a functional $F[f]$ as

$$\begin{aligned}
F[f + \Delta f] &= F[f] + \sum_{n=1}^{\infty} \frac{1}{n!} \int \cdots \int \frac{\delta^{(n)} F[f]}{\delta f(x_1) \dots \delta f(x_n)} \Delta f(x_1)\Delta f(x_2) \dots \Delta f(x_n) \mathrm{d}x_1 \mathrm{d}x_2 \dots \mathrm{d}x_n \\
F[f + \Delta f] &= F[f] + \int \frac{\delta F[f]}{\delta f(x)} \Delta f(x)\, \mathrm{d}x + \frac{1}{2} \int\int \frac{\delta^2 F[f]}{\delta f(x)\delta f(x')} \Delta f(x)\Delta f(x')\, \mathrm{d}x\, \mathrm{d}x' + \dots,
\end{aligned} \tag{C.7}$$

where $\Delta f(x)$ is a finite (not infinitesimal) variation in the function $f(x)$ [60].

Bibliography

[1] John-Anders Stende. "Constructing high-dimensional neural network potentials for molecular dynamics". MA thesis. University of Oslo, 2017.

[2] Håkon Treider. "Speeding up ab-initio molecular dynamics with artificial neural networks". MA thesis. University of Oslo, 2017.

[3] Svenn-Arne Dragly. "Bridging Quantum Mechanics and Molecular Dynamics with Artificial Neural Networks". MA thesis. University of Oslo, 2014.

[4] Wei Hu, Lin Lin, and Chao Yang. "DGDFT: A massively parallel method for large scale density functional theory calculations". In: *The Journal of Chemical Physics* 143.12 (2015), p. 124110. DOI: 10.1063/1.4931732. eprint: https://doi.org/10.1063/1.4931732. URL: https://doi.org/10.1063/1.4931732.

[5] D.R. Bowler and T. Miyazaki. "Calculations for millions of atoms with density functional theory: linear scaling shows its potential". In: *Journal of Physics: Condensed Matter* 22.7 (2010), p. 074207. URL: http://stacks.iop.org/0953-8984/22/i=7/a=074207.

[6] Joost VandeVondele, Urban Borštnik, and Jürg Hutter. "Linear Scaling Self-Consistent Field Calculations with Millions of Atoms in the Condensed Phase". In: *Journal of Chemical Theory and Computation* 8.10 (2012). PMID: 26593003, pp. 3565–3573. DOI: 10.1021/ct200897x. eprint: http://dx.doi.org/10.1021/ct200897x. URL: http://dx.doi.org/10.1021/ct200897x.

[7] Gongpu Zhao et al. "Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics". In: *Nature* 497 (May 2013). URL: http://dx.doi.org/10.1038/nature12162.

[8] Tyler Reddy et al. "Nothing to Sneeze At: A Dynamic and Integrative Computational Model of an Influenza A Virion". In: *Structure* 23.3 (2015), pp. 584–597. ISSN: 0969-2126. DOI: https://doi.org/10.1016/j.str.2014.12.019. URL: https://www.sciencedirect.com/science/article/pii/S0969212615000325.

[9] TIMOTHY C. GERMANN and KAI KADAU. "TRILLION-ATOM MOLECULAR DYNAMICS BECOMES A REALITY". In: *International Journal of Modern Physics C* 19.09 (2008), pp. 1315–1319. DOI: 10.1142/S0129183108012911. eprint: http://www.worldscientific.com/doi/pdf/10.1142/S0129183108012911. URL: http://www.worldscientific.com/doi/abs/10.1142/S0129183108012911.

[10] R. E. Angulo et al. "Scaling relations for galaxy clusters in the Millennium-XXL simulation". In: *Monthly Notices of the Royal Astronomical Society* 426.3 (2012), pp. 2046–2062. DOI: 10.1111/j.1365-2966.2012.21830.x. eprint: /oup/backfile/content_public/journal/mnras/426/3/10.1111/j.1365-2966.2012.21830.x/2/426-3-2046.pdf. URL: +%20http://dx.doi.org/10.1111/j.1365-2966.2012.21830.x.

[11] Juhan Kim et al. "The new horizon run cosmological N-body simulations". In: *arXiv preprint arXiv:1112.1754* (2011).

[12] E. Weinan. *Principles of Multiscale Modeling*. Cambridge University Press, 2011. ISBN: 1-107-09654-5.

[13] T. Helgaker, P. Jørgensen, and J. Olsen. *Molecular Electronic-Structure Theory*. John Wiley & Sons, 2000. ISBN: 0-471-96755-6.

[14] S. Kvaal. *Lecture notes for FYS-KJM4480*. Lecture notes. Sept. 2017. URL: http://theory.rutgers.edu/~giese/notes/DFT.pdf.

[15] I. Shavitt and R.J. Bartlett. *Many-Body Methods in Chemistry and Physics. MBPT and Coupled-Cluster Theory*. Cambridge University Press, 2009. ISBN: 0-521-81832-X.

[16] M. Hjorth-Jensen. *Computational Physics*. Lecture notes. Aug. 2015. URL: https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf.

[17] B.L. Hammond, W.A. Lester Jr., and P.J. Reynolds. *Monte Carlo Methods in Ab Initio Quantum Chemistry*. Wspc, 1994. ISBN: 981-02-0322-5.

[18] Harold Conroy. "Molecular Schrödinger Equation. II. Monte Carlo Evaluation of Integrals". In: *The Journal of Chemical Physics* 41.5 (1964), pp. 1331–1335. DOI: 10.1063/1.1726069. eprint: https://doi.org/10.1063/1.1726069. URL: https://doi.org/10.1063/1.1726069.

[19] J.B. Anderson. *Quantum Monte Carlo. Origins, Development, Applications*. Oxford University Press, 2007. ISBN: 0-19-531010-1.

[20] D. R. Hartree. "The Wave Mechanics of an Atom with a Non-Coulomb Central Field. Part I. Theory and Methods". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 24.1 (1928), pp. 89–110. DOI: 10.1017/S0305004100011919.

[21] V. Fock. "Näherungsmethode zur Lösung des quantenmechanischen Mehrkörperproblems". In: *Zeitschrift für Physik* 61.1 (Jan. 1930), pp. 126–148. ISSN: 0044-3328. DOI: 10.1007/BF01340294. URL: https://doi.org/10.1007/BF01340294.

[22] A. Szabo and N.S. Ostlund. *Modern Quantum Chemistry*. Dover Publications, 1996. ISBN: 0-486-69186-1.

[23] W. Kohn and L. J. Sham. "Self-Consistent Equations Including Exchange and Correlation Effects". In: *Phys. Rev.* 140 (4A Nov. 1965), A1133–A1138. DOI: 10.1103/PhysRev.140.A1133. URL: https://link.aps.org/doi/10.1103/PhysRev.140.A1133.

[24] R.M. Martin. *Electronic Structure*. Cambridge University Press, 2004. ISBN: 0-521-53440-2.

[25] Axel D. Becke. "Perspective: Fifty years of density-functional theory in chemical physics". In: *The Journal of Chemical Physics* 140.18 (2014), 18A301. DOI: 10.1063/1.4869598. eprint: https://doi.org/10.1063/1.4869598. URL: https://doi.org/10.1063/1.4869598.

[26] Laura E. Ratcliff et al. "Challenges in large scale quantum mechanical calculations". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 7.1 (2017). e1290, e1290–n/a. ISSN: 1759-0884. DOI: 10.1002/wcms.1290. URL: http://dx.doi.org/10.1002/wcms.1290.

[27] R. Car and M. Parrinello. "Unified Approach for Molecular Dynamics and Density-Functional Theory". In: *Phys. Rev. Lett.* 55 (22 Nov. 1985), pp. 2471–2474. DOI: 10.1103/PhysRevLett.55.2471. URL: https://link.aps.org/doi/10.1103/PhysRevLett.55.2471.

[28] Renata M. Wentzcovitch and JoséLuís Martins. "First principles molecular dynamics of Li: Test of a new algorithm". In: *Solid State Communications* 78.9 (1991), pp. 831–834. ISSN: 0038-1098. DOI: https://doi.org/10.1016/0038-1098(91)90629-A. URL: http://www.sciencedirect.com/science/article/pii/003810989190629A.

[29] R. N. Barnett et al. "Born–Oppenheimer dynamics using density-functional theory: Equilibrium and fragmentation of small sodium clusters". In: *The Journal of Chemical Physics* 94.1 (1991), pp. 608–616. DOI: 10.1063/1.460327. eprint: https://doi.org/10.1063/1.460327. URL: https://doi.org/10.1063/1.460327.

[30] Moshe Y Vardi. "Artificial intelligence: past and future". In: *Communications of the ACM* 55.1 (2012), pp. 5–5.

[31] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260. ISSN: 0036-8075. DOI: 10.1126/science.aaa8415. eprint: http://science.sciencemag.org/content/349/6245/255.full.pdf. URL: http://science.sciencemag.org/content/349/6245/255.

[32] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. eprint: arXiv:1712.01815.

[33] Tord Romstad, Marco Costalba, and Joona Kiiski. *Stockfish Chess*. https://stockfishchess.org/. 2008–2017.

[34] Jörg Behler and Michele Parrinello. "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces". In: *Phys. Rev. Lett.* 98 (14 Apr. 2007), p. 146401. DOI: 10.1103/PhysRevLett.98.146401. URL: https://link.aps.org/doi/10.1103/PhysRevLett.98.146401.

[35] Lin Shen, Jingheng Wu, and Weitao Yang. "Multiscale Quantum Mechanics/Molecular Mechanics Simulations with Neural Networks". In: *Journal of Chemical Theory and Computation* 12.10 (2016). PMID: 27552235, pp. 4934–4946. DOI: 10.1021/acs.jctc.6b00663. eprint: http://dx.doi.org/10.1021/acs.jctc.6b00663. URL: http://dx.doi.org/10.1021/acs.jctc.6b00663.

[36] Jörg Behler. "Perspective: Machine learning potentials for atomistic simulations". In: *The Journal of Chemical Physics* 145.17 (2016), p. 170901. DOI: 10.1063/1.4966192. eprint: https://doi.org/10.1063/1.4966192. URL: https://doi.org/10.1063/1.4966192.

[37] A. Danial. *cloc*. https://github.com/AlDanial/cloc. 2017.

[38] *Quantities and units – Part 2: Mathematical signs and symbols to be used in the natural sciences and technology*. Standard. Geneva, CH: International Organization for Standardization, Dec. 2009.

[39] H. Goldstein, C.P. Poole Jr., and J.L. Safko. *Classical Mechanics*. 3rd ed. Pearson, 2001. ISBN: 0-201-65702-3.

[40] H. Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, 2010. ISBN: 0-387-70913-4.

[41] J.M. Leinaas. *Modern Quantum Mechanics*. Lecture notes. Aug. 2016. URL: http://www.uio.no/studier/emner/matnat/fys/FYS4110/h17/undervisningsmateriale/lecturenotes2016.pdf.

[42] B.P. Rynne and M.A. Youngson. *Linear Functional Analysis*. 2nd ed. Springer-Verlag London, 2008. ISBN: 1-84800-004-9.

[43] S. Hassani. *Mathematical Physics*. Springer-Verlag, 1999. ISBN: 0-387-98579-4.

[44] R. Shankar. *Principles of Quantum Mechanics*. 2nd ed. Plenum Press, 2011. ISBN: 0-306-44790-8.

[45] J.J. Sakurai. *Modern Quantum Mechanics*. revised. Addison Wesley, 1993. ISBN: 0-201-53929-2.

[46] D.J. Griffiths. *Introduction to Quantum Mechanics*. 2nd ed. Pearson Prentice Hall, 2005. ISBN: 0-13-191175-9.

[47] G.D. Mahan. *Quantum Mechanics in a Nutshell*. Princeton University Press, 2009. ISBN: 0-691-13713-7.

[48] C. Itzykson and J.B. Zuber. *Quantum Field Theory*. McGraw-Hill Book Company, 1980. ISBN: 0-07-032071-3.

[49] Theodore A. Welton. "Some Observable Effects of the Quantum-Mechanical Fluctuations of the Electromagnetic Field". In: *Phys. Rev.* 74 (9 Nov. 1948), pp. 1157–1167. DOI: 10.1103/PhysRev.74.1157. URL: https://link.aps.org/doi/10.1103/PhysRev.74.1157.

[50] U. D. Jentschura et al. "Long-range interactions of hydrogen atoms in excited states. II. Hyperfine-resolved $2S-2S$ systems". In: *Phys. Rev. A* 95 (2 Feb. 2017), p. 022704. DOI: 10.1103/PhysRevA.95.022704. URL: https://link.aps.org/doi/10.1103/PhysRevA.95.022704.

[51] Paul Adrien Maurice Dirac. "Quantum mechanics of many-electron systems". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 123. 792. The Royal Society. 1929, pp. 714–733.

[52] A Bolotin. "Any realistic model of a physical system must be computationally realistic". In: *Journal of Physics: Conference Series* 574.1 (2015), p. 012088. URL: http://stacks.iop.org/1742-6596/574/i=1/a=012088.

[53] D.V. Schroeder. *An Introduction to Thermal Physics*. Pearson, 2005. ISBN: 0-201-38027-7.

[54] M. Born and R. Oppenheimer. "Zur Quantentheorie der Molekeln". In: *Annalen der Physik* 389.20 (1927), pp. 457–484. ISSN: 1521-3889. DOI: 10.1002/andp.19273892002. URL: http://dx.doi.org/10.1002/andp.19273892002.

[55] S. Weinberg. *Lectures on Quantum Mechanics*. 2nd ed. Cambridge University Press, 2015. ISBN: 1-107-11166-8.

[56] J. C. Slater. "The Theory of Complex Spectra". In: *Phys. Rev.* 34 (10 Nov. 1929), pp. 1293–1322. DOI: 10.1103/PhysRev.34.1293. URL: https://link.aps.org/doi/10.1103/PhysRev.34.1293.

[57] J. Thijssen. *Computational Physics*. 2nd ed. Cambridge University Press, 2007. ISBN: 1-107-67713-0.

[58] D.C. Lay. *Linear Algebra and Its Applications*. 4th ed. Pearson Education, 2012. ISBN: 0-321-62335-5.

[59] L. Salasnich. *Quantum Physics of Light and Matter. A Modern Introduction to Photons, Atoms and Many-Body Systems*. Springer, 2016. ISBN: 3-319-38271-3.

[60] R.G. Parr and W. Yang. *Density-functional theory of atoms and molecules*. Oxford University Press, 1989. ISBN: 0-19-504279-4.

[61] E.S. Kryachko and E.V. Ludeña. *Energy Density Functional Theory of Many-Electron Systems*. Kluwer Academic Publishers, 1990. ISBN: 0-7923-0641-4.

[62] Max Born. "Quantenmechanik der Stoßvorgänge". In: *Zeitschrift für Physik* 38.11 (Nov. 1926), pp. 803–827. ISSN: 0044-3328. DOI: 10.1007/BF01397184. URL: https://doi.org/10.1007/BF01397184.

[63] J.A. Wheeler and W.H. Zurek. *Quantum Theory and Measurement*. Princeton University Press, 1983. ISBN: 0-691-61316-8.

[64] E.K.U. Gross, E.Runge, and O. Heinonen. *Many-Particle Theory*. Adam Hilger, 1991. ISBN: 0-7503-0155-4.

[65] J. Katriel and E. R. Davidson. "Asymptotic behavior of atomic and molecular wave functions". In: *Proceedings of the National Academy of Sciences* 77.8 (1980), pp. 4403–4406. eprint: http://www.pnas.org/content/77/8/4403.full.pdf. URL: http://www.pnas.org/content/77/8/4403.abstract.

[66] M. Weissbluth. *Atoms and Molecules*. Academic Press, 1978. ISBN: 0-12-744450-5.

[67] Tosio Kato. "On the eigenfunctions of many-particle systems in quantum mechanics". In: *Communications on Pure and Applied Mathematics* 10.2 (1957), pp. 151–177. ISSN: 1097-0312. DOI: 10.1002/cpa.3160100201. URL: http://dx.doi.org/10.1002/cpa.3160100201.

[68] M. Hjorth-Jensen. *Computational Physics*. Lecture notes. Aug. 2015. URL: https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf.

[69] J. Toulouse, R. Assaraf, and C.J. Umrigar. "Introduction to the Variational and Diffusion Monte Carlo Methods". In: *Advances in Quantum Chemistry, Volume 73*. Ed. by P.E. Hoggan and T. Ozdogan. Elsevier, 2016, pp. 285–314. ISBN: 978-0-12-803060-8.

[70] A. Bijl. "The lowest wave function of the symmetrical many particles system". In: *Physica* 7.9 (1940), pp. 869–886. ISSN: 0031-8914. DOI: https://doi.org/10.1016/0031-8914(40)90166-5. URL: http://www.sciencedirect.com/science/article/pii/0031891440901665.

[71] Robert Jastrow. "Many-Body Problem with Strong Forces". In: *Phys. Rev.* 98 (5 June 1955), pp. 1479–1484. DOI: 10.1103/PhysRev.98.1479. URL: https://link.aps.org/doi/10.1103/PhysRev.98.1479.

[72] K. Atkinson and W. Han. *Spherical Harmonics and Approximations on the Unit Sphere: An Introduction*. Springer-Verlag Berlin Heidelberg, 2012. ISBN: 3-642-25982-0.

[73] K. Rottmann. *Matematisk Formelsamling*. 10th ed. Spektrum Forlag, 2008. ISBN: 8278220050.

[74] J. C. Slater. "Atomic Shielding Constants". In: *Phys. Rev.* 36 (1 July 1930), pp. 57–64. DOI: 10.1103/PhysRev.36.57. URL: https://link.aps.org/doi/10.1103/PhysRev.36.57.

[75] Clarence Zener. "Analytic Atomic Wave Functions". In: *Phys. Rev.* 36 (1 July 1930), pp. 51–56. DOI: 10.1103/PhysRev.36.51. URL: https://link.aps.org/doi/10.1103/PhysRev.36.51.

[76] C.J. Cramer. *Essentials of Computational Chemistry. Theories and Models*. 2nd ed. Wiley, 2004. ISBN: 0-470-09182-7.

[77] Bruno Klahn and Werner A. Bingel. "The convergence of the Rayleigh-Ritz Method in quantum chemistry". In: *Theoretica chimica acta* 44.1 (Mar. 1977), pp. 27–43. ISSN: 1432-2234. DOI: 10.1007/BF00548027. URL: https://doi.org/10.1007/BF00548027.

[78] J. Fernández Rico et al. "Efficiency of the algorithms for the calculation of Slater molecular integrals in polyatomic molecules". In: *Journal of Computational Chemistry* 25.16 (2004), pp. 1987–1994. ISSN: 1096-987X. DOI: 10.1002/jcc.20131. URL: http://dx.doi.org/10.1002/jcc.20131.

[79] I. Ema et al. "Polarized basis sets of Slater-type orbitals: H to Ne atoms". In: *Journal of Computational Chemistry* 24.7 (2003), pp. 859–868. ISSN: 1096-987X. DOI: 10.1002/jcc.10227. URL: http://dx.doi.org/10.1002/jcc.10227.

[80] S Francis Boys. "Electronic wave functions. I. A general method of calculation for the stationary states of any molecular system". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 200. 1063. The Royal Society. 1950, pp. 542–554.

[81] H.P. Langtangen. *Numerical Methods for Partial Differential Equations*. Lecture notes. 2016. URL: http://hplgit.github.io/num-methods-for-PDEs/doc/pub/index.html.

[82] H. Hochstadt. *The Functions of Mathematical Physics*. revised. Dover Publications, 2012. ISBN: 0-486-65214-9.

[83] T. Helgaker and P.R. Taylor. "Gaussian basis sets and molecular integrals". In: *Modern Electronic Structure Theory, Part II*. Ed. by David Yarkony. World Scientific, 1995, pp. 725–856. ISBN: 981-02-2988-7. URL: %5Curl%7Bhttp://folk.uio.no/helgaker/reprints/ModElectStructTheory_GaussianBasisIntegrals_1995.pdf%7D.

[84] T. Helgaker. "Molecular Integral Evaluation". The 11th Sostrup Summer School: Quantum Chemistry and Molecular Properties. 2010. URL: %5Curl%7Bhttp://folk.uio.no/helgaker/talks/SostrupIntegrals_10.pdf%7D.

[85] Larry E McMurchie and Ernest R Davidson. "One- and two-electron integrals over cartesian gaussian functions". In: *Journal of Computational Physics* 26.2 (1978), pp. 218–231. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(78)90092-X. URL: http://www.sciencedirect.com/science/article/pii/002199917890092X.

[86] W. J. Hehre, R. F. Stewart, and J. A. Pople. "Self-Consistent Molecular-Orbital Methods. I. Use of Gaussian Expansions of Slater-Type Atomic Orbitals". In: *The Journal of Chemical Physics* 51.6 (1969), pp. 2657–2664. DOI: 10.1063/1.1672392. eprint: https://doi.org/10.1063/1.1672392. URL: https://doi.org/10.1063/1.1672392.

[87] Abraham Charnes, William W Cooper, and Robert O Ferguson. "Optimal estimation of executive compensation by linear programming". In: *Management science* 1.2 (1955), pp. 138–151.

[88] Roger Koenker and Gilbert Bassett Jr. "Regression quantiles". In: *Econometrica: journal of the Econometric Society* (1978), pp. 33–50.

[89] Jorge J Moré and Danny C Sorensen. "Computing a trust region step". In: *SIAM Journal on Scientific and Statistical Computing* 4.3 (1983), pp. 553–572.

[90] S.S. Zumdahl. *Chemical Principles*. 6th ed. Brooks-Cole Publishing, 2009. ISBN: 0-538-73456-6.

[91] Warren J Hehre, Robert Ditchfield, and John A Pople. "Self—consistent molecular orbital methods. XII. Further extensions of gaussian—type basis sets for use in molecular orbital studies of organic molecules". In: *The Journal of Chemical Physics* 56.5 (1972), pp. 2257–2261.

[92] James D Dill and John A Pople. "Self-consistent molecular orbital methods. XV. Extended Gaussian-type basis sets for lithium, beryllium, and boron". In: *The Journal of Chemical Physics* 62.7 (1975), pp. 2921–2923.

[93] 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.

[94] I.N. Levine. *Quantum Chemistry*. 7th ed. Pearson, 2014. ISBN: 0-321-80345-0.

[95] C. C. J. Roothaan. "New Developments in Molecular Orbital Theory". In: *Rev. Mod. Phys.* 23 (2 Apr. 1951), pp. 69–89. DOI: 10.1103/RevModPhys.23.69. URL: https://link.aps.org/doi/10.1103/RevModPhys.23.69.

[96] "The molecular orbital theory of chemical valency VIII. A method of calculating ionization potentials". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 205.1083 (1951), pp. 541–552. ISSN: 0080-4630. DOI: 10.1098/rspa.1951.0048. eprint: http://rspa.royalsocietypublishing.org/content/205/1083/541.full.pdf. URL: http://rspa.royalsocietypublishing.org/content/205/1083/541.

[97] J. A. Pople and R. K. Nesbet. "Self-Consistent Orbitals for Radicals". In: *The Journal of Chemical Physics* 22.3 (1954), pp. 571–572. DOI: 10.1063/1.1740120. eprint: http://dx.doi.org/10.1063/1.1740120. URL: http://dx.doi.org/10.1063/1.1740120.

[98] Werner Kutzelnigg and John D. Morgan III. "Rates of convergence of the partial-wave expansions of atomic correlation energies". In: *The Journal of Chemical Physics* 96.6 (1992), pp. 4484–4508. DOI: 10.1063/1.462811. eprint: https://doi.org/10.1063/1.462811. URL: https://doi.org/10.1063/1.462811.

[99] Frank Jensen. "Estimating the Hartree—Fock limit from finite basis set calculations". In: *Theoretical Chemistry Accounts* 113.5 (June 2005), pp. 267–273. ISSN: 1432-2234. DOI: 10.1007/s00214-005-0635-2. URL: https://doi.org/10.1007/s00214-005-0635-2.

[100] J. Toulouse. *Introduction to density-functional theory*. Lecture notes. June 2017. URL: http://www.lct.jussieu.fr/pagesperso/toulouse/enseignement/introduction_dft.pdf.

[101] P. Hohenberg and W. Kohn. "Inhomogeneous Electron Gas". In: *Phys. Rev.* 136 (3B Nov. 1964), B864–B871. DOI: 10.1103/PhysRev.136.B864. URL: https://link.aps.org/doi/10.1103/PhysRev.136.B864.

[102] B.O. Roos, ed. *Lecture Notes in Quantum Chemistry II*. Springer-Verlag Berlin Heidelberg, 1994. ISBN: 3-540-58620-2.

[103] E. Engel and R.M. Dreizler. *Density Functional Theory*. Springer-Verlag Berlin Heidelberg, 2011. ISBN: 3-642-14089-0.

[104] Paul AM Dirac. "Note on exchange phenomena in the Thomas atom". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 26. 3. Cambridge University Press. 1930, pp. 376–385.

[105] David M Ceperley and BJ Alder. "Ground state of the electron gas by a stochastic method". In: *Physical Review Letters* 45.7 (1980), p. 566.

[106] S. H. Vosko, L. Wilk, and M. Nusair. "Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis". In: *Canadian Journal of Physics* 58.8 (1980), pp. 1200–1211. DOI: 10.1139/p80-159. eprint: https://doi.org/10.1139/p80-159. URL: https://doi.org/10.1139/p80-159.

[107] P. M. Boerrigter, G. Te Velde, and J. E. Baerends. "Three-dimensional numerical integration for electronic structure calculations". In: *International Journal of Quantum Chemistry* 33.2 (1988), pp. 87–113. ISSN: 1097-461X. DOI: 10.1002/qua.560330204. URL: http://dx.doi.org/10.1002/qua.560330204.

[108] C.H.L. Beentjes. *Quadrature on a Spherical Surface*. Lecture notes. 2015. URL: http://people.maths.ox.ac.uk/beentjes/Essays/QuadratureSphere.pdf.

[109] A. D. McLaren. "Optimal Numerical Integration on a Sphere". In: *Mathematics of Computation* 17.84 (1963), pp. 361–383. ISSN: 00255718, 10886842. URL: http://www.jstor.org/stable/2003998.

[110] S.L. Sobolev. *Selected Works of S.L. Sobolev*. Vol. 1. Springer, 2006. ISBN: 0-387-34148-X.

[111] V. I. Lebedev. "Values of the nodes and weights of quadrature formulas of Gauss–Markov type for a sphere from the ninth to seventeenth order of accuracy that are invariant with respect to an octahedron group with inversion". In: *USSR Computational Mathematics and Mathematical Physics* 15.1 (1975), pp. 48–54.

[112] J.R. Sack and J. Urrutia. *Handbook of Computational Chemistry*. North Holland, 1999. ISBN: 0-444-82537-1.

[113] G. te Velde and E.J. Baerends. "Numerical integration for polyatomic systems". In: *Journal of Computational Physics* 99.1 (1992), pp. 84–98. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(92)90277-6. URL: http://www.sciencedirect.com/science/article/pii/0021999192902776.

[114] A. D. Becke. "A multicenter numerical integration scheme for polyatomic molecules". In: *The Journal of Chemical Physics* 88.4 (1988), pp. 2547–2553. DOI: 10.1063/1.454033. eprint: http://dx.doi.org/10.1063/1.454033. URL: http://dx.doi.org/10.1063/1.454033.

[115] M. Kuczma, B. Choczewski, and R. Ger. *Iterative Functional Equations*. Cambridge University Press, 1990. ISBN: 0-521-35561-3.

[116] José M. Pérez-Jordá, Axel D. Becke, and Emilio San-Fabián. "Automatic numerical integration techniques for polyatomic molecules". In: *The Journal of Chemical Physics* 100.9 (1994), pp. 6520–6534. DOI: 10.1063/1.467061. eprint: https://doi.org/10.1063/1.467061. URL: https://doi.org/10.1063/1.467061.

[117] Dario Bressanini and Peter Reynolds. "Between Classical and Quantum Monte Carlo Methods: "Variational" QMC". In: *Advances in Chemical Physics*. Vol. 105. Mar. 2007, pp. 37–64. ISBN: 0-470-14164-6.

[118] Nicholas Metropolis et al. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114. eprint: https://doi.org/10.1063/1.1699114. URL: https://doi.org/10.1063/1.1699114.

[119] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* (1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97. eprint: /oup/backfile/content_public/journal/biomet/57/10.1093/biomet/57.1.97/3/57-1-97.pdf. URL: +%20http://dx.doi.org/10.1093/biomet/57.1.97.

[120] W.H. Press et al. *Numerical Recipes. The Art of Scientific Computing*. 3rd ed. Cambridge University Press, 2007. ISBN: 0-521-88068-8.

[121] W.R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1995. ISBN: 0-412-05551-1.

[122] C.W. Gardiner. *Handbook of Stochastic Methods. for Physics, Chemistry and the Natural Sciences*. 3rd ed. Springer-Verlag, 2004. ISBN: 3-540-20882-8.

[123] M. Chaichian and A. Demichev. *Path Integrals in Physics. Stochastic Processes and Quantum Mechanics*. Vol. 1. CRC Press, 2001. ISBN: 0-7503-0801-X.

[124] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods*. Second, Revised and Enlarged. Wiley-VCH, 2008. ISBN: 3-527-40760-X.

[125] W. W. Wood and F. R. Parker. "Monte Carlo Equation of State of Molecules Interacting with the Lennard-Jones Potential. I. A Supercritical Isotherm at about Twice the Critical Temperature". In: *The Journal of Chemical Physics* 27.3 (1957), pp. 720–733. DOI: 10.1063/1.1743822. eprint: https://doi.org/10.1063/1.1743822. URL: https://doi.org/10.1063/1.1743822.

[126] E.H. Hauge. *Kinetisk Teori. Transportteori*. Lecture notes. 1970.

[127] N.G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. 3rd ed. North Holland, 2007. ISBN: 0-444-52965-9.

[128] E. Strohmaier et al. *Top 500: November 2017*. 2017. URL: %5Curl%7Bhttps://www.top500.org/lists/2017/11/%7D (visited on 11/15/2017).

[129] J.L. Devore and K.N. Berk. *Modern Mathematical Statistics with Applications*. Brooks/Cole, 2006. ISBN: 0-495-11022-1.

[130] H. Flyvbjerg and H. G. Petersen. "Error estimates on averages of correlated data". In: *The Journal of Chemical Physics* 91.1 (1989), pp. 461–466. DOI: 10.1063/1.457480. eprint: https://doi.org/10.1063/1.457480. URL: https://doi.org/10.1063/1.457480.

[131] L.M. Raff et al. *Neural networks in chemical reaction dynamics*. Oxford University Press, 2012. ISBN: 0-19-976565-0.

[132] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: http://www.sciencedirect.com/science/article/pii/0893608089900208.

[133] J.M. McDonald and N.A. Weiss. *A course in Real Analysis*. 2nd ed. Academic Press, 2013. ISBN: 0-12-387774-1.

[134] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (Oct. 1986), URL: http://dx.doi.org/10.1038/323533a0.

[135] JONATHAN BARZILAI and JONATHAN M. BORWEIN. "Two-Point Step Size Gradient Methods". In: *IMA Journal of Numerical Analysis* 8.1 (1988), pp. 141–148. DOI: 10.1093/imanum/8.1.141. eprint: /oup/backfile/content_public/journal/imajna/8/1/10.1093/imanum/8.1.141/2/8-1-141.pdf. URL: +%20http://dx.doi.org/10.1093/imanum/8.1.141.

[136] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. eprint: arXiv:1609.04747.

[137] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[138] Karen L. Schuchardt et al. "Basis Set Exchange: A Community Database for Computational Sciences". In: *Journal of Chemical Information and Modeling* 47.3 (2007). PMID: 17428029, pp. 1045–1052. DOI: 10.1021/ci600510j. eprint: http://dx.doi.org/10.1021/ci600510j. URL: http://dx.doi.org/10.1021/ci600510j.

[139] *TURBOMOLE V7.2 2017, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from*
`http://www.turbomole.com`.

[140] J Stephen Binkley, John A Pople, and Warren J Hehre. "Self-consistent molecular orbital methods. 21. Small split-valence basis sets for first-row elements". In: *Journal of the American Chemical Society* 102.3 (1980), pp. 939–947.

[141] RBJS Krishnan et al. "Self-consistent molecular orbital methods. XX. A basis set for correlated wave functions". In: *The Journal of Chemical Physics* 72.1 (1980), pp. 650–654.

[142] I. I. Guseinov and B. A. Mamedov. "Evaluation of the Boys Function using Analytical Relations". In: *Journal of Mathematical Chemistry* 40.2 (Aug. 2006), pp. 179–183. ISSN: 1572-8897. DOI: 10.1007/s10910-005-9023-3. URL: https://doi.org/10.1007/s10910-005-9023-3.

[143] N. M. Temme. "A Set of Algorithms for the Incomplete Gamma Functions". In: *Probability in the Engineering and Informational Sciences* 8.2 (1994), pp. 291–307. DOI: 10.1017/S0269964800003417.

[144] I.I. Guseinov and B.A. Mamedov. "Evaluation of Incomplete Gamma Functions Using Downward Recursion and Analytical Relations". In: *Journal of Mathematical Chemistry* 36.4 (Aug. 2004), pp. 341–346. ISSN: 1572-8897. DOI: 10.1023/B:JOMC.0000044521.18885.d3. URL: https://doi.org/10.1023/B:JOMC.0000044521.18885.d3.

[145] M.L. Boas. *Mathematical Methods in the Physical Sciences*. 3rd ed. John Wiley & Sons, 2006. ISBN: 0-471-19826-9.

[146] Péter Pulay. "Convergence acceleration of iterative sequences. the case of scf iteration". In: *Chemical Physics Letters* 73.2 (1980), pp. 393–398. ISSN: 0009-2614. DOI: https://doi.org/10.1016/0009-2614(80)80396-4. URL: http://www.sciencedirect.com/science/article/pii/0009261480803964.

[147] A.S. Householder. *The Theory of Matrices in Numerical Analysis*. Blaisdell Publishing Co., 1964. ISBN: 1-124-05391-3.

[148] D. Ceperley, G. V. Chester, and M. H. Kalos. "Monte Carlo simulation of a many-fermion study". In: *Phys. Rev. B* 16 (7 Oct. 1977), pp. 3081–3099. DOI: 10.1103/PhysRevB.16.3081. URL: https://link.aps.org/doi/10.1103/PhysRevB.16.3081.

[149] Paul Hanson (https://math.stackexchange.com/users/219585/paul-hanson). *Insightful proofs for Sherman-Morrison Formula and Matrix Determinant Lemma*. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/1167112 (version: 2015-02-27). eprint: https://math.stackexchange.com/q/1167112. URL: https://math.stackexchange.com/q/1167112.

[150] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: %5Curl%7Bhttps://www.tensorflow.org/%7D.

[151] Anaconda. *Anaconda Software Distribution (version 2-2.4.0)*. 2016. URL: %5Curl%7Bhttps://anaconda.com%7D.

[152] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[153] Dandelion Mane et al. *TensorBoard*. https://github.com/tensorflow/tensorboard. 2017.

[154] H. P. Langtangen. *A Primer on Scientific Programming with Python*. 2nd ed. Springer-Verlag Berlin Heidelberg, 2011. ISBN: 978-3-642-18365-2.

[155] Stephen R. Langhoff, Charles W. Bauschlicher, and Harry Partridge. "Theoretical Dissociation Energies for Ionic Molecules". In: *Comparison of Ab Initio Quantum Chemistry with Experiment for Small Molecules: The State of the Art Proceedings of a Symposium Held at Philadelphia, Pennsylvania, 27–29 August, 1984*. Ed. by Rodney J. Bartlett. Dordrecht: Springer Netherlands, 1985, pp. 357–407. ISBN: 978-94-009-5474-8. DOI: 10.1007/978-94-009-5474-8_13. URL: https://doi.org/10.1007/978-94-009-5474-8_13.

[156] A. Hinchliffe. *Modelling Molecular Structures*. 2nd ed. Wiley, 2000. ISBN: 0-471-62380-6.

[157] Claudia Filippi and C. J. Umrigar. "Multiconfiguration wave functions for quantum Monte Carlo calculations of first-row diatomic molecules". In: *The Journal of Chemical Physics* 105.1 (1996), pp. 213–226. DOI: 10.1063/1.471865. eprint: https://doi.org/10.1063/1.471865. URL: https://doi.org/10.1063/1.471865.

[158] P. Atkins and J. de Paula. *Atkins' Physical Chemistry*. 10th ed. Oxford University Press, 2014. ISBN: 987-0-19-969740-3.

[159] E. Buendía et al. "Quantum Monte Carlo ground state energies for the atoms Li through Ar". In: *The Journal of Chemical Physics* 131.4 (2009), p. 044115. DOI: 10.1063/1.3187526. eprint: https://doi.org/10.1063/1.3187526. URL: https://doi.org/10.1063/1.3187526.

[160] Jules W Moskowitz and MH Kalos. "A new look at correlations in atomic and molecular systems. I. Application of fermion Monte Carlo variational method". In: *International Journal of Quantum Chemistry* 20.5 (1981), pp. 1107–1119.

[161] E. Buendía, F.J. Gálvez, and A. Sarsa. "Correlated wave functions for the ground state of the atoms Li through Kr". In: *Chemical Physics Letters* 428.4 (2006), pp. 241–244. ISSN: 0009-2614. DOI: https://doi.org/10.1016/j.cplett.2006.07.027. URL: http://www.sciencedirect.com/science/article/pii/S0009261406010372.

[162] James D Talman and William F Shadwick. "Optimized effective atomic central potential". In: *Physical Review A* 14.1 (1976), p. 36.

[163] A Ma et al. "Scheme for adding electron–nucleus cusps to Gaussian orbitals". In: *The Journal of chemical physics* 122.22 (2005), p. 224322.

[164] E. Clementi, D. L. Raimondi, and W. P. Reinhardt. "Atomic Screening Constants from SCF Functions. II. Atoms with 37 to 86 Electrons". In: *The Journal of Chemical Physics* 47.4 (1967), pp. 1300–1307. DOI: 10.1063/1.1712084. eprint: https://doi.org/10.1063/1.1712084. URL: https://doi.org/10.1063/1.1712084.

[165] O. El Akramine, A. C. Kollias, and W. A. Lester Jr. "Quantum Monte Carlo study of singlet–triplet transition in ethylene". In: *The Journal of Chemical Physics* 119.3 (2003), pp. 1483–1488. DOI: 10.1063/1.1579466. eprint: https://doi.org/10.1063/1.1579466. URL: https://doi.org/10.1063/1.1579466.

[166] Peter J. Reynolds et al. "Fixed-node quantum Monte Carlo for moleculesa) b)". In: *The Journal of Chemical Physics* 77.11 (1982), pp. 5593–5603. DOI: 10.1063/1.443766. eprint: https://doi.org/10.1063/1.443766. URL: https://doi.org/10.1063/1.443766.

[167] Sebastian Manten and Arne Lüchow. "On the accuracy of the fixed-node diffusion quantum Monte Carlo method". In: *The Journal of Chemical Physics* 115.12 (2001), pp. 5362–5366. DOI: 10.1063/1.1394757. eprint: https://doi.org/10.1063/1.1394757. URL: https://doi.org/10.1063/1.1394757.

[168] Frank H. Stillinger and Thomas A. Weber. "Computer simulation of local order in condensed phases of silicon". In: *Phys. Rev. B* 31 (8 Apr. 1985), pp. 5262–5271. DOI: 10.1103/PhysRevB.31.5262. URL: https://link.aps.org/doi/10.1103/PhysRevB.31.5262.

[169] P.J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. 2nd ed. Academic Press, 1984. ISBN: 0-12-206360-0.

[170] T. Lindstrøm. *Mathematical Analysis*. Lecture notes. 2017. URL: http://folk.uio.no/snorrec/17VMat2400/Spaces.pdf.

[171] K. Mørken. *Numerical Algorithms and Digital Representation*. Lecture notes. Sept. 2017. URL: http://www.uio.no/studier/emner/matnat/math/MAT-INF1100/h17/kompendiet/matinf1100.pdf.

[172] V.I. Krylov. *Approximate Calculation of Integrals*. 1st ed. Dover Publications, 2005. ISBN: 0-486-44579-8.

[173] P. Echenique and J.L. Alonso. "A mathematical and computational review of Hartree-Fock SCF methods in quantum chemistry". In: *Molecular Physics* 105.23-24 (2007), pp. 3057–3098. DOI: 10.1080/00268970701757875. eprint: http://dx.doi.org/10.1080/00268970701757875. URL: http://dx.doi.org/10.1080/00268970701757875.