Bachelor's Thesis

Numerical Solution of the Dirac Equation for an Arbitrary Central Potential

Christopher Gerrit Mertens

July 8, 2024

Fundamentale Physik für Metrologie
Prof. Dr. Andrey Surzhykov

Supervisor:
Andrey Surzhykov

Statement of Originality

This thesis has been performed independently with the support of my supervisor/s. To the best of the author's knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text.

Braunschweig, July 8, 2024

Abstract

Due to their simple structure, hydrogen-like systems are well suited to study the influence of certain physical effects on the experimental outcome. These systems are described in quantum mechanics using the Schrödinger equation, or, considering relativistic effects, using the Dirac equation. In this thesis we will first recall the analytic solutions for a point-like nucleus and then derive a method to solve both equations numerically to find solutions for an arbitrary central potential. The main focus is on an easy-to-understand discussion of the used numerical methods to provide an introduction into the world of atomic computer codes. To see an actual implementation of the discussed methods, as a main result the Python package `dish` (**DI**rac **S**olver for **H**ydrogen-like systems) is developed and benchmarked against existing codes. Also, a user guide to `dish` is provided.

Zusammenfassung

Wasserstoffähnliche Atome eignen sich durch ihre einfache Struktur gut, um den Einfluss bestimmter physikalischer Effekte auf experimentelle Ergebnisse zu berechnen. Quantenmechanisch werden diese Systeme durch die Schrödinger Gleichung, oder, unter Berücksichtigung relativistischer Effekte, durch die Dirac Gleichung beschrieben. In dieser Arbeit werden wir zunächst die analytischen Lösungen für den Spezialfall eines punktförmigen Kerns diskutieren und anschließend eine Methode entwickeln, beide Gleichungen numerisch für beliebige Zentralpotentiale zu lösen. Das Hauptaugenmerk liegt dabei auf einer einfach verständlichen Einführung in die verwendeten numerischen Methoden, um einen Einstieg in die Welt der *atomic codes* zu geben. Als Hauptergebnis wird das Python-Paket `dish` (**DI**rac **S**olver for **H**ydrogen-like systems), das den diskutierten numerischen Ansatz implementiert, entwickelt und dessen Ergebnisse gegen bestehende Software getestet. Außerdem wird eine Anleitung zur Verwendung von `dish` bereitgestellt.

Contents

List of Figures

List of Tables

1. Introduction

The hydrogen atom is one of the best understood systems in quantum mechanics. Because of its simplicity it is well suited for studying results found in experiments and predicting the influence of certain physical effects on the experimental outcome.
The hydrogen atom is described by the Dirac equation or by the Schrödinger equation when neglecting relativistic effects. For specific cases there are analytic solutions to these equations, but for most problems they need to be solved numerically. To find these solutions a wide range of atomic computer codes (e.g. GRASP [1–4], ATSP [5, 6], FAC [7] and JAC [8]) was developed since computers became fast enough to be useful for this task in the 1970's. Since then, there has made been huge advance in the technology and in the theory which resulted in growing codes with many more features and application possibilities. Yet, they are mostly developed based on the original codes and, hence, restricted by the used programming language (mostly Fortran) which allows very performant code but is rather hard to understand and maintain especially in large code bases. Additionally, the codes were and are developed by people working in the field often for many years who therefore are experts and can find their way through all the used mathematics and notations even with the limited documentation available.
There are great books available for atomic physics like I.P. Grant's "bible" [9] or Walter Johnson's lectures on atomic structure theory [10] but, because they cover an enormous amount of content, they are also tremendous. This thesis aims to give an easy introduction into numerical methods used in atomic physics. It is written for people who are familiar with the content of a Quantum Mechanics I course, have some knowledge about special relativity, and want to dive deeper in the computational methods used to describe simple atomic systems.
As a main result a small atomic code toolkit has been developed which allows looking at an actual implementation along the mathematical methods compiled in the thesis. The tool is called `dish` (**DI**rac **S**olver for **H**ydrogen-like systems) and is implemented in Python. As Python is a well established programming language at universities and has a simple syntax the goal was to be able to focus on the subject-matter which is facilitated by self-documenting names throughout `dish`and its structure in general.

As a start to this thesis in chapter 2 for both, the relativistic and non-relativistic description, the respective equations modeling that case are recalled and the properties of its solutions are investigated while deriving these for the model of a point-like nucleus. In the next chapter an approach is developed for finding solutions when the nucleus is modeled using other central potentials aiming to reflect effects such as a finite nuclear size. Then `dish`, based on this approach, is benchmarked against analytical results and existing codes. Finally, in chapter 5 an introduction is given on how to use `dish`.

2. Theoretical Background

2.1. Hydrogen-like Atoms

Hydrogen-like atoms (also called hydrogenic atoms) are atoms or ions that consist of a positively charged nucleus and a single electron [11, p. 1219]. Beside Hydrogen itself these are particularly highly-charged ions like Xe^{53+}, Yb^{69+} or U^{91+} which are used to study quantum electrodynamics (QED) effects like the lamb shift. Their usage is also considered in atomic clocks for searching for a variation in fundamental constants like the Fine-structure constant α [12, 13]. Atoms or ions with a single valence electron can also be approximated as hydrogenic by considering the valence electron in the effective field of the nucleus and the closed shell electrons.
The simplicity of this system allows for a precise theoretical analysis since the single electron is only effected by the potential of the nucleus and no other interactions are considered. And therefore, the state of the system can be described by a single-electron-wavefunction.

2.2. Angular Momentum Operator Algebra

Before starting into the discussion of hydrogenic atoms some knowledge of the angular momentum operator algebra is required which will be referenced later in the discussion. Here just a short introduction into the angular momentum operator algebra will be given and the results important for this thesis will be covered. A more detailed discussion can be found in [10, 14].
Per definition, a vector operator $\hat{\mathbf{J}} = \{\hat{\mathbf{J}}_x, \hat{\mathbf{J}}_y, \hat{\mathbf{J}}_z\}$ that fulfills the commutator relation

$$\left[\hat{\mathbf{J}}_j, \hat{\mathbf{J}}_k\right] = i\hbar\varepsilon_{jkl}\hat{\mathbf{J}}_l\,, \quad \text{with } j,k,l \in \{x,y,z\} \tag{2.1}$$

where ε_{jkl} is the Levi-Civita-symbol, is called an angular momentum operator.
By defining the operator $\hat{\mathbf{J}}^2 = \hat{\mathbf{J}}_x^2 + \hat{\mathbf{J}}_y^2 + \hat{\mathbf{J}}_z^2$ it follows directly that

$$\left[\hat{\mathbf{J}}^2, \hat{\mathbf{J}}_k\right] = 0\,, \quad \text{with } k \in \{x,y,z\} \tag{2.2}$$

and, therefore, one finds a set of simultaneous eigenfunctions $\{\phi\}$ of the operators $\hat{\mathbf{J}}^2$ and $\hat{\mathbf{J}}_z$. For each operator which is in the set of commuting operators there is a quantum number which we will call in this case j for $\hat{\mathbf{J}}^2$ and m for $\hat{\mathbf{J}}_z$, i.e.

$$\begin{aligned}\hat{\mathbf{J}}^2\phi_{jm} &= \lambda_j\phi_{jm} \quad \text{and}\\ \hat{\mathbf{J}}_z\phi_{jm} &= \lambda_m\phi_{jm}\,.\end{aligned}$$

Further inspection of the commutators shows for the eigenvalues

$$\lambda_j = \hbar^2 j\,(j+1) \quad \text{and} \quad \lambda_m = \hbar m\,,$$

where the quantum numbers are restricted by the following rules:

- The value of j is positive and can either be integral or half-integral.
- $|m_{\min}| = m_{\max} = j$
- The difference between two adjacent values of both j and m is ± 1.

j is a measure for the magnitude of the angular momentum and m yields the projection of $\hat{\mathbf{J}}$ on the z-axis.
To describe hydrogenic systems two angular momentum operators are important, namely the orbital angular momentum of the electron

$$\hat{\mathbf{L}} = \hat{\mathbf{r}} \times \hat{\mathbf{p}} \tag{2.3}$$

where the quantum numbers will be called l and m for $\hat{\mathbf{L}}^2$ and $\hat{\mathbf{L}}_z$ respectively and l is integral, and the spin angular momentum (short: *spin*) with the quantum numbers s and μ. In the case $s = \frac{1}{2}$ the spin of an electron is described, and the operator is given by

$$\hat{\mathbf{S}} = \frac{1}{2}\hbar\hat{\boldsymbol{\sigma}} = \frac{1}{2}\hbar\left(\sigma_x, \sigma_y, \sigma_z\right)^\top \tag{2.4}$$

where σ_k are the Pauli-matrices

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \text{and} \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.5}$$

Since $s = \frac{1}{2}$ is fixed μ can have the values $\mu = \pm\frac{1}{2}$ and the simultaneous eigenstates of $\hat{\mathbf{S}}^2$ and $\hat{\mathbf{S}}_z$, the so called *spinors*, are

$$\chi_{1/2} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \chi_{-1/2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.6}$$

In position space the momentum operator is given by $\hat{\mathbf{p}} = -i\hbar\boldsymbol{\nabla}$. Using this, $\hat{\mathbf{L}}$ in spherical coordinates (r, θ, φ) results into

$$\hat{\mathbf{L}} = i\hbar \begin{pmatrix} \sin\varphi\frac{\partial}{\partial\theta} + \cos\varphi\cot\theta\frac{\partial}{\partial\varphi} \\ -\cos\varphi\frac{\partial}{\partial\theta} + \sin\varphi\cot\theta\frac{\partial}{\partial\varphi} \\ -\frac{\partial}{\partial\varphi} \end{pmatrix} \tag{2.7}$$

and the simultaneous eigenfunctions of $\hat{\mathbf{L}}^2$ and $\hat{\mathbf{L}}_z$ are the so called spherical harmonics

$$Y_{lm}(\theta, \varphi) = \frac{1}{\sqrt{2\pi}} e^{im\varphi}(-1)^m \sqrt{\frac{(2l+1)(l-m)!}{2(l+m)!}} P_l^m\left(\cos\theta\right) \tag{2.8}$$

where $P_l^m\left(\cos\theta\right)$ are the associated Legendre polynomials. In correspondence with the name of the operator, l is called the *orbital angular momentum quantum number*, also known as the *azimuthal quantum number*, and m is called *magnetic quantum number*.

2.2.1. Coupling of Angular Momenta

Two angular momenta $\hat{\mathbf{J}}_1$ and $\hat{\mathbf{J}}_2$ which act in two separate Hilbert spaces can couple via

$$\hat{\mathbf{J}} = \hat{\mathbf{J}}_1 + \hat{\mathbf{J}}_2 \,, \tag{2.9}$$

where $\hat{\mathbf{J}}$ fulfills eq. (2.1) and, hence, is called total angular momentum.
Since $\hat{\mathbf{J}}_1$ and $\hat{\mathbf{J}}_2$ act in separate Hilbert spaces $\hat{\mathbf{J}}_1^2, \hat{\mathbf{J}}_{1,z}, \hat{\mathbf{J}}_2^2$ and $\hat{\mathbf{J}}_{2,z}$ commute and a set of common eigenfunctions $\{|j_1 m_1, j_2 m_2\rangle\}$[1] exists. It can be shown for the total angular momentum that $\hat{\mathbf{J}}^2$ also commutes with the above-mentioned operators but $\hat{\mathbf{J}}_z$ does not commute with $\hat{\mathbf{J}}_{1/2,z}$ and therefore has not the eigenfunctions $|j_1 m_1, j_2 m_2\rangle$. But since $\hat{\mathbf{J}}_z$ commutes with $\hat{\mathbf{J}}^2, \hat{\mathbf{J}}_1^2$ and $\hat{\mathbf{J}}_2^2$ common eigenfunctions $\left|j m_j j_1 j_2\right\rangle$ of these operators exist where the latter eigenfunctions are a superposition of the former ones:

$$\left|j m_j j_1 j_2\right\rangle = \sum_{m_1, m_2} \left\langle j_1 m_1, j_2 m_2 \middle| j m_j \right\rangle \left|j m_j j_1 j_2\right\rangle \tag{2.10}$$

The coefficients are called Clebsch-Gordan coefficients and the notation $\left\langle j_1 m_1, j_2 m_2 \middle| j m_j \right\rangle$ is a short form for the scalar product $\left\langle j_1 m_1, j_2 m_2 \middle| j m_j j_1 j_2 \right\rangle$. The eigenvalues j and m_j can have the values $|j_1 - j_2| < j < j_1 + j_2$ and $\left|m_j\right| \leq j$ respectively.
The coupling between the orbital and the spin angular momentum of a spin-$\frac{1}{2}$-particle is therefore described by $\hat{\mathbf{J}} = \hat{\mathbf{L}} + \hat{\mathbf{S}}$ where j can have the values $l \pm \frac{1}{2}$. Using the eigenfunctions associated with $\hat{\mathbf{L}}$ (eq. (2.7)) and with $\hat{\mathbf{S}}$ (eq. (2.6)) in eq. (2.10) one gets the so-called *spherical spinors*

$$\Omega_{jlm}(\theta, \varphi) = \sum_{\mu} \left\langle l, m-\mu; \frac{1}{2}\mu \middle| j, m \right\rangle Y_{l,m-\mu}(\theta, \varphi) \chi_\mu \,. \tag{2.11}$$

From computation of the Clebsch-Gordan coefficients an explicit version can be found:

$$\Omega_{l+\frac{1}{2},l,m}(\theta, \varphi) = \begin{pmatrix} \sqrt{\frac{l+m+1/2}{2l+1}} Y_{l,m-\frac{1}{2}}(\theta, \varphi) \\ \sqrt{\frac{l-m+1/2}{2l+1}} Y_{l,m+\frac{1}{2}}(\theta, \varphi) \end{pmatrix} \quad \text{and} \tag{2.12a}$$

$$\Omega_{l-\frac{1}{2},l,m}(\theta, \varphi) = \begin{pmatrix} -\sqrt{\frac{l-m+1/2}{2l+1}} Y_{l,m-\frac{1}{2}}(\theta, \varphi) \\ \sqrt{\frac{l+m+1/2}{2l+1}} Y_{l,m+\frac{1}{2}}(\theta, \varphi) \end{pmatrix} . \tag{2.12b}$$

Because the spherical spinors are eigenfunctions of $\hat{\mathbf{J}}^2 = \hat{\mathbf{L}}^2 + 2\hat{\mathbf{S}}\hat{\mathbf{L}} + \hat{\mathbf{S}}^2$ they are also eigenfunctions of $\hat{\sigma}\hat{\mathbf{L}}$ and therefore of $K := -1 - \hat{\sigma}\hat{\mathbf{L}}$. The eigenvalue equation

$$K\Omega_{jlm}(\theta, \varphi) = \kappa \Omega_{jlm}(\theta, \varphi) \tag{2.13}$$

is fulfilled for the eigenvalues $\kappa = \mp\left(j + \frac{1}{2}\right)$ for $j = l \pm \frac{1}{2}$. The value of κ is integer and determines both j and l and therefore the notation $\Omega_{\kappa m} \equiv \Omega_{jlm}$ will be used.
The spherical spinors are orthonormal:

$$\int_0^\pi \sin\theta \, \mathrm{d}\theta \int_0^{2\pi} \mathrm{d}\varphi \; \Omega^\dagger_{\kappa' m'}(\theta, \varphi) \Omega_{\kappa m}(\theta, \varphi) = \delta_{\kappa' \kappa} \delta_{m' m} \tag{2.14}$$

[1] In the following often the Bra-ket notation will be used. In the appendix in chapter A.1 a short introduction is given.

2.3. Non-relativistic Model of Hydrogen-like Atoms

Here a brief introduction in describing a hydrogenic system using non-relativistic quantum mechanics is given. It is based on [15, 16] where more details can be found.

2.3.1. The Schrödinger Equation

The probability density of the electron is given by $|\Psi|^2$ where, in a non-relativistic approximation, the wave function Ψ is the solution of the Schrödinger-equation (SE)

$$i\hbar\frac{\partial\Psi}{\partial t} = \hat{H}\Psi\,. \tag{2.15}$$

The Hamiltonian

$$\hat{H} = \frac{\hat{\mathbf{p}}^2}{2m} + \hat{V}(\mathbf{r}) \tag{2.16}$$

describes the hydrogenic system. The first term is the kinetic energy part with the momentum operator $\hat{\mathbf{p}} = -i\hbar\boldsymbol{\nabla}$ and the particles mass m. The second term corresponds to the potential energy by the operator defining the potential $\hat{V} = V$.
In hydrogen-like system the electron experiences the potential caused by the electric charge Ze of the nucleus, where Z is the number of its protons and $e \approx 1.602 \times 10^{-19}\,\mathrm{C}$ is the elementary charge. The coordinate system can be chosen so that the charge is at the origin, and, since the space is uniform, it is spherical symmetric. Using spherical coordinates (r, θ, φ) to describe the problem, the potential yields $V(\mathbf{r}) = V(r)$. As a good approximation the nucleus can be considered point-like, and then the potential is the Coulomb-potential

$$V(r) = -\frac{1}{4\pi\varepsilon_0}\frac{Ze^2}{r} \tag{2.17}$$

where the proportionality constant $k_C = 1/(4\pi\varepsilon_0)$ is called Coulomb's constant and ε_0 is the electric constant. Since in this thesis only time-invariant potentials will be considered, the wave function $\Psi(\mathbf{r}, t)$ can be separated into a spacial and a time-dependent part

$$\Psi(\mathbf{r}, t) = \psi(\mathbf{r})\phi(t)\,. \tag{2.18}$$

By plugging the *ansatz* eq. (2.18) in the SE eq. (2.15) it can be seen that

$$\phi(t) = \mathrm{e}^{-iEt/\hbar} \tag{2.19}$$

where E is the total energy of the electron which is found by solving the remaining eigenvalue-equation

$$\hat{H}\psi = E\psi\,. \tag{2.20}$$

There are two cases which need to be distinguished: If the electrons kinetic energy $\hat{\mathbf{p}}^2/2m$ is larger than the potential energy the electron is in a so-called *free state*. And, if the potential energy $|V|$ is larger than the electron is in a *bound state*. As the name suggests, in the former case the electron is not tied to the nucleus and therefore has non-zero probability density far away from it. Nonetheless, the electron is influenced by the nucleus. This thesis will handle the latter case where the electron is bound to the nucleus i.e. its probability density is negligible in sufficient distance from the origin.

In the regime of lightweight nuclei the SE is a good approximation for a Hydrogen-like system. For heavy nuclei relativistic effects are larger, and therefore the system is described more accurately by the Dirac-Equation which will be discussed in the next section.

Hartree Atomic Units

For all further derivations and especially calculations it will be convenient to work in another system of units: Hartree atomic units (a.u.) are widely used in atomic physics as it simplifies the coefficients of many equations, and they are better suited for calculations since numerical errors from combining units and the atomic scales, which values differ in many orders of magnitude, are avoided.
The coherent units of this unit system (i.e. their value is 1) are the mass of the electron $m_e \approx 9.109 \times 10^{-31}\,\mathrm{kg}$, the electrons charge $|e| \approx 1.602 \times 10^{-19}\,\mathrm{C}$, the Coulomb's constant $k_e = 1/(4\pi\varepsilon_0) \approx 8.988 \times 10^{9}\,\mathrm{Nm^2C^{-2}}$ and the reduced Planck's constant $\hbar \approx 6.582 \times 10^{-16}\,\mathrm{eV\,s}$. Therefore, the atomic unit of mass is the electron rest mass, the atomic unit of length is the Bohr radius $a_0 = 4\pi\varepsilon_0\hbar^2/me^2 \approx 0.529\,\text{Å}$ and for energy it is the hartree $E_h = me^4/(4\pi\varepsilon_0\hbar)^2 \approx 27.211\,\mathrm{eV}$. Another important constant is the dimensionless fine structure constant $\alpha = e^2/4\pi\varepsilon_0\hbar c \approx 1/137$ as in atomic units the speed of light is $c = \alpha^{-1}$.
Units for other quantities can be derived from the above ones. [10, ch. 1]
For the values that are currently recommended for the physical constants see [17].
To convert results computed in a.u. the `dish` software provides a utility function which is described in chapter 5.2.1.

2.3.2. An Electron in a Coulomb Potential

The following derivation is based on [10]. Only the important steps and considerations will be given here, the rest of the calculations will be left to the reader or can be found in more detail in [9, 10, 15] or many other text books.
In spherical coordinates the Laplace-operator is given by

$$\Delta \equiv \boldsymbol{\nabla}^2 = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\varphi^2}\,. \tag{2.21}$$

Seeking a solution of the form

$$\psi(\mathbf{r}) = \frac{1}{r} R(r) Y(\theta, \varphi) \tag{2.22}$$

of the SE in spherical coordinates

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial \psi}{\partial r}\right) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial \psi}{\partial \theta}\right) + \frac{1}{r^2 \sin^2\theta}\frac{\partial^2 \psi}{\partial \varphi^2} + 2m\left(E - V(r)\right)\psi = 0 \tag{2.23}$$

one obtains

$$\frac{1}{\sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial Y}{\partial \theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2 Y}{\partial \varphi^2} + \lambda Y = 0 \tag{2.24a}$$

$$\frac{\mathrm{d}^2 R}{\mathrm{d}r^2} + 2\left(m(E - V(r)) - \frac{\lambda}{2r^2}\right) R = 0 \tag{2.24b}$$

for the functions R and Y, where λ is a separation constant. Setting $\lambda := l(l+1), l \in \mathbb{N}_0$ eq. (2.24a) is solved by the spherical harmonics $Y_{lm}(\theta, \varphi)$ (see eq. (2.8)), which have been introduced in the last section as the eigenfunctions of $\hat{\mathbf{L}}^2$ and $\hat{\mathbf{L}}_z$.
The function $R(r)$ is called *radial function* and it's normalization condition

$$\int_0^\infty R^2(r)\mathrm{d}r = 1 \tag{2.25}$$

arises from normalization of the whole wave function $\psi(\mathbf{r})$ by

$$\int \psi^\dagger(\mathbf{r})\psi(\mathbf{r})\mathrm{d}r = 1\,. \tag{2.26}$$

Modeling the nucleus using a Coulomb potential eq. (2.17), eq. (2.24b) becomes

$$\frac{\mathrm{d}^2 R}{\mathrm{d}r^2} + 2\left(mE + \frac{Zm}{r} - \frac{l(l+1)}{2r^2}\right) R = 0\,, \tag{2.27}$$

which is called the radial SE.
The aim is to find solutions that are normalizable using eq. (2.25), which only exist for discrete eigenvalues of the energy $E = E_{nl}$. To find these eigenvalues and the corresponding eigenfunctions $R_{nl}(r)$, we look at the asymptotic behavior of functions fulfilling eq. (2.27). Using a monomial *ansatz* one finds for $r \to 0$,

$$R(r) \to \begin{cases} r^{l+1} & \text{or} \\ r^{-l} \end{cases} \tag{2.28}$$

and since the solution must be normalizable only the first option is valid since it is regular at the origin. For $r \to \infty$ the potential vanishes and the asymptotic version of eq. (2.27), $\mathrm{d}_r^2 R(r) + 2mER(r) = 0$ can be solved by

$$R(r) \to \begin{cases} \mathrm{e}^{-\sqrt{-2mE}r} & \text{or} \\ \mathrm{e}^{\sqrt{-2mE}r}\,, \end{cases} \tag{2.29}$$

where again only the first variant is normalizable.
Using the information obtained from the asymptotic behavior we write

$$R(r) = r^{l+1} \mathrm{e}^{-\sqrt{-2mE}r} F(r) \tag{2.30}$$

and when using this *ansatz* in eq. (2.27), we find that F satisfies Kummer's equation

$$x \frac{\mathrm{d}^2 F(x)}{\mathrm{d}x^2} + (b - x) \frac{\mathrm{d}F}{\mathrm{d}x} - aF = 0\,, \tag{2.31}$$

where $x = 2\sqrt{-2mE}r, a = l + 1 - Zm/\sqrt{-2mE}$ and $b = 2(l+1)$. Its solutions, that are regular at the origin, are the *confluent hypergeometric functions*

$$F(a, b, x) = \sum_{k=0}^{\infty} \frac{(a+k-1)!}{(a-1)!} \frac{(b-1)!}{(b+k-1)!} \frac{x^k}{k!}\,, \quad \text{with} \quad a, b \in \mathbb{Z}\,. \tag{2.32}$$

For $a \leq 0$ F is a polynomial of degree $|a|$, and $n_r := -a$ equals the number of nodes (roots) of $R(r)$ for $r > 0$ (which is the definition range of R). From $a = l + 1 - Zm/\sqrt{-2mE}$ follows

$$\begin{aligned} \sqrt{-2mE} &= \frac{Zm}{n_r + l + 1} = \frac{Zm}{n}\,, \quad \text{where} \quad n := n_r + l + 1 \\ \iff E &= E_n = -\frac{Z^2 m}{2n^2} \end{aligned} \tag{2.33}$$

for the energy eigenvalue of eq. (2.15). Because n_r and l are positive integers n is also a positive integer, and is called the *principal quantum number*. Since a single value of n can be constructed from the values $l = 0, 1, \ldots, n-1$ and a corresponding number of nodes, the eigenvalue is degenerate of degree n. Hence, there are n corresponding eigenfunctions

$$R_{nl} = N_{nl} \left(\frac{2Zmr}{n}\right)^{l+1} \mathrm{e}^{-Zmr/n} F\left(-n + l + 1, 2l + 2, 2\frac{Zmr}{n}\right) \tag{2.34a}$$

where the normalization constant N_{nl} is determined by evaluating eq. (2.25) analytically, which yields

$$N_{nl} = \frac{1}{n(2l+1)!} \sqrt{\frac{Zm(n+l)!}{(n-l-1)!}}\,. \tag{2.34b}$$

In the case of normal atoms the particle which we are interested in is an electron. In atomic units its mass is $m = m_e = 1$. The equations above are also valid for exotic atoms which are formed by a positively charged nucleus and a muon, which is also a negatively charged spin $\frac{1}{2}$ particle like the electron, but has a mass that is approximately 200 times larger than the electrons. Some results for this case will be shown in chapter 4.1.3.

In fig. 2.1 the radial functions R of the states $3p$ and $5d$ of hydrogenic uranium and their derivatives Q are plotted as two examples.

Recalling the above results, a solution to the time-independent SE for a Coulomb potential

$$\left(\frac{\hat{\mathbf{p}}^2}{2m} - \frac{Ze^2}{r}\right) \psi(\mathbf{r}) = E\psi(\mathbf{r}) \tag{2.35}$$

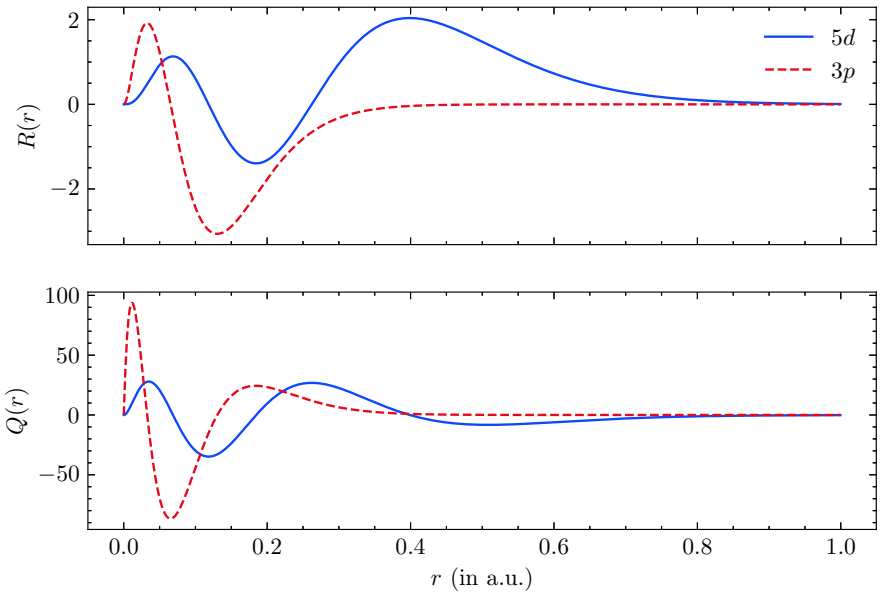

Figure 2.1.: The radial part of the wave function and it's derivative for the states $3p$ and $5d$ for hydrogen-like uranium.

is characterized by the three quantum numbers n, l and m. The eigenfunction, which we will call an eigenstate (or just state), is given by

$$\psi \equiv \psi_{nlm}(\mathbf{r}) = \frac{1}{r} R_{nl}(r) Y_{lm}(\theta, \varphi)\,. \tag{2.36}$$

In this thesis $R_{nl}(r)$ given by eqs. (2.34) will be referenced as radial function, where in literature usually $R_{nl}(r)/r$ is called radial function. This choice is done to avoid numerical errors that would occur by dividing by small r.
Without any external fields like in this discussed case, the eigenenergy $E \equiv E_n$ given by eq. (2.33) is independent of both quantum numbers associated with the angular momentum of the state.

In the above derivations, the nucleus was assumed to be static which is equivalent to an infinite mass. Describing the nucleus with a finite mass M resulting in experience of recoil, the problem can be solved in the center of mass system. The formulas of the wave function and the energy are the same and only the mass m of the particle needs to be replaced by the reduced mass

$$\mu = \frac{mM}{M+m} = \frac{m}{1+\frac{m}{M}}\,. \tag{2.37}$$

As can be seen from eq. (2.33), the energy is proportional to the reduced mass and therefore the binding is weaker for lighter nuclei.

2.4. Relativistic Model of Hydrogen-like Atoms

In coordinate space it can be easily seen, that because of the derivatives of different orders for time and space the SE cannot be Lorentz-invariant. Here, a very short outline of how such an equivalent of the SE that fulfills the relativistic requirements was found historically, is given. It is based on [14, 18]. The focus will be on solving it for the hydrogenic systems which is discussed in the next subsection. It is assumed that the reader is familiar with the basic concepts of special relativity and the required mathematics, especially the concept of co- and contravariant vectors. As a convention here the Minkowski-space is defined with the space-time position four-vector $x^\mu = (ct, \mathbf{r})^\top$ and the metric $g = \mathrm{diag}(1, -1, -1, -1)$.

Applying the relativistic energy-momentum-relation for a free particle

$$E = \sqrt{\mathbf{p}^2c^2 + m^2c^4} \implies \hat{H} = \sqrt{\hat{\mathbf{p}}^2c^2 + m^2c^4} \tag{2.38}$$

in the SE one finds

$$i\frac{\partial}{\partial t}\psi = \sqrt{-c^2\boldsymbol{\nabla}^2 + m^2c^4}\,\psi \tag{2.39}$$

in position space. Because the expansion of the square root yields derivatives $\boldsymbol{\nabla}^{2n}, n \in \mathbb{N}$ it is not possible to construct a Lorentz-invariant wave equation. By applying $-\partial/\partial t$ twice this problem is avoided and one gets

$$-\frac{\partial^2}{\partial t^2}\psi = \left(-c^2\boldsymbol{\nabla}^2 + m^2c^4\right)\psi\,. \tag{2.40}$$

Using the four-gradient $\partial/\partial x^\mu = \left(\partial/\partial ct\,, \boldsymbol{\nabla}\right)^\top \equiv \partial_\mu$ this equation can be written more compact as

$$0 = \left(\partial_\mu\partial^\mu c^2 + m^2c^4\right)\psi \tag{2.41}$$

and is called *Klein-Gordan-equation*. One can show that this equation is Lorentz-invariant and, by construction, fulfills the relativistic dispersion eq. (2.38).
From the Klein-Gordan-equation the continuity equation

$$\partial_\mu j^\mu = \partial_\mu\left(\frac{i}{2m}\left[\psi^*\partial^\mu\psi - (\partial^\mu)^*\psi\right]\right) \tag{2.42}$$

can be derived. The first component of the four-vector current j^μ is the probability density

$$\rho(\mathbf{r}, t) \equiv j^0(\mathbf{r}, t) = \frac{i}{2m}\left[\psi^*\frac{\partial\psi}{\partial t} - \left(\frac{\partial\psi}{\partial t}\right)^*\psi\right] \tag{2.43}$$

which is conserved as in the non-relativistic case, but, in opposite to that, it is not positive definite. Since this cannot be interpreted in the framework of understanding the wave function as an indicator for the probability of position, people tried to find another equation that fits in the relativistic framework but does have a positive definite probability density.

2.4.1. The Dirac Equation

Starting from the Klein-Gordan-equation eq. (2.41) in position space, factorizing yields

$$\begin{aligned} 0 &= \left(\partial_\mu \partial^\mu c^2 + m^2 c^4\right) \psi \\ &= -\left(i\gamma^\mu \partial_\mu c - mc^2\right)\left(i\gamma^\nu \partial_\nu c + mc^2\right) \psi \end{aligned} \tag{2.44}$$

where $\gamma^\mu \partial_\mu \gamma^\nu \partial_\nu = \partial_\mu \partial^\mu = g^{\mu\nu} \partial_\nu \partial_\mu$ is required. Therefore, the four quantities $\gamma^\mu, \mu \in \{0,1,2,3\}$ need to satisfy the relation

$$\frac{1}{2}\left\{\gamma^\mu, \gamma^\nu\right\} = g^{\mu\nu}\,, \tag{2.45}$$

where $\{\cdot,\cdot\}$ is the anti-commutator and $g^{\mu\nu}$ the inverse metric tensor.
To fulfill eq. (2.44) one of the factors needs to be zero, and, choosing the former one, we get

$$\left(\gamma^\mu \hat{p}_\mu c - mc^2\right) \psi = 0\,. \tag{2.46}$$

This equation was first derived by Paul Dirac [19] and hence is called Dirac equation (DE). Separating the spatial and time-like part and multiplying both sides with γ^0 yields

$$i\frac{\partial}{\partial t}\psi = \left(\gamma^0 \boldsymbol{\gamma} \cdot \hat{\mathbf{p}} c + \gamma^0 mc^2\right) \psi\,. \tag{2.47}$$

Using $\alpha_i := \gamma^0 \gamma^i$ and $\beta := \gamma^0$ we retrieve the Schrödinger form of the DE

$$i\frac{\partial}{\partial t}\psi = \underbrace{\left(c\boldsymbol{\alpha} \cdot \hat{\mathbf{p}} + \beta mc^2\right)}_{=:\hat{H}_D} \psi\,, \tag{2.48}$$

where H_D is the so-called *Dirac Hamiltonian.*

We now substitute the kinetic momentum $\hat{p}^\mu$ with the momentum $\hat{p}^\mu - qA^\mu$ in presence of an electromagnetic field, where $A^\mu = (\Phi, \mathbf{A})^\top$ is the four-potential and q the charge of the particle. By setting $\mathbf{A} = 0$, we retrieve the Dirac Hamiltonian in the presence of an electrostatic potential $V(\mathbf{r}) = q\Phi(\mathbf{r})$

$$\hat{H}_D = c\boldsymbol{\alpha} \cdot \hat{\mathbf{p}} + \beta mc^2 + V(\mathbf{r}) \tag{2.49}$$

by using the property $\left(\gamma^0\right)^2 = 1$. This substitution is originated in the construction of a relativistic Lagrangian in presence of an electromagnetic field [20, ch. 12.1]. The detailed derivation of eq. (2.49) can be found in [18].

From the properties eq. (2.45) we see that the γ^μ cannot be scalars. Some square matrices in $\mathbb{C}^{4\times4}$ have these properties, but these matrices are not uniquely defined and also larger matrices fulfill eq. (2.45). Because $\hat{H}_D$ needs to be Hermitian as the energy of a system is an observable, we find that $\boldsymbol{\alpha}$ and β need to be Hermitian. Choosing

$$\boldsymbol{\alpha} = \begin{pmatrix} 0 & \hat{\sigma} \\ \hat{\sigma} & 0 \end{pmatrix} \quad \text{and} \quad \beta = \begin{pmatrix} \mathbb{1}_2 & 0 \\ 0 & -\mathbb{1}_2 \end{pmatrix}, \tag{2.50}$$

where $\hat{\sigma}$ is the vector of Pauli-matrices eq. (2.5), it can be shown that this choice of 4×4 matrices describes a particle with spin $\frac{1}{2}$. From the dimension of the matrices follows that the relativistic wave function ψ must be a vector in $\mathbb{C}^4$.

The results obtained from solving the DE show a better agreement than those calculated with the SE [21]. But the experiments show that the description using the DE is not complete since there are systematic effects like the Lamb shift [22] that cannot be explained using the relativistic formulas. To investigate such further-reaching effects one needs to describe the hydrogenic system using quantum electrodynamics which would go far beyond the scope of this work. An introduction can be found, for example, in [23].

2.4.2. The Dirac-Equation for a Spherical Symmetric Potential

Before searching for a solution for a specific potential some general statements about the eigenfunctions can be derived [10, 18].
Since in hydrogenic systems the potential is spherical symmetric ($V(\mathbf{r}) = V(r)$) it is reasonable to describe the problem in spherical coordinates like in the non-relativistic case. We also try to separate the variables, for which we use the findings from section 2.2. In this case the spin-operator is given by the 4×4 matrix

$$\hat{\mathbf{S}} = \frac{1}{2} \begin{pmatrix} \hat{\sigma} & 0 \\ 0 & \hat{\sigma} \end{pmatrix},$$

and it can be shown for the total angular momentum $\hat{\mathbf{J}} = \hat{\mathbf{L}} + \hat{\mathbf{S}}$ that $\hat{\mathbf{J}}^2$ and $\hat{\mathbf{J}}_z$ commute with $\hat{H}_D$. Therefore, the eigenstates of $\hat{H}_D$ can be classified regarding the energy and the eigenvalues of $\hat{\mathbf{J}}^2$ and $\hat{\mathbf{J}}_z$, and these eigenfunctions are proportional to the eigenfunctions found in 2.2.1, the spherical spinors $\Omega_{\kappa m}(\theta, \varphi)$.

In total analogy to the SE, we use the *ansatz* $\psi(\mathbf{r}, t) = \mathrm{e}^{-iEt/\hbar}\psi(\mathbf{r})$ to retrieve the time-independent DE

$$\left(c\boldsymbol{\alpha} \cdot \hat{\mathbf{p}} + \beta mc^2 + V(r)\right) \psi(\mathbf{r}) = E\psi(\mathbf{r}) \,. \tag{2.51}$$

Seeking an eigenfunction of eq. (2.51) that has the form

$$\psi_{\kappa m}(\mathbf{r}) = \frac{1}{r} \begin{pmatrix} i f_\kappa(r)\Omega_{\kappa m}(\theta, \varphi) \\ g_\kappa(r)\Omega_{-\kappa m}(\theta, \varphi) \end{pmatrix} \tag{2.52}$$

we find after some calculations (see appendix A.2.1) the coupled first-order ordinary differential equations

$$\left(V + mc^2\right) f_\kappa + c\left(\frac{\mathrm{d}}{\mathrm{d}r} - \frac{\kappa}{r}\right) g_\kappa = E f_\kappa \quad \text{and} \tag{2.53a}$$

$$-c\left(\frac{\mathrm{d}}{\mathrm{d}r} + \frac{\kappa}{r}\right) f_\kappa + \left(V - mc^2\right) g_\kappa = E g_\kappa \,. \tag{2.53b}$$

Furthermore, the ψ_κ should hold the normalization condition

$$\int_{\mathbb{R}^3} \psi_\kappa^\dagger(\mathbf{r})\psi_\kappa(\mathbf{r})\mathrm{d}\mathbf{r} = 1 \,, \tag{2.54}$$

which becomes

$$\int_0^\infty \left(f_\kappa^2(r) + g_\kappa^2(r)\right) \mathrm{d}r = 1 \tag{2.55}$$

using the orthonormality of the spherical spinors eq. (2.14).

2.4.3. An Electron in a Coulomb Potential

In this section the analytical solution for the DE for a particle with mass m in the presence of a Coulomb field eq. (2.17) of a nucleus with charge Ze is discussed. In atomic units the Coulomb potential is given by $V(r) = -Z/r$. Only the most important steps are recapped here. A more detailed solution can be taken from many textbooks such as [10, 18].
To find solutions of the eqs. (2.53) we again examine the asymptotic behavior. For large values of $r \to \infty$, eqs. (2.53) become

$$c\frac{\mathrm{d}g_\kappa}{\mathrm{d}r} = \left(E - mc^2\right) f_\kappa \tag{2.56a}$$

$$c\frac{\mathrm{d}f_\kappa}{\mathrm{d}r} = -\left(E + mc^2\right) g_\kappa \tag{2.56b}$$

which can be merged into the second order differential equation

$$c^2\frac{\mathrm{d}^2 f_\kappa}{\mathrm{d}r^2} + \left(E^2 - m^2c^4\right) f_\kappa = 0\,. \tag{2.57}$$

This equation has the independent solutions $\exp(\pm\lambda r)$, where $\lambda = \sqrt{m^2c^2 - E^2/c^2}$, of which only

$$f_\kappa(r)\Big|_{r\to\infty} = \mathrm{e}^{-\lambda r} \tag{2.58}$$

is normalizable and therefore physically reasonable. Using this solution in eq. (2.56b) we find

$$g_\kappa(r)\Big|_{r\to\infty} = \sqrt{\frac{mc^2 - E}{mc^2 + E}}\mathrm{e}^{-\lambda r}\,. \tag{2.59}$$

Using these asymptotic information, we write the radial functions in the form

$$f_\kappa(r) = \sqrt{1 + E/(mc^2)}\mathrm{e}^{-\lambda r}\left(F_1(r) + F_2(r)\right) \quad \text{and} \tag{2.60a}$$

$$g_\kappa(r) = \sqrt{1 - E/(mc^2)}\mathrm{e}^{-\lambda r}\left(F_1(r) - F_2(r)\right) \tag{2.60b}$$

and, by substituting this *ansatz* into eqs. (2.53), we find for the functions F_1 and F_2

$$\frac{\mathrm{d}F_1}{\mathrm{d}x} = \frac{EZ}{c^2\lambda x}F_1(x) + \left(\frac{Zm}{\lambda x} - \frac{\kappa}{x}\right)F_2 \quad \text{and} \tag{2.61a}$$

$$\frac{\mathrm{d}F_2}{\mathrm{d}x} = -\left(\frac{Zm}{\lambda x} + \frac{\kappa}{x}\right)F_1 + \left(1 - \frac{EZ}{c^2\lambda x}\right)F_2\,, \tag{2.61b}$$

where $x = 2\lambda r$. Using the *ansatz* $F_{1/2} = a_{1/2} x^\gamma$ in eqs. (2.61) in the limiting case $x \to 0$, we find

$$\frac{a_2}{a_1} = \frac{\gamma - \frac{EZ}{c^2\lambda}}{-\kappa + Zm/\lambda} = \frac{-\kappa - Zm/\lambda}{\gamma + \frac{EZ}{c^2\lambda}} . \tag{2.62}$$

Evaluating the right-hand equality yields $\gamma^2 = \kappa^2 - (\alpha Z)^2$, where we used $c^{-1} = \alpha$ in atomic units. Normalizable solutions are only acquired by positive values for γ, i.e. $\gamma = \sqrt{\kappa^2 - \alpha^2 Z^2}$. Combining both eqs. (2.61) by rearranging the latter one to express F_2 in terms of F_1 yields

$$F_2(x) = \frac{1}{-\kappa + \frac{Zm}{\lambda}} \left(x \frac{\mathrm{d}F_1}{\mathrm{d}x} - \frac{EZ}{c^2\lambda} F_1 \right) , \tag{2.63}$$

and, using this in the first equation, we find

$$x \frac{\mathrm{d}^2 F_1}{\mathrm{d}x^2} + (1 - x) \frac{\mathrm{d}F_1}{\mathrm{d}x} - \left(\frac{\gamma^2}{x} - \frac{EZ}{c^2\lambda} \right) F_1 . \tag{2.64}$$

By using the *ansatz* $F_1(x) = x^\gamma \tilde{F}(x)$, we find that $\tilde{F}(x)$ satisfies the Kummer's equation

$$x \frac{\mathrm{d}^2 \tilde{F}}{\mathrm{d}x^2} + (b - x) \frac{\mathrm{d}\tilde{F}}{\mathrm{d}x} - a\tilde{F} = 0 , \tag{2.65}$$

where $a = \gamma - EZ/(c^2\lambda)$ and $b = 2\gamma + 1$. Since this is the same equation we found in the non-relativistic case (c.f. eq. (2.31), but with different parameters a, b), we know that the solutions $\tilde{F}(x) \equiv F(a, b, x)$ that are regular at the origin are the confluent hypergeometric functions eq. (2.32), and therefore

$$F_1(x) = x^\gamma F(a, b, x) . \tag{2.66}$$

Using this result in eq. (2.63), we get for F_2,

$$F_2(x) = \frac{x^\gamma}{-\kappa + \frac{Zm}{\lambda}} \left(x \frac{\mathrm{d}F}{\mathrm{d}x} + aF \right) = \frac{\gamma - \frac{EZ}{c^2\lambda}}{-\kappa + \frac{Zm}{\lambda}} x^\gamma F(a + 1, b, x) , \tag{2.67}$$

and combining those findings with the *ansatz* (2.60) yields for the radial parts of the Dirac wave function

$$f_\kappa(r) = \sqrt{1 + \frac{E}{mc^2}} \mathrm{e}^{-x/2} x^\gamma \left[\left(-\kappa + \frac{Zm}{\lambda} \right) F(a, b, x) + \left(\gamma - \frac{EZ}{c^2\lambda} \right) F(a + 1, b, x) \right] , \tag{2.68a}$$

$$g_\kappa(r) = \sqrt{1 - \frac{E}{mc^2}} \mathrm{e}^{-x/2} x^\gamma \left[\left(-\kappa + \frac{Zm}{\lambda} \right) F(a, b, x) - \left(\gamma - \frac{EZ}{c^2\lambda} \right) F(a + 1, b, x) \right] , \tag{2.68b}$$

where $x = 2\lambda r$. To find the normalization factor for these functions, we first need to have a look at the behavior of these functions. Since the confluent hypergeometric functions are normalizable only if they are polynomials, we need to require $\mathbb{N} \ni a \leq 0$, analogue to

the non-relativistic case. Defining the principal quantum number n as $n = k + n_r$, where $k = |\kappa| = j + \frac{1}{2}$ and the radial quantum number is defined like in the non-relativistic case as $n_r := -a$, we find from the value of a

$$\left.\begin{aligned} a &= \gamma - \frac{EZ}{c^2\lambda} \\ n &= k + n_r = k - a \end{aligned}\right\} \implies \frac{EZ}{c^2\lambda} = \gamma + n - k\,. \tag{2.69}$$

This finding is actually obtained from solving the time-independent DE eq. (2.51), which is an eigenvalue equation for the energy, using the radial functions (2.68), but since this requires a longer calculation, the above heuristic, which is also correct since we derived a from eq. (2.51), is given here to motivate this result. But from this calculation the case $a = -n_r = 0$ cannot be derived, where eq. (2.69) also holds. In this case one finds, from solving the eigenvalue problem, $k = Zm/\lambda$. From this follows that the factors $-\kappa + Zm/\lambda$ and $\gamma - EZ/(c^2\lambda)$ in eqs. (2.68) vanish for $\kappa = k > 0$. Therefore, states with $n_r = 0$ occur only for $\kappa < 0$, and for a given value of $n > 0$ there are $2n - 1$ possible eigenfunctions as there are n eigenfunctions with $\kappa = -n, -n+1, \ldots, -1$ and $n - 1$ eigenfunctions with $\kappa = 1, 2, \ldots, n - 1$.
Solving eq. (2.69), we obtain for the Dirac energy eigenvalue

$$E_{n\kappa} = \frac{mc^2}{\sqrt{1 + \frac{\alpha^2 Z^2}{(\gamma + n - k)^2}}}\,. \tag{2.70}$$

Note that the energy levels for a given n only depend on $|\kappa| = k$ and therefore are degenerate for the same value of j but differing values of l. The levels $2s_{1/2}$ and $2p_{1/2}$ for example have the same energy, but the energy value of the $2p_{1/2}$ and $2p_{3/2}$ levels differ in contrast to the non-relativistic case. This energy splitting between such two levels with the same l but different j is called *fine-structure* splitting. A qualitative visualization is shown in fig. 2.2.

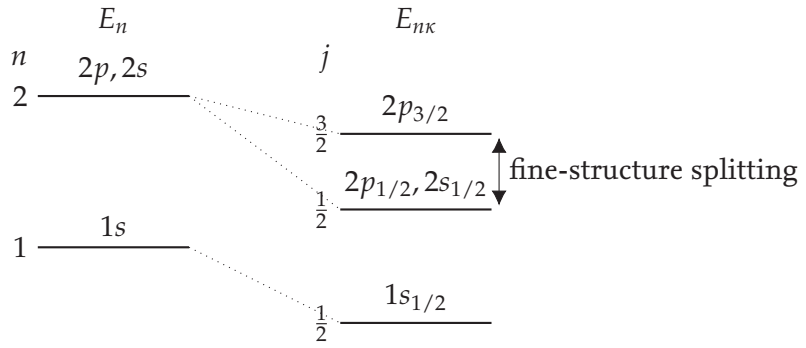

Figure 2.2.: Schematic comparison of the energy levels E_n of non-relativistic states and $E_{n\kappa}$ of relativistic states. The relativistic states exhibit the fine-structure splitting.

We still need to normalize the radial functions. For this, we introduce the generalization of the principal quantum number $N = \frac{Zm}{\lambda} = (\gamma + n - k)\,\frac{mc^2}{E}$, which does not have to be

integer. Applying eq. (2.70) we find

$$N = \sqrt{n^2 - 2(n-k)(k-\gamma)}\,, \tag{2.71}$$

which yields $N = n$ for $n = k$. Using N we can write the coefficients of the confluent hypergeometric functions in eqs. (2.68) as

$$-\kappa + \frac{Zm}{\lambda} = N - k \quad \text{and} \tag{2.72a}$$

$$\gamma - \frac{EZ}{c^2\lambda} = -(n-k)\,. \tag{2.72b}$$

The normalization factor for the radial functions is given by

$$N_{n\kappa} = \frac{1}{N\Gamma(2\gamma+1)}\sqrt{\frac{Zm\Gamma(2\gamma+1+n-k)}{2(n-k)!(N-k)}}\,, \tag{2.73}$$

where $\Gamma(\cdot)$ is the gamma-function. The radial Dirac wave functions in the presence of a Coulomb field are therefore given by

$$f_{n\kappa}(r) = \sqrt{1+\frac{E_{n\kappa}}{mc^2}}N_{n\kappa}\mathrm{e}^{-x/2}x^{\gamma}\Big[(N-\kappa)F(-n+k,2\gamma+1,x) - (n-k)F(-n+k+1,2\gamma+1,x)\Big] \quad \text{and} \tag{2.74a}$$

$$g_{n\kappa}(r) = \sqrt{1-\frac{E_{n\kappa}}{mc^2}}N_{n\kappa}\mathrm{e}^{-x/2}x^{\gamma}\Big[(N-\kappa)F(-n+k,2\gamma+1,x) + (n-k)F(-n+k+1,2\gamma+1,x)\Big]\,, \tag{2.74b}$$

and, using the coefficient given in eq. (2.73), they satisfy the normalization condition eq. (2.55). Comparing the ratio of the scale factors of the radial functions yields

$$\frac{\sqrt{1+\frac{E_\kappa}{mc^2}}}{\sqrt{1-\frac{E_{n\kappa}}{mc^2}}} \approx \frac{\alpha Z}{2n}\,, \tag{2.75}$$

which means for $Z = 1$ that $f_{n\kappa}$ is about two orders of magnitude larger than $g_{n\kappa}$. Hence, $f_{n\kappa}$ is called the large and $g_{n\kappa}$ the small component of the radial Dirac Coulomb wave function.

The shape of the large component $f_{n\kappa}$ is very similar to the non-relativistic Coulomb wave function. In the non-relativistic limit $E_{nk} \to mc^2 - \frac{mc^2(Z\alpha)^2}{2n^2}$ they are actually the same. Therefore, the number of nodes for the large component $f_{n\kappa}$ is also given by $n-l-1$, which is the same for the small component $g_{n\kappa}$ when $\kappa < 0$. For $\kappa > 0$ it is given by $n-l$. This can be seen in fig. 2.3, where the radial components of the states $3p_{3/2}$, $3p_{1/2}$ and $5d_{5/2}$ are plotted. One can see that the large component f of the $3p_x$ states is, except for the sign,

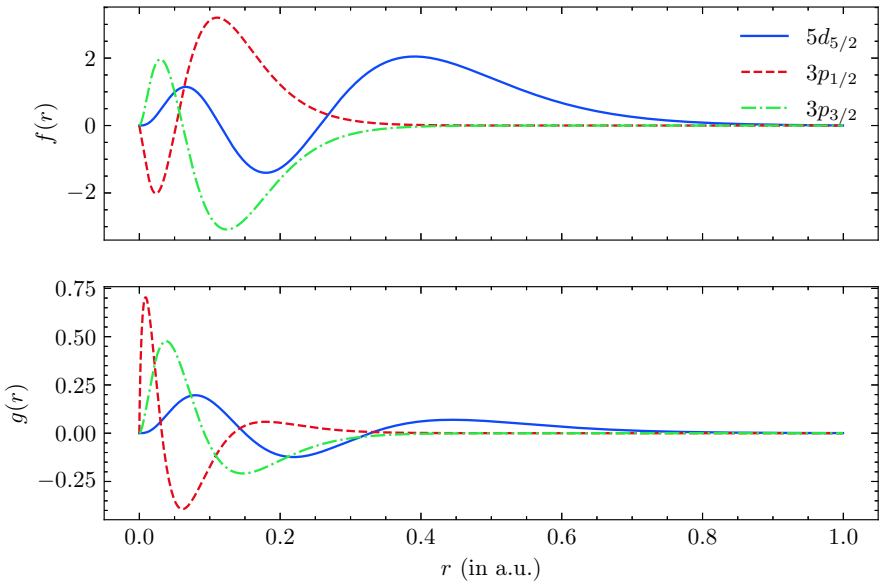

Figure 2.3.: Radial components f and g of the states $3p_{1/2}$, $3p_{3/2}$ and $5d_{5/2}$ for hydrogenic uranium.

very similar while there is no such similarity in the small components as the number of nodes for these components differs by one. By comparing the large components of $3p_{3/2}$ and $5d_{5/2}$ with their non-relativistic counterparts in fig. 2.1 their similarity is apparent.

To see the differences between the relativistic solutions and the non-relativistic ones, in fig. 2.4 the radial density distributions are shown for the relativistic $3p_x$ wave functions and the non-relativistic $3p$ wave function. The radial density distribution is given by

$$D_{nl}(r) = |R_{nl}(r)|^2 \tag{2.76}$$

for solutions of the SE and

$$D_{n\kappa}(r) = |f_{n\kappa}(r)|^2 + |g_{n\kappa}(r)|^2 \tag{2.77}$$

for solutions of the DE. In the figure, it can be seen that both relativistic states' extrema are closer to the origin than the non-relativistic counterpart and, indeed, it can be shown that for the scaled mean radii the inequality

$$\langle n\kappa|2Zr|n\kappa\rangle < \langle nl|2Zr|nl\rangle \tag{2.78}$$

holds. This corresponds to the inequality for the energies

$$E_{n,\kappa=l} < E_{n,\kappa=-l-1} < E_{nl} = E_nk \tag{2.79}$$

resulting from the energy for the Dirac states eq. (2.70) and the energy for the SE solutions eq. (2.33). From this, it can be seen that Dirac electrons are more tightly bound. The first inequality eq. (2.78) is the above discussed fine-structure splitting that is apparent in the

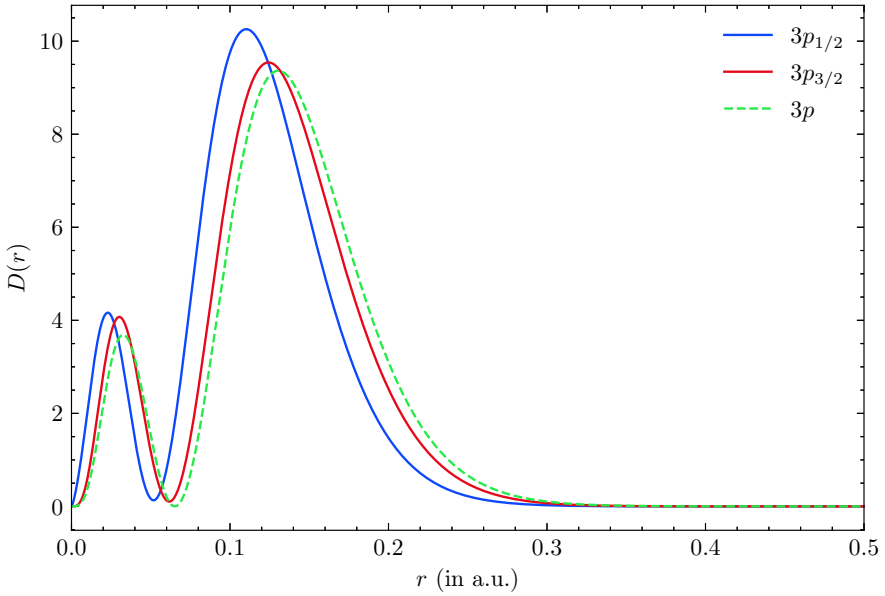

Figure 2.4.: Probability density for the relativistic states $3p_{3/2}$, $3p_{1/2}$ and non-relativistic state $3p$ for hydrogen-like uranium.

wave functions as the radial density distribution of the $3p_{1/2}$ state is more compact (confined closer to the origin) as the one of the $3p_{3/2}$ state.
In analogy to the non-relativistic case the equations are valid for normal hydrogenic atoms where the particle in the field of the nucleus is an electron with mass $m = 1$ as well as for exotic atoms where the particle is a muon with mass $m \approx 200$. On the contrary, there is no easy way to accommodate finite nuclear mass that results in nuclear recoil. This effect, which results in small shifts in the energy, is addressed in quantum electrodynamics. As this is beyond the limits of this thesis, all relativistic calculations will be performed just for an infinite mass nucleus. For nuclear recoil corrections for point-like nuclei see [24, 25] and for finite size nuclei see [26].

2.5. Potential of the Nucleus

To find the solutions for both, the SE and the DE, the nuclear potential of the hydrogenic system was modeled using a Coulomb-potential $V_C(r) = -Z/r$. This is the potential of a point-like nucleus. Its charge density is given by $\rho_C(r) = \delta(r) \cdot Z$, where $\delta(r)$ is the Dirac delta distribution. Since experiments showed that the nucleus has a finite size which changes the form of the potential, other models are better suited to describe it. In `dish` two other common models are implemented: A simple approximation for a finite nucleus is to describe it as a homogeneously charged sphere with radius R_0, i.e. its charge density

is

$$\rho_B(r) = \begin{cases} \rho_0 & , r \leq R_0 \,, \\ 0 & , t > R_0 \,, \end{cases} \quad \text{where} \quad \rho_0 = \frac{Z}{4\pi \int_0^{R_0} r^2 \mathrm{d}r} \tag{2.80}$$

and the resulting potential for a negatively charged particle is given by [27]

$$V_B(r) = \begin{cases} -\frac{Z}{R_0}\left(\frac{3}{2} - \frac{r^2}{2R_0^2}\right) & , r < R_0 \,, \\ -\frac{Z}{r} & , r \geq R_0 \,. \end{cases} \tag{2.81}$$

To describe the nucleus without sharp edges the charge density can be modeled using a Fermi-distribution [28] as

$$\rho_F(r) = \frac{\rho_0}{1 + \exp\left(\frac{r-c}{a}\right)} \,, \tag{2.82}$$

where c is the 50% fall-off radius and the parameter a, which describes the diffuseness of the edge, is related to the 90% − 10% fall-off distance t by $a = t/(4\ln(3)t)$. A good approximation for most nuclei is given by $t = 2.3\,\mathrm{fm}$ [29]. For vanishing diffuseness $a \to 0$ the Fermi model reduces to the uniform distribution with $R_0 = c$. The normalization constant ρ_0, determined by the requirement

$$Z = 4\pi\rho_0 \int_0^\infty \frac{r^2}{1 + \exp\left(\frac{r-c}{a}\right)} \mathrm{d}r \,,$$

is given as

$$\rho_0 = \frac{3}{4\pi c^3} \frac{Z}{\mathcal{N}} \,, \quad \text{where} \quad \mathcal{N} = 1 + \frac{a^2}{c^2}\pi^2 + \frac{6a^3}{c^3} S_3 \,,$$

with S_3 from

$$S_k = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^k} \exp\left(-n \cdot \frac{c}{a}\right) .$$

The potential of such a charge density distribution is

$$V_F(r) = \begin{cases} \frac{Z}{c\mathcal{N}}\left(\frac{3}{2} - \frac{r^2}{2c^2} + \frac{a^2\pi^2}{2c^2} + \frac{3a^2}{c^2} P_2 + \frac{6a^3}{c^2 r}\left[S_3 - P_3\right]\right) & , r < c \,, \\ \frac{Z}{\mathcal{N}r}\left(1 + \frac{a^2\pi^2}{c^2} - 3\frac{a^2 r}{c^3} P_2 + 6\frac{a^3}{c^3}\left[S_3 - P_3\right]\right) & , r > c \,, \end{cases} \tag{2.83}$$

where

$$P_k = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^k} \exp\left(-n\frac{|r-c|}{a}\right) .$$

In fig. 2.5 the three charge distributions and their respective potentials are shown. One can see that for large distances the potentials for finite size models agree with the Coulomb potential and for small r have a finite value in opposite to the singularity of a point-like nucleus. In the upper plot additionally the root-mean-square radius of the charge densities is marked, which is given by

$$R_{\mathrm{rms},B} = \sqrt{\frac{3}{5}} \cdot R_0 \tag{2.84}$$

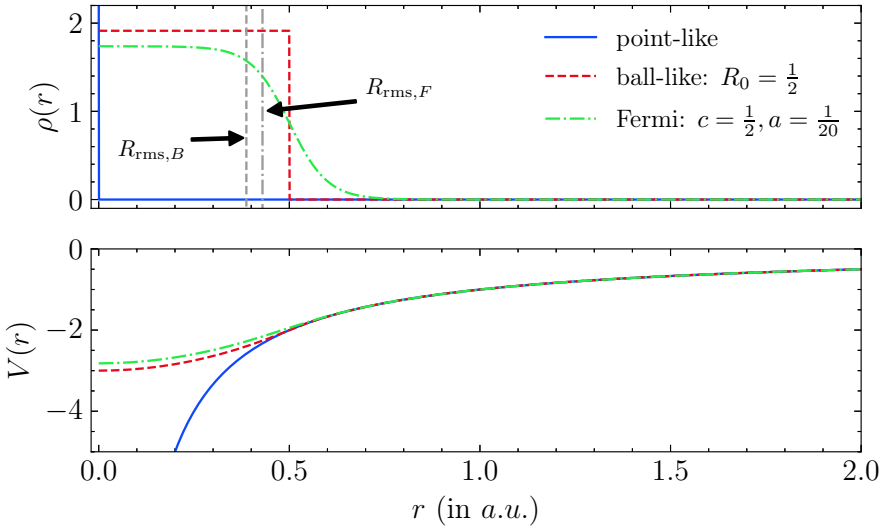

Figure 2.5.: Charge density and electrostatic potential of a nucleus modeled by the three given charge distributions point-like, sphere-like and Fermi-like.

for the sphere-like distribution. For the Fermi distribution a good approximation[2] is given by [30]

$$R_{\mathrm{rms},F} \approx \sqrt{\frac{3}{5}c^2 + \frac{7}{5}\pi^2 a^2}\,. \tag{2.85}$$

Since the R_{rms} for the uniformly charged sphere is closer to the origin in comparison with the Fermi distribution the absolute value of the potential is also larger. You can easily extend `dish` by adding an arbitrary[3] central potential as shown in chapter 5.2.2 to find solutions for this potential.

[2] In chapter A.2.2 is shown how $R_{\mathrm{rms},F}$ is calculated numerically.

[3] Arbitrary, but physical meaningful as the asymptotics for $r \to \infty$ should be the same as for a Coulomb potential.

3. Numerical Methods and Their Implementation

To both, the Schrödinger and the Dirac equation, analytical solutions are just known for a few specific central potentials, e.g. the Coulomb potential which was discussed in the last chapter. Since the Coulomb potential describes the electrostatic potential of a point charge, and we know from experiments that the nucleus of atoms has a finite size, other potentials are used to model the potential more accurate [31–34]. For these potentials no analytical solutions are known, and we need to use numerical methods to obtain these solutions. But since we were able to separate both the SE and the DE into spherical and radial parts under the only premise that the potential is spherical symmetric, the spherical parts are still known analytically, and we only need to find the radial parts numerically. While there is a multitude of numerical methods to solve systems of ordinary differential equations, a lot of them refined for specific use cases, we will focus here on a single method to offer a good start into such numerical calculations. In this thesis we will follow the approach described by Walter R. Johnson in [10], and while he includes an implementation in Fortran 77 with his supplementary material, for the `dish` package which is developed along this thesis, it is reimplemented in Python. In this chapter the associated parts of the code will be referenced and some implementation details are discussed.

3.1. The General Procedure

The goal is to find the solutions of the radial SE eq. (2.24b) and the radial DE eq. (2.53). To do so we will discretize the space from $r = 0$ to a point a_∞ that is considered to be a practical infinity as the wave function is close to zero in this region far away from the origin. In this outer region there are two solutions to the SE and the DE, which behave like $\exp(\pm\lambda r)$, from which only $\exp(-\lambda r)$ is physically acceptable as discussed in the last chapter. Nonetheless, small numerical errors introduce admixtures of the other, so-called complementary, solution which will propagate during the solving process. To minimize these related errors we start in this region at the most outer points and integrate inwards as the complementary solution decreases in this direction. On the other hand there are also complementary solutions to both the SE and the DE near the origin which behave like r^{-l} instead of r^{l+1} and $x^{-\gamma}$ instead of x^{γ} respectively. To minimize the error here we start at the origin with regular solutions and integrate outwards. The integration is carried out using a point-by-point scheme, which will be discussed section 3.3, unto the outer *classical turning point*, meaning the maximum distance that the particle with given energy

can reach from a classical point of view. Analogue we integrate from the outer region inwards until this point, starting with an approximation obtained from the asymptotic behavior of the regular solution. The starting values and the integration depend on the energy E that is initially guessed from the analytical solution E_C for a Coulomb potential. At the classical turning point the solutions of the radial Schrödinger wave function and the large component respectively are matched to be continuous. After verifying whether these functions are having the correct number of nodes, it is checked if in the case of the SE the derivative of the wave function and in the case of the DE the small component are also continuous. If this is the case we found the aspired solution consisting of the eigenfunction and the eigenenergy. If not we adjust the energy and repeat the process as shown in fig. 3.1.

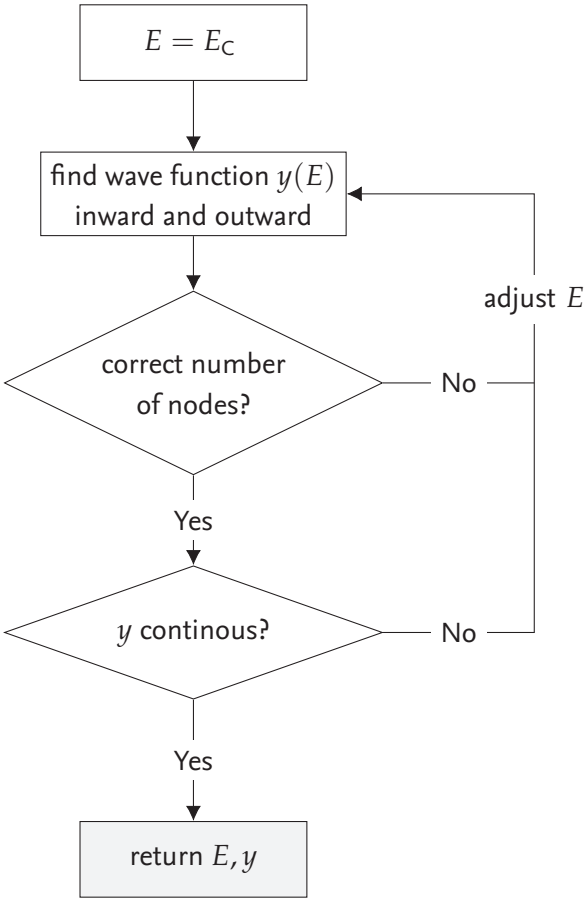

Figure 3.1.: Simplified flow chart of the solving process used in `dish`.

Here we will first cover some general points and the finite difference method. Afterward we will discuss how to find solutions for the radial function for the SE, and finally we will apply our developed methods on the radial functions of the DE.

3.2. The Grid

Near the origin the value of the wave function changes rapidly as the wave function oscillated in the classical allowed region. Beyond the classical turning point there is no such rapid change anymore as the wave functions absolute value gradually falls to zero. Taking into account this behavior, we choose a non-uniform grid which spacing is fine for small values of r and more loose for larger r. There are a lot of possible choices that fulfill these requirements, but we chose one that has proven both, its convenience and flexibility [1, 10, 35].
We use a grid

$$r[i] = r_0 \left(\mathrm{e}^{t[i]} - 1 \right) , \quad \text{with} \tag{3.1a}$$

$$t[i] = i \cdot h, \quad i = 0, 1, \ldots, N-1 , \tag{3.1b}$$

where the grid $r[i]$ fulfills the above-mentioned characteristics while we also have an equidistant grid $t[i]$ which will be beneficial, as will be shown later. To convert between these two grids we will need the derivative

$$\frac{\mathrm{d}r}{\mathrm{d}t}[i] = r_0 \cdot \mathrm{e}^{t[i]} . \tag{3.2}$$

Finding a good choice of parameters r_0, h and N will be discussed in the next chapter along some applications and results. The notation $f[i]$ for an arbitrary function $f : \mathbb{R} \to \mathbb{R}^x$ means that the function is evaluated on the i^{th} grid point. We will use this notation to distinguish the discrete version from the continuous function $f(r)$ or $f(r(t))$.
In `dish` the grid is implemented in `dish.util.radial.grid.grid` in the `DistanceGrid`-class.

3.3. Adams-Moulton Procedure

We want to solve a first order ordinary differential equation of the form

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y) \tag{3.3}$$

on a finite grid $t[i]$, where $m \in \mathbb{N}$, $\Omega \subset \mathbb{R} \times \mathbb{R}^m$, $f : \Omega \to \mathbb{R}^m$, at the moment is an arbitrary function. We seek a solution $y : \mathbb{R} \to \mathbb{R}^m, y \in \mathcal{C}^1$ that fulfills eq. (3.3). In this thesis everything is discussed in $\mathbb{R}$ as the problem we want to tackle is real-valued but can be done analogously for complex-valued functions.
From eq. (3.3) we can see, that if we know the value of $y[n]$ at one grid point $t[n]$, we could find the value at the next point $t[n+1]$ by solving the integral

$$y[n+1] = y[n] + \int_{t[n]}^{t[n+1]} f(t, y(t)) \mathrm{d}t . \tag{3.4}$$

Since the analytic solution of this integral might not exist or is (computationally) hard to evaluate, we want to find another way to carry out the integration numerically. For this we will use an *Adams-Moulton method*. To understand this method, we need to reference some basics of the calculus for finite difference operators. A more extensive introduction into these operators can be found in [36, ch. 3.3]. The rest of the introduction into *Adams*-procedures is based on [10] and [37, ch. 8.3].
We define the backward difference operator as

$$\nabla f[n] = f[n] - f[n-1]\,, \tag{3.5}$$

and rewrite it to $(1-\nabla)f[n] = f[n-1]$. Inverting this expression and applying it multiple times yields

$$f[n+x] = (1-\nabla)^{-1} f[n+x-1] = \ldots = (1-\nabla)^{-x} f[n]\,. \tag{3.6}$$

This expression interpolates equally spaced points in general. The expression $(1-\nabla)^{-x}$ needs to be understood as a series expansion in ∇

$$f[n+x] = \sum_{k=0}^{\infty} \frac{\prod_{m=1}^{k}(x+m-1)}{k!} \nabla^k f[n] \tag{3.7}$$

and ∇^k means an iterative appliance of eq. (3.5) which yields

$$\nabla^k f[n] = \sum_{j=0}^{k} (-1)^j \binom{k}{j} f[n-j]\,. \tag{3.8}$$

Combining eq. (3.7) and eq. (3.8) and truncating the expansion after the k^{th} term we get a polynomial in x of degree k that interpolates the points $(t[n-k], f[n-k]), \ldots, (t[n], f[n])$. We can then integrate this polynomial analytically to approximate the integral in eq. (3.4) which yields

$$y[n+1] = y[n] - \frac{h\nabla}{(1-\nabla)\log(1-\nabla)}\,. \tag{3.9}$$

This extrapolation procedure is called an *Adams-Bashford*-method (AB-method).
Using eq. (3.6) for $x = 1$ we can rewrite this formula as an interpolation procedure

$$y[n+1] = y[n] - \frac{h\nabla}{\log(1-\nabla)} f[n+1]\,, \tag{3.10}$$

which is called an *Adams-Moulton*-method (AM-method). Expanding eq. (3.10) into a series in ∇ and using eq. (3.8) we get

$$y[n+1] = y[n] + h\sum_{j=0}^{\infty} c_j \nabla^j f[n+1] = y[n] + h\sum_{j=0}^{\infty} \tilde{a}[j] f[n+1-j]\,. \tag{3.11}$$

Truncating after the $(k-1)^{\text{th}}$ order in ∇ we can write this as

$$y[n+1] = y[n] + \frac{h}{D}\sum_{j=1}^{k+1} a[j] f[n-k+j]\,, \tag{3.12}$$

where D is the largest positive integer so that the coefficients $a[j]$ are integer and that $a[j]/D = \tilde{a}[k+1-j]$. This is the k-point or $(k+1)^{\text{th}}$-order AM method. The notation $a[i]$ for the coefficients here is used even though these are discrete values and there is no underlying continuous function. It is adopted to show the connection to the values of f at the grid points.

While the error for both methods, AB and AM, is $\mathcal{O}(h^{k+1}y^{(k+1)}[n])$, it is by an order of magnitude smaller for the AM methods as here interpolation is used instead of extrapolation. Usually one uses AB to predict $y[n+1]$, and then takes this value to find $f[n+1]$ using AM. This procedure is therefore called a predictor-corrector method. In the case of a linear function f one can skip the extrapolation step. This is done by writing f using the transformation matrix $G \in \mathbb{R}^{m\times m}$ as $f(t,y) = G(t)\,y$. We then can write eq. (3.12) as

$$\underbrace{\left(1 - \frac{ha[k+1]}{D}G[n+1]\right)}_{=:M[n+1]} y[n+1] = y[n] + \frac{h}{D}\sum_{j=1}^{k} a[j]f[n-k+j]\,, \tag{3.13a}$$

and, if $M[n+1]$ is non-singular, multiply by its inverse

$$y[n+1] = M^{-1}[n+1]\left(y[n] + \frac{h}{D}\sum_{j=1}^{k} a[j]f[n-k+j]\right)\,, \tag{3.13b}$$

which we can use directly to find the next point in one step while maintaining the better accuracy of the AM method.
As it can be seen from eq. (3.13b) the last k values of f are required to find $y[n+1]$ and therefore also k values are required in the beginning that need to be derived somehow else. How these values are inferred will be covered in sections 3.5 and 3.6 regarding the SE and DE respectively.

The AM method is implemented in `dish.util.numeric.adams` module. The function

```
from dish.util.numeric.adams import adams
y_result = adams(j:int, direction: str, y_start: np.ndarray,
                 G: np.ndarray, h:float)
```

performs the $(\mathtt{k}+1)^{\text{th}}$ AM method as described above, where `h` is the integration step width, `y_start` are the starting values and the dimensions of `G` determine how many steps are inferred. By passing "`in`" or "`out`" to the `direction` argument it is determined whether the integration is performed inward or outward, which results essentially in reordering the result in the inward case. The coefficients $a[j]$ and the divisor D from eq. (3.12) are calculated like eq. (3.11) by symbolic Taylor series expansion using the computer algebra package `sympy`. Since this is computationally very expensive and hence slow, it is only calculated once if a new order is requested and then stored in a lookup table (LUT) in the `dish.util.LUTs.adams_moulton` module which is then accessed by the `adams` function. There are potentially many points on which the wave function should be evaluated and,

therefore, many iterations where eq. (3.13b) needs to be calculated. Thus, this is a time critical part, and an efficient implementation is essential. The first optimization is that $M^{-1}[n+1]$ is not actually calculated, but in reality the linear system of m equations eq. (3.13a) is solved by performing a LU-decomposition which is approximately three times less work than computing the inverse directly while also having greater accuracy [38, ch. 5.3]. As a second step `dish` ships with a Fortran implementation of the AM (see *dish/util/numeric/adams_f.f90*) which offers a speedup of one to two orders of magnitude for a number of steps in the order of 10^3 to 10^6 as can be seen later in the results in chapter 4.1.1. This requires a Fortran compiler while installing `dish` (see chapter 5.1), and it is highly recommended to use the Fortran version because of the immense speed improvements, but to calculate a small number of wave functions on a grid of a few hundred or thousand points is perfectly fine using the pure Python implementation as this takes way less than one second (see again chapter 4.1.1). If the Fortran code is compiled in the installation process, it will be automatically detected and used.

3.4. Lagrangian Differentiation Formulas

The *Lagrangian differentiation formulas* will be used to obtain the initial values for the outward integration that are required for the AM scheme. The formulas are found from the interpolation formula for finite differences eq. (3.6) by differentiation and substituting $x \to -j$, which yields

$$\frac{\mathrm{d}y}{\mathrm{d}x}[n-j] = -\log(1-\nabla)(1-\nabla)^j y[n]\,. \tag{3.14}$$

Expanding this equation in ∇ and keeping the first k terms, we gain the $k+1$ point Lagrangian differentiation formulas by evaluating it at the $k+1$ points $j = 0, 1, \ldots, k$. If we evaluate these formulas for a differentiable function y on the linear grid $t[i]$, these $k+1$ equations can be written as

$$\frac{\mathrm{d}y}{\mathrm{d}t}[i] = \sum_{j=0}^{k} m[i,j]y[j], \quad i = 0, 1, \ldots, k\,, \tag{3.15}$$

where the coefficients $m[i,j]$ are determined from the expansion of eq. (3.14). For $y \in \mathcal{C}^{k+1}(\mathbb{R}, \mathbb{R})$ the error of these formulas is $\mathcal{O}(h^k y^{(k+1)})$.
We will use these formulas to approximate the derivative of the first k points by expressing the derivative through the values itself. The coefficients $m[i,j]$ in eq. (3.15) are calculated in `dish.util.numeric.lagrangian_differentiation`. Because the analytic expansion is again computationally expensive like for the AM coefficients, here a LUT is used to store values once calculated.

3.5. Solutions to the Schrödinger Equation

In this section it will be discussed how the AM method can be used to solve the radial SE

$$\frac{\mathrm{d}^2R}{\mathrm{d}r^2} + 2\left(\mu(E - V(r)) - \frac{l(l+1)}{2r^2}\right) R = 0 \tag{3.16}$$

(cf. eq. (2.24b), $m \to \mu$) in order to find the radial function of solutions to the SE. First we transform this second order ODE into a system of first order equations by substituting

$$\frac{\mathrm{d}R}{\mathrm{d}r} = Q(r)\,, \tag{3.17a}$$

which yields

$$\frac{\mathrm{d}Q}{\mathrm{d}r} = -2\left(\mu(E - V(r)) - \frac{l(l+1)}{2r^2}\right) R(r) \tag{3.17b}$$

as the second equation. Since the AM procedure works on an equidistant grid, we refer the radial function to the linear grid $t[i]$ and find our system of equations which is in the required form eq. (3.3)

$$y(t) = \begin{bmatrix} R(r(t)) \\ Q(r(t)) \end{bmatrix}, \quad \text{and} \tag{3.18a}$$

$$f(t,y) = \frac{\mathrm{d}r}{\mathrm{d}t}\begin{bmatrix} Q(r(t)) \\ -2\left(\mu(E - V(r)) - \frac{l(l+1)}{2r^2}\right) R(r(t)) \end{bmatrix}. \tag{3.18b}$$

From this we can read

$$G(t) = \begin{bmatrix} 0 & b(t) \\ c(t) & 0 \end{bmatrix}, \tag{3.19a}$$

where the matrix elements are given by

$$b(t) = \frac{\mathrm{d}r}{\mathrm{d}t} \quad \text{and} \quad c(t) = -2\frac{\mathrm{d}r}{\mathrm{d}t}\left(\mu(E - V(r)) - \frac{l(l+1)}{2r^2}\right) \tag{3.19b}$$

to write $f(t,y) = G(t)\,y(t)$.
To start the integration using the AM method, we need to find the starting values for the inward and outward integration respectively. To obtain the k initial values for the outward integration, we first factor out r^{l+1} from the radial wave function, gaining $R(t) = r^{l+1}R'(t)$ which yields

$$\frac{\mathrm{d}R'}{\mathrm{d}t} = \frac{\mathrm{d}r}{\mathrm{d}t}Q'(t) \quad \text{and} \tag{3.20a}$$

$$\frac{\mathrm{d}Q'}{\mathrm{d}t} = -2\frac{\mathrm{d}r}{\mathrm{d}t}\left(\mu(E - V(r))R'(t) + \left(\frac{l+1}{r}\right) Q'(t)\right), \tag{3.20b}$$

for eqs. (3.18), and use the above discussed Lagrangian differentiation formulas to obtain a system of $2k$ equations

$$\sum_{j=1}^{k} m[i,j]R'[j] - \frac{\mathrm{d}r}{\mathrm{d}t}[i]Q'[i] = -m[i,0]R'[0]\,, \tag{3.21a}$$

$$\sum_{j=1}^{k} m[i,j]Q'[j] + \left(2\frac{\mathrm{d}r}{\mathrm{d}t}\mu(E-V(r))\right)[i]R'[i] + \left(2\frac{\mathrm{d}r}{\mathrm{d}t}\left(\frac{l+1}{r}\right)\right)[i]Q'[i] = -m[i,0]Q'[0] \tag{3.21b}$$

for the $2k$ variables $R'[i], Q'[i], i = 1,\ldots,k$. $R'[0]$ and $Q'[0]$ are initial values which we derive from the following considerations: Assuming for $r \to 0$ the potential $V(r)$ is dominated by the nuclear Coulomb potential $-Z/r$, from eq. (3.20b) follows

$$\frac{Q'[0]}{R'[0]} = -\frac{Z\mu}{l+1}\,. \tag{3.22}$$

We choose $R'[0] = 1$ arbitrarily since we will rescale our solution later anyway, and with that we can solve the linear system (3.21) using algebraic methods. The k initial values are then obtained from

$$R[i] = r^{l+1}[i]R'[i] \quad \text{and} \tag{3.23a}$$

$$Q[i] = r^{l+1}[i]\left(Q'[i] + \frac{l+1}{r[i]}R'[i]\right)\,. \tag{3.23b}$$

There are other approaches to find those k initial values like for example using a Runge-Kutta scheme[37, ch.8.2], but this requires knowledge of the potential's form in between the grid points. If the potential is not known analytically this requires interpolation. To avoid unnecessary interpolation while maintaining a high accuracy we chose the above described procedure which in `dish` is implemented in `dish.schrodinger.outsch`.

To find the most outer k values to start the inward integration, we assume that the potential behaves asymptotically like as $V(r \to a_\infty) = -\zeta/r$ where ζ is the charge of the system when one electron is removed. Then eqs. (3.18), which are equivalent to the radial SE eq. (3.16)), become

$$\frac{\mathrm{d}R}{\mathrm{d}r} = Q(r) \quad \text{and} \tag{3.24a}$$

$$\frac{\mathrm{d}Q}{\mathrm{d}r} = -2\left(\mu\left(E - \frac{\zeta}{r}\right) - \frac{l(l+1)}{2r^2}\right)R(r)\,. \tag{3.24b}$$

We use as a power-series *ansatz*, which factors in the asymptotic behavior,

$$R(r) = r^\sigma \mathrm{e}^{-\lambda r}\sum_{n=0}^{\infty}\frac{a_n}{r^n}\,, \tag{3.25a}$$

$$Q(r) = r^\sigma \mathrm{e}^{-\lambda r}\sum_{n=0}^{\infty}\frac{b_n}{r^n}\,, \tag{3.25b}$$

and by substituting this *ansatz* in eqs. (3.24) we find from the leading terms

$$\lambda = \sqrt{-2\mu E} \quad \text{and} \quad \sigma = \frac{\zeta\mu}{\lambda}\,. \tag{3.26}$$

By equating the coefficients, we find the relations

$$a_n = \frac{l(l+1) - (\sigma - n)(\sigma - n + 1)}{2n\lambda} a_{n-1}\,, \tag{3.27a}$$

$$b_n = \frac{(\sigma + n)(\sigma - n + 1) - l(l+1)}{2n} a_{n-1}\,, \tag{3.27b}$$

and by setting $a_0 = 1$ arbitrarily it follows $b_0 = -\lambda$. Since near the practical infinity a_∞ the expansion parameter λr is large, only a small number of terms is sufficient to provide an accurate approximation of the wave function in this region. We use this asymptotic expansion, to obtain the final k points on the grid. In `dish` this is implemented in `dish.schrodinger.insch`.

Using the AM procedure we can integrate inwards and outwards for a given E to the outer classical turning point a_c, where $E = V(a_c)$. As we used arbitrary values for some starting values, the scaling is to be neglected, and we scale the first component $R^E_{\text{out}}(r)$ of the result from the inward integration to match the value of the outward result $R^E_{\text{in}}(r)$ at a_c, i.e. $R^E_{\text{in}}(r) \leftarrow R^E_{\text{in}}(r) \cdot R^E_{\text{out}}(a_c)\,/\,R^E_{\text{in}}(a_c)$. We also need to rescale Q^E_{in} by the same factor to keep the ratio constant. The results from the inward and outward integration are then concatenated as

$$R^E(r) = \begin{cases} R^E_{\text{out}}(r) & ,r \le a_c \\ R^E_{\text{in}}(r) & ,r > a_c \end{cases} \quad \text{and} \quad Q^E(r) = \begin{cases} Q^E_{\text{out}}(r) & ,r \le a_c \\ Q^E_{\text{in}}(r) & ,r > a_c \end{cases}\,. \tag{3.28}$$

Since for all physically meaningful potentials $V(r)$ the characteristics of the wave function will the same as for the Coulomb potential discussed above, we know the number of radial nodes is $n_r = n - l - 1$ (cf. chapter 2.3.2). We check if $R^E(r)$ fulfills this property, and, if it is smaller than n_r, the energy ($E < 0$) is increased by a factor of 0.9 or, if there are too many nodes, E is reduced by a factor of 1.1. If the number of nodes matches with n_r, we check if R^E is a solution of the radial SE by checking if $Q = \frac{\mathrm{d}R}{\mathrm{d}r}$ is continuous, i.e. $Q^E_{\text{out}}(a_c) \stackrel{?}{=} Q^E_{\text{in}}(a_c)$. If not we adjust E from the following considerations: Let $R_{1/2}$ be two solutions of the radial SE and $Q_{1/2}$ the corresponding derivatives. Then from the radial SE follows

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}r}\left(Q_2 R_1 - R_2 Q_1\right) &= \frac{\mathrm{d}^2 R_2}{\mathrm{d}r^2} R_1 + Q_2 Q_1 - Q_1 Q_2 - R_2 \frac{\mathrm{d}^2 R_1}{\mathrm{d}r^2} \\ &= -2\left(\mu(E_2 - V) - \frac{l(l+1)}{2r^2}\right) R_2 R_1 + 2R_2 \left(\mu(E_1 - V) - \frac{l(l+1)}{2r^2}\right) R_1 \\ &= 2\mu\left(E_1 - E_2\right) R_1 R_2\,, \end{aligned} \tag{3.29}$$

and by integrating this we find

$$2\mu(E_1 - E_2)\int_{a_c}^{\infty} R_1 R_2 \mathrm{d}r = -\left(Q_2 R_1 - R_2 Q_1\right)^{+},$$
$$2\mu(E_1 - E_2)\int_{0}^{a_c} R_1 R_2 \mathrm{d}r = \left(Q_2 R_1 - R_2 Q_1\right)^{-},$$

where the superscripts $^{\pm}$ indicate whether the expression should be evaluated from above or below a_c. Combining these equations yields

$$E_1 - E_2 = \frac{\left(Q_1^{+} - Q_1^{-}\right) R_2(a_c) + \left(Q_2^{-} - Q_2^{+}\right) R_1(a_c)}{2\mu \int_0^{\infty} R_1 R_2 \mathrm{d}r}, \tag{3.31}$$

and, supposing the Q_2 is continuous at a_c while Q_1 is not and applying the approximation $R_1 \approx R_2$, it follows

$$E_2 \approx E_1 + \frac{\left(Q_1^{-} - Q_1^{+}\right) R_1(a_c)}{2\mu \int_0^{\infty} R_1^2 \mathrm{d}r}. \tag{3.32}$$

We use eq. (3.32) to adjust E if the number of nodes was correct but Q not continuous. If this value is lower than a value E_l which was a former guess for the energy where the number of radial nodes was too few, then instead the energy is adjusted by $E \leftarrow (E + E_l)/2$. The same adjustment is done in the case for an upper limit E_u, where the node count was too high.

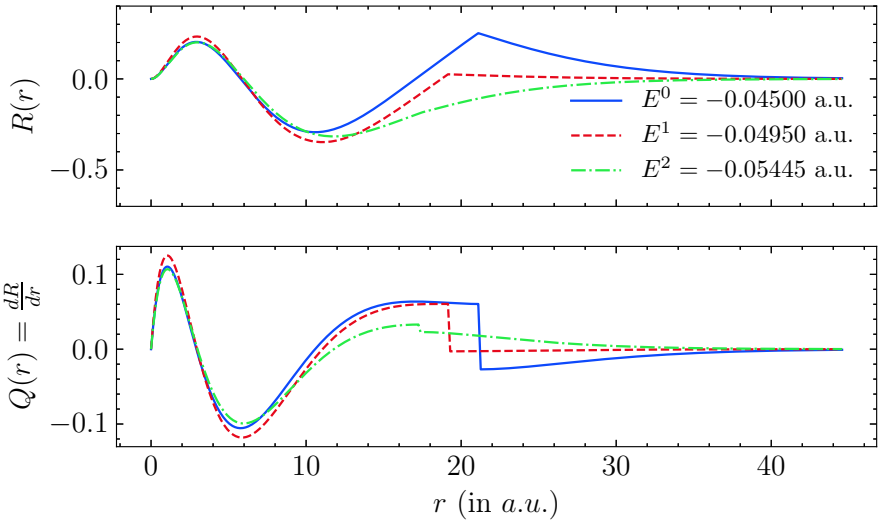

Figure 3.2.: Three iterations of the master routine to find the energy E and radial wave function R for the $3p$-state of Hydrogen modeled by a Coulomb potential.

We repeat the process described above and adjust the energy until the number of nodes is correct and $Q^E(r)$ is continuous at a_c in the order of the machine precision ($< 10^{-15}$). In that case we found the eigenenergy E, and the related radial wave function is $R(r) = N \cdot R^E(r)$ with the normalization factor N so that $R(r)$ fulfills eq. (2.25). The integral in the

normalization condition is calculated numerically, which will be covered in chapter 3.7. In fig. 3.2 the first three iterations of finding the $3p$ wave function for Hydrogen are depicted. For demonstration purposes the starting guess for the energy $E^0 = -0.045\,E_\mathrm{h}$ is quite a bit off as here the discontinuity in $Q(r)$ can be seen easily. To minimize the required number of iterations, it is recommended to start with the analytical energy solution for a Coulomb potential E_C as this is already a good approximation for physically meaningful potentials and, therefore, the energy for such potentials is close to E_C.

In `dish`, this routine to find the eigenenergy and radial function is implemented in `dish.schrodinger.master`. The implementation is close to the above described proceeding and is not very pleasant to work with as mostly low level data structures are used. It is recommended to use the high level interface described in the next chapter which wraps the function `master()` from the module `dish.schrodinger.master`.

3.6. Solutions to the Dirac Equation

To find solutions to the radial functions of the DE for hydrogenic systems

$$\begin{aligned}\left(V + mc^2\right) f_\kappa + c\left(\frac{\mathrm{d}}{\mathrm{d}r} - \frac{\kappa}{r}\right) g_\kappa &= E f_\kappa\,,\\ -c\left(\frac{\mathrm{d}}{\mathrm{d}r} + \frac{\kappa}{r}\right) f_\kappa + \left(V - mc^2\right) g_\kappa &= E g_\kappa\end{aligned}$$

(cf. eqs (2.53)) we will closely follow the procedure for the non-relativistic calculations discussed in the last section. By defining

$$y(t) = \begin{bmatrix} f_\kappa(r) \\ g_\kappa(r) \end{bmatrix}, \quad \text{where} \quad r = r(t) \quad \text{and} \tag{3.34a}$$

$$f(t,y) = \frac{\mathrm{d}r}{\mathrm{d}t}\begin{bmatrix} -\frac{\kappa}{r} f_\kappa(r) - \alpha\left(W_\kappa - V(r) + 2m\alpha^{-2}\right) g_\kappa(r) \\ \frac{\kappa}{r} g_\kappa(r) + \alpha\left(W_\kappa - V(r)\right) f_\kappa(r) \end{bmatrix}, \tag{3.34b}$$

we expressed the problem in the form eq. (3.3). Here $W_\kappa := E_\kappa - mc^2$ is used as an abbreviation for better comparability with the non-relativistic results. Writing this system of linear ODEs as $f(t,y) = G(t)y(t)$, we find

$$G(t) = \begin{pmatrix} a(t) & b(t) \\ c(t) & d(t) \end{pmatrix}, \tag{3.35a}$$

where the matrix elements are given as

$$\begin{aligned} a(t) &= -\tfrac{\mathrm{d}r}{\mathrm{d}t}\tfrac{\kappa}{r}\,, & b(t) &= -\alpha\tfrac{\mathrm{d}r}{\mathrm{d}t}\left(W_\kappa - V(r) + 2m\alpha^{-2}\right)\,,\\ c(t) &= \alpha\tfrac{\mathrm{d}r}{\mathrm{d}t}\left(W_\kappa - V(r)\right) \quad \text{and} & d(t) &= \tfrac{\mathrm{d}r}{\mathrm{d}t}\tfrac{\kappa}{r}\,. \end{aligned} \tag{3.35b}$$

The initial k steps required to use the AM method for outward integration are again obtained using Lagrangian integration formulas. Factorizing r^γ where $\gamma = \sqrt{\kappa^2 - (\alpha Z)^2}$

from the radial functions f_κ and g_κ, yields

$$f_\kappa(r) = r^\gamma u(r(t)) \tag{3.36a}$$
$$g_\kappa(r) = r^\gamma v(r(t)) \tag{3.36b}$$

and using this we find from eqs. (3.34a)

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \underbrace{-\frac{\mathrm{d}r}{\mathrm{d}t}\frac{\gamma+\kappa}{r}}_{=:A(t)} u(t) \underbrace{-\alpha\frac{\mathrm{d}r}{\mathrm{d}t}\left(W_\kappa - V(r) + 2m\alpha^{-2}\right)}_{=:B(t)} v(t) \quad \text{and} \tag{3.37a}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \underbrace{\alpha\frac{\mathrm{d}r}{\mathrm{d}t}\left(W_\kappa - V(r)\right)}_{=:C(t)} u(t) \underbrace{-\frac{\mathrm{d}r}{\mathrm{d}t}\frac{\gamma-\kappa}{r}}_{=:D(t)} v(t). \tag{3.37b}$$

Assuming the potential satisfies $V(r \to 0) = -Z/r$ and setting $u(0) = 1$ arbitrarily, it follows

$$v(0) = -\frac{\kappa+\gamma}{\alpha Z} = \frac{\alpha Z}{\gamma-\kappa}\,, \tag{3.38}$$

where the two expressions on the right hand-side are equivalent. To avoid loss of precision due to subtractive cancellation we use the first one for $\kappa > 0$ and the latter one for $\kappa < 0$. We then approximate the derivatives in eqs. (3.37) at the first $k+1$ grid points using the Lagrangian differentiation formulas eq. (3.15) and find the system of $2k$ inhomogeneous linear equations

$$\sum_{j=1}^{k} m[i,j]u[j] - A[i]u[i] - B[i]v[i] = -m[i,0]u[0] \quad \text{and} \tag{3.39a}$$

$$\sum_{j=1}^{k} m[i,j]v[j] - C[i]u[i] - D[i]v[i] = -m[i,0]v[0]\,, \tag{3.39b}$$

which can be solved using Gaussian elimination to give $u[i], v[i]$ for $i = 1, \ldots, k$. The initial values of the radial functions are found by using eqs. (3.36) on the grid points $i = 1, \ldots, k$. This procedure is implemented in `dish.dirac.outdir`.

To find the initial values for the inward integration, one can use an expansion of the radial functions like in the non-relativistic case. Assuming that $V(r \to \infty) = -\zeta/r$ where $\zeta = Z - N + 1$ is the ionic charge of the hydrogenic system, we use the *ansatz*es

$$f_\kappa(r) = r^\sigma \mathrm{e}^{-\lambda r}\left(\sqrt{\frac{c^2+E}{2c^2}}\sum_{n=0}^{\infty}\frac{a_n}{r^n} + \sqrt{\frac{c^2-E}{2c^2}}\sum_{n=1}^{\infty}\frac{b_n}{r^n}\right), \tag{3.40a}$$

$$g_\kappa(r) = r^\sigma \mathrm{e}^{-\lambda r}\left(\sqrt{\frac{c^2+E}{2c^2}}\sum_{n=0}^{\infty}\frac{a_n}{r^n} - \sqrt{\frac{c^2-E}{2c^2}}\sum_{n=1}^{\infty}\frac{b_n}{r^n}\right), \tag{3.40b}$$

where $\lambda = \sqrt{m^2c^2 - E^2/c^2}$. The radial DEs can only by solved using this *ansatz* for $\sigma =$

$E\zeta/(c^2\lambda)$. It can be shown that the expansion coefficients are given by

$$a_0 = 1 \quad \text{and} \quad b_1 = \frac{1}{2c}\left(\kappa + \frac{\zeta m}{\lambda}\right), \tag{3.41a}$$

$$b_{n+1} = \frac{1}{2n\lambda}\left(\kappa^2 - (n-\sigma)^2 - \frac{\zeta^2}{c^2}\right) b_n, \quad n = 1, 2, \dots, \tag{3.41b}$$

$$a_n = \frac{c}{n\lambda}\left(\kappa + (n-\sigma)\frac{E}{mc^2} - \frac{\zeta\lambda}{mc^2}\right) b_n, \quad n = 1, 2, \dots. \tag{3.41c}$$

Here also just a few terms of eqs. (3.40) are required since λr is large in this asymptotic region. The implementation in `dish` can be found in `dish.dirac.indir`.

To find the energy and wave functions for a specific state, the same iterative approach as in the non-relativistic case is used. The steps are fully analogue, just the routines to find the initial values are replaced, and the AM procedure is performed using a different G. While solving the SE, R was made continuous by scaling the outer part, and it was checked whether the derivative Q was continuous. Here, the large component f_κ is scaled and the small component is checked for continuity. The derivation of the energy adjustment in the case of the correct number of nodes is also in full analogy to eqs. (3.29) to (3.32):

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(f_1 g_2 - f_2 g_1\right) = \frac{1}{c}(W_2 - W_1)(f_1 f_2 + g_1 g_2) \tag{3.42}$$

$$\implies f_1(a_c)\left(g_2^- - g_2^+\right) + f_2(a_c)\left(g_1^+ - g_2^-\right) = \frac{1}{c}\left(W_2 - W_1\right)\int_0^\infty (f_1 f_2 + g_1 g_2)\,\mathrm{d}r \tag{3.43}$$

$$\implies W_2 \approx W_1 + \frac{c f_1(a_c)(g_1^+ - g_1^-)}{\int_0^\infty (f_1^2 + g_1^2)\mathrm{d}r} \tag{3.44}$$

The solving routine is implemented in the module `dish.dirac.master`.

3.7. Integration on a Finite Grid

To normalize wave functions or to calculate the radial part of matrix elements one needs to calculate integrals of the form

$$I_0^\infty(f) = \int_0^\infty f(r)\mathrm{d}r, \quad f : \mathbb{R} \to \mathbb{C}. \tag{3.45}$$

We cannot find the analytic solution since we do not necessarily know the analytic form of f and, therefore, we approximate the integral using a quadrature rule

$$I_0^\infty(f) \approx Q(f) = \sum_{n=0}^{N} w_n f(r[n]), \tag{3.46}$$

where $f(r[n])$ are the values of the function f at the grid points $r[n], n \in \{0, 1, \dots, N\}$ and w_n are the corresponding weights. There is a multitude of quadrature rules that choose diverse weights to minimize the numerical error for specific function classes and grids.

An introduction to numerical integration theory can be found in many textbooks about numerical analysis. Here, we will briefly discuss the two methods used in `dish`, while more details can be found in [36, 38].

We transform the integral eq. (3.45) using the transformation rule to an integral in t, which we chose to be an equidistant grid (cf. eq. (3.1b)) of spacing width h. And since we cannot integrate to infinity using a quadrature rule, we apply another approximation by integrating only until a_∞. This approximates the integral sufficiently well since we want to integrate wave functions and chose a_∞ so that the wave functions are negligible in this distance. For the integral this yields

$$I_0^\infty(f) \approx I_0^{a_\infty} = \int_0^{t(a_\infty)} f(r(t)) \frac{\mathrm{d}r}{\mathrm{d}t} \mathrm{d}t \approx \sum_{n=0}^{N} w_n f[n] \,, \tag{3.47}$$

where $f[n]$ is the value of f at the n^{th} grid point.
The composite trapezoidal rule

$$T(h) = h \cdot \left(\frac{1}{2} f[0] + \sum_{n=1}^{N-1} f[n] + \frac{1}{2} f[N] \right) \tag{3.48}$$

with an error of

$$R_T = -\frac{1}{12} t(a_\infty) h^2 f^{(2)}(\xi) \quad \text{for a} \quad \xi \in [0, t(a_\infty)] \tag{3.49}$$

for $f \in C^2(\mathbb{R}, \mathbb{C})$ has the advantage of simplicity and flexibility while converging like $\mathcal{O}(h^2)$ for $h \to 0$.
If the number of grid points $N = 2^k + 1$, where k is a positive integer, one can use Romberg's method which is a modification of the trapezoidal rule that uses repeated Richardson extrapolation. Since this requires more extensive knowledge in numerical analysis it will not be covered here in detail. The idea is to use the trapezoidal rule and then extrapolate the next order correction in h by using the already computed values. The extrapolation is repeated recursively using Neville's scheme for an efficient evaluation. The truncation error for $f \in C^{2k+2}([0, t(a_\infty)], \mathbb{C})$ is $\mathcal{O}(h^{2k} f^{(2k)})$, which yields a significant improvement in convergence speed.

In fig. 3.3 the error of both methods is compared. The function $f(r) = \mathrm{e}^{-0.4r} \sin(3r)$ of which the integral was approximated was chosen because it has a similar behavior as wave functions. For small r it oscillates and it decays for large r exponentially. For integration a `DistanceGrid` with the parameters $r0 = 0.1, r\_max = 100$ was used while h (and hence the number of grid points N) was varied. From the figure it can be seen that at $h \approx 10^{-3}$ Romberg's method reaches machine precision while the error of the trapezoidal rule is in the order of 10^{-8}.

In `dish` for both integration methods the implementations of the scientific computation package `scipy` are used.
To use Romberg's method, integration should be performed on a radial grid which is

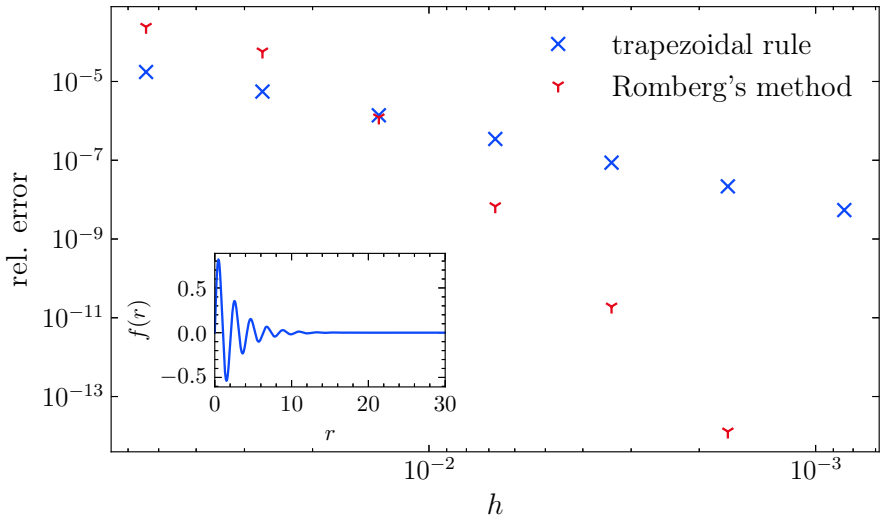

Figure 3.3.: Convergence speed of the trapezoidal rule and Romberg's method. It is compared by approximating the radial integral $f(r) = \mathrm{e}^{-0.4r}\sin(3r)$.

an instance of `dish.util.radial.grid.grid.RombergIntegrationGrid`. This is a subclass of `DistanceGrid`.
To integrate on both, a `DistanceGrid` or a `RombergIntegrationGrid`, one can use

```
from dish.util.radial.integration import integrate_on_grid
integrate_on_grid(y:np.ndarray, grid: DistanceGrid)
```

where `y` should be an array of the functions values at the grid points. `dish` also offers a high-level interface to calculate matrix elements which is discussed in chapter 5.3.2.

4. Benchmarking

In this chapter, we will benchmark the results produced by `dish` by comparing it in various settings with results from analytical calculations or other atomic codes.
All benchmarks are performed on a system with an AMD Ryzen 7 7840u processor and 48GiB of 5600MT/s memory running Ubuntu 22.04.4 LTS and Python 3.9.18.

4.1. Energies and Wave Functions

Solving the SE and DE for a specific state using the procedure described in the previous chapter, yields as a result the energy of the state and its radial wave function. To classify the quality of the results produced by `dish`, in this chapter, the absolute and relative error

$$\epsilon_{\text{abs}}(E) = E_{\text{dish}} - E_{\text{ref}} \quad \text{and} \quad \epsilon_{\text{rel}} = \left| \frac{E_{\text{dish}} - E_{\text{ref}}}{E_{\text{ref}}} \right| \tag{4.1}$$

will be given for all scalar values like the energies E. Since for the radial wave functions ϕ many values are very close to zero, and, therefore, insignificant differences would have huge impacts, no relative error is given for these, but, instead, the agreement of the wave functions is quantified by the sum of absolute errors

$$\epsilon_{\text{abs}}(\phi) = \sum_{i=0}^{N-1} |\phi_{\text{dish}}[i] - \phi_{\text{ref}}[i]| \tag{4.2}$$

and by the mean squared error

$$\epsilon_{\text{mse}}(\phi) = \sum_{i=0}^{N-1} \frac{(\phi_{\text{dish}}[i] - \phi_{\text{ref}}[i])^2}{N}. \tag{4.3}$$

For relativistic wave functions this will be given separately for the large component f and the small component g, and for non-relativistic wave functions just the radial component R will be checked.

4.1.1. Point-like Nucleus

For a point-like nucleus, whose potential is a Coulomb potential, the analytical form of the wave functions is known and was already derived in sections 2.3.2 and 2.4.3. Therefore, this case is well suited as a benchmark because it does not rely on any particular implementation or numerical methods.

First we will investigate the influence of the parameters in the solving process, namely the grid parameters and the order of the Adams-Moulton procedure. As discussed in the

previous chapter the fineness of the grid and, therefore, the accuracy of the numerical procedures, is determined by the grid's parameter h. Hence, we will analyze the choice of other parameters in dependence of h. Here, we will discuss the results mostly at the example of the $3p_{1/2}$ state of Hydrogen which was already shown as an example in fig. 2.3. The trends found for this state coincide with the findings for other states.

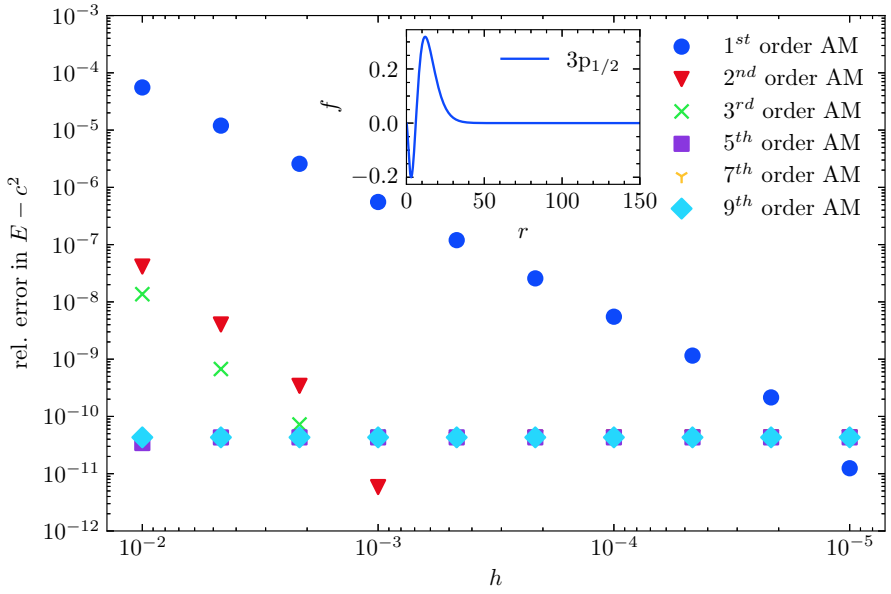

Figure 4.1.: Relative error in the energy of the $3p_{1/2}$ state of point-like hydrogen in dependence of `h` for different orders of the AM procedure. The other parameters of the grid used are `r0` $= 10^{-6}$ and `r_max` $= 150$. In the inner plot the large component of the radial wave function is displayed.

In fig. 4.1 it can be seen that a higher order AM procedure yields better accuracy of the state's energy by converging faster to the maximal possible accuracy. As can be seen in the figure, usually a good trade-off between accuracy and speed is choosing a fifth order AM procedure.

In fig. 4.2 the energy for the same state was calculated on a similar grid where only the maximal grid value was decreased to `r_max` $= 50$. Even though the wave function looks like being negligible in that region (see the small, inner plot in the figure) it has a major influence on the result: The relative error in the energy to which the solving routine converges is more than three orders of magnitude worse compared to the results shown on the larger grid in fig. 4.1. This means that in that case the numerical error induced by the order of the AM procedure is dominated by this error, so make sure you choose a suitable large grid. To find an appropriate value for `r_max`, `dish` provides utility functions `find_suitable_number_of_integration_points_dirac` and `..._schrodinger` in the module `dish.util.misc`.

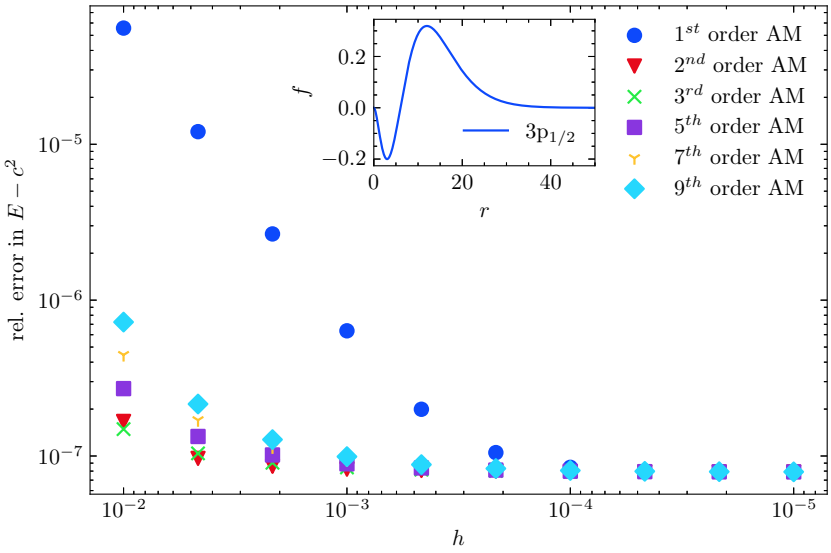

Figure 4.2.: Relative error in the energy of the $3p_{1/2}$ state of point-like hydrogen in dependence of `h` for different orders of the AM procedure. The other parameters of the grid used are `r0` $= 10^{-6}$ and `r_max` $= 50$. In the inner plot the large component of the radial wave function is displayed.

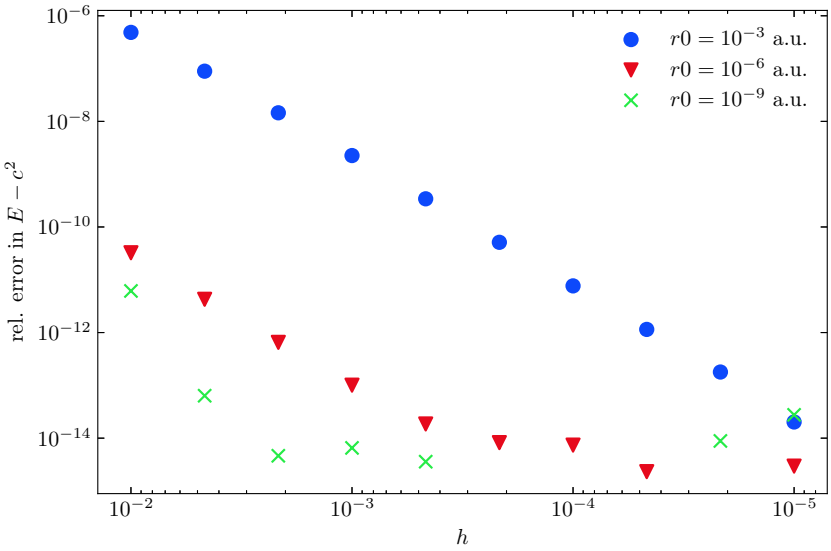

Figure 4.3.: Relative error in the energy of the $3p_{1/2}$ state of hydrogenic uranium U^{91+}, modeled with a point-like nucleus, in dependence of `h` for different values of `r0`. A fifth order AM procedure was used and `r_max` $= 15$.

The choice of the value for r_0 does not have a great impact for nuclei like hydrogen with a small charge. For highly charged hydrogenic systems like U^{91+} however, it does influence the result as can be seen in fig. 4.3. We have observed, choosing $r_0 = 10^{-4}$ for lightweight nuclei up to $Z \approx 20$ is sufficient, $r_0 = 10^{-6}$ works for most systems, and for extremely highly charged systems one might consider using even smaller values.

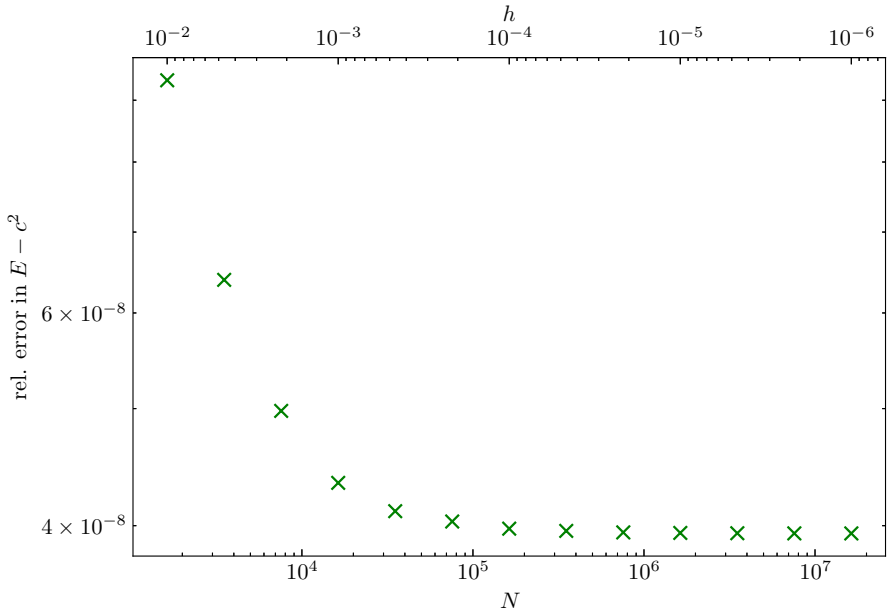

Figure 4.4.: Relative error in the energy of the $3p_{1/2}$ state of point-like hydrogen in dependence of `N/h`. A fifth order AM procedure was used on a grid defined by the parameters `r0` $= 10^{-6}$, `h` and `r_max` $= 15$.

Having chosen an appropriate order of the AM procedure and values for `r0` and `r_max` for the grid, one is not limited by these values and the accuracy of the results (up to a boundary value) is only dependent on the fineness `h` of the grid. This can be seen in fig. 4.4 for the same example state that was discussed in the previous steps. As a trade-off for the accuracy, there is the computational expense that grows linearly with the number of grid points N which goes reciprocal with h. In fig. 4.5 the time to solve the DE dependent on the number of grid points is shown. You can see that the time required for solving on a grid of 10 000 points is already in the order of seconds using the pure Python implementation. This shows the main disadvantage of Python which is the execution speed. An applied profiler showed that the most time-consuming part is the k^{th} order AM procedure where for every grid point a $k \times k$ system of equations needs to be solved. To increase the performance, this critical part was also implemented in a Fortran subroutine which can be compiled during the building process and is used automatically if present. (See 5.1 for more details on how to compile this during the setup.) Using this Fortran extension increases the solving times by up one to two orders of magnitude. Also, the coefficients which are required for the routines `adams` and `outdir/outsch` are only calculated when

the routines are called for the first time and cached afterward which is the reason for the first call taking a few milliseconds longer.
On modern computers the memory cost is not the limiting factor as the memory requirements during the solving process only hit the order of GB for grids with $N \approx 10^7$ which brings no significant improvement in accuracy.

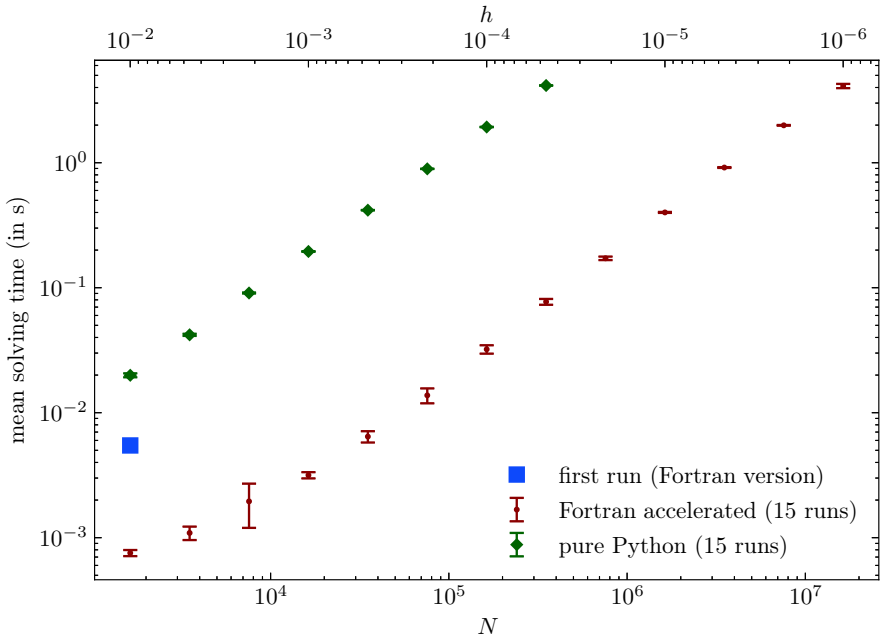

Figure 4.5.: Time required to solve the DE for the $3p_{1/2}$ state of point-like hydrogen in dependence of `N/h`.

For most calculations $h = 10^{-2}$ or $h = 10^{-3}$ yields results that are precise enough while providing a high performance. To improve the accuracy of the results, a value in the order of $h = 10^{-4}$ can be used. Choosing even lower values for h comes with a high computational cost while the numerical errors have larger influences and hence usually is not worth.

As the main focus of this thesis is on the relativistic calculations, for the rest of this chapter only relativistic results are examined if not explicitly stated otherwise. The deviations of the comparisons of the non-relativistic results of `dish` are in the same order of magnitude.

4.1.2. Finite-sized Nucleus

To evaluate the accuracy of energies and wave functions for finite sized nuclei calculated by `dish`, its results will be compared with the results from two other atomic codes, namely GRASP2018 [4] and JAC [8]. Here, the results for the states up to $5g_{9/2}$ for the hydrogenic systems ${}^{1}_{1}\mathrm{H}$, ${}^{40}_{20}\mathrm{Ca}^{19+}$ and ${}^{238}_{92}\mathrm{U}^{91+}$ are compared as representatives for a lightweight,

a medium and a heavy system. For most stable nuclei a good approximation for the Fermi parameter a is $2.3\,\mathrm{fm}/(4\ln 3)$ [29]. For nuclei being sufficiently large ($R_{\mathrm{rms}}^2 > \frac{7}{5}\pi^2 a^2$) the value for the parameter c can be calculated from the root-mean-square radius R_{rms} as

$$c \approx \sqrt{\frac{5}{3}R_{\mathrm{rms}}^2 - \frac{7}{3}\pi^2 a^2} \quad \text{[cf. eq. (2.27)]} \quad . \tag{4.4}$$

Therefore, for small nuclei the Fermi-like model is not applicable with the above choice of the parameter a. For smaller values of a one can use this model (like done below for hydrogen when comparing with GRASP, cf. table 4.1), but, since the edge of the nucleus is chosen sharper in this case one can also use the uniformly charged sphere as a good approximation (like done in the comparison with JAC).
Using the values for R_{rms} from [34], we obtain as the Fermi parameters in table 4.1.

Table 4.1.: Root-mean-square radii and used Fermi parameters $t = 4\ln(3)a$ and c for ${}^{1}_{1}\mathrm{H}$,${}^{40}_{20}\mathrm{Ca}$ and ${}^{238}_{92}\mathrm{U}$. All parameters are given in fm.

	${}^{1}_{1}\mathrm{H}$	${}^{40}_{20}\mathrm{Ca}$	${}^{238}_{92}\mathrm{U}$
R_{rms}	0.87830	3.47760	5.85710
c	0.69975	3.72125	7.13215
t	1.0	2.3	2.3

Since the reference atomic codes use a grid with other parameters or a slightly different structured grid, the wave functions obtained from `dish` are interpolated on these other grids to compare the values.

GRASP

The General-purpose Relativistic Atomic Structure Package (GRASP) is a well established atomic code written in Fortran which can be used to calculate various atomic properties and processes for complex systems. It implements the fully relativistic multiconfiguration Dirac-Hartree-Fock method [9, ch.6] which is suited for medium to heavy atomic systems. Since we are only interested in hydrogenic systems, we can use the one-electron wave functions produced from GRASP, where the self-consistent field calculations are not needed, also for lightweight atoms as a reference. A similar approach as in `dish` is used (in the routine `rwfnestimate`) to generate these.

To perform calculations, GRASP provides several interface scripts which perform parts of the calculations and exchange data via files. Because GRASP is very versatile and has grown a lot during its development there are a lot of such scripts which can be combined in various configurations for different calculations. Hence, even for rather basic calculations like finding the atomic state functions the setup is quite complex. In the appendix in chapter A.3.1 an example of such a calculation is shown, and it is covered how this process was automated to produce the following results.

In the tables 4.2, 4.3 and 4.4 the relative error of the `dish` results are given in comparison with the results produced by GRASP for the energy of the states $1s_{1/2}$ up to $5g_{9/2}$. Also, the mean squared error for the large and small component is given. One can see that the energy calculated by `dish` matches the result from GRASP very well as the $\epsilon_{\text{rel}}(E) < 1.1 \times 10^{-9}$ for all the given states for all three systems. Also the agreement of the radial components of the wave functions is good. For hydrogen $\epsilon_{\text{mse}}(f) < 10^{-15}$ and mostly $\epsilon_{\text{mse}}(g) < 10^{-20}$. Here only for the $s_{1/2}$ and $p_{1/2}$ states, where the probability density is non-zero at and close to the origin, the deviations are slightly worse with $\epsilon_{\text{mse}}(g) < 10^{-15}$. For hydrogenic calcium this trend is similar as $\epsilon_{\text{mse}}(f) < 10^{-13}, \epsilon_{\text{mse}}(g) < 10^{-16}$ mostly and for the $s_{1/2}$ and $p_{1/2}$ state a little worse. The same can be observed for hydrogenic uranium as for the most states $\epsilon_{\text{mse}}(f) < 10^{-11}, \epsilon_{\text{mse}}(g) < 10^{-10}$ while for the $s_{1/2}, p_{1/2}$ states the agreement is a few orders of magnitude worse. This is probably originated in the derivation of the most inner points where the potential is approximated to be Coulombic. For heavier and hence larger nuclei this approximation gets worse as the nucleus is modeled having a finite size and therefore the radial wave functions describe the problem slightly worse.

Table 4.2.: Deviation of the results obtained from `dish` and GRASP2018 for states of hydrogen up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 1.000 \times 10^{-6}, h = 0.01$ and $r_{\text{max}} = 185$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$1s_{1/2}$	6.624×10^{-13}	7.186×10^{-16}	1.171×10^{-16}
$2s_{1/2}$	3.499×10^{-13}	8.265×10^{-16}	1.396×10^{-17}
$2p_{1/2}$	9.137×10^{-13}	4.480×10^{-16}	6.663×10^{-21}
$2p_{3/2}$	1.841×10^{-13}	4.339×10^{-16}	1.186×10^{-21}
$3s_{1/2}$	8.623×10^{-12}	4.410×10^{-16}	4.019×10^{-18}
$3p_{1/2}$	8.912×10^{-12}	2.618×10^{-16}	1.489×10^{-21}
$3p_{3/2}$	2.633×10^{-12}	2.880×10^{-16}	7.160×10^{-22}
$3d_{3/2}$	2.633×10^{-12}	2.420×10^{-16}	1.014×10^{-21}
$3d_{5/2}$	9.018×10^{-14}	2.087×10^{-16}	3.630×10^{-22}
$4s_{1/2}$	4.561×10^{-11}	3.784×10^{-16}	1.658×10^{-18}
$4p_{1/2}$	4.573×10^{-11}	4.597×10^{-16}	1.294×10^{-21}
$4p_{3/2}$	2.256×10^{-11}	5.052×10^{-16}	4.864×10^{-22}
$4d_{3/2}$	2.256×10^{-11}	6.884×10^{-16}	5.476×10^{-22}
$4d_{5/2}$	5.030×10^{-12}	6.374×10^{-16}	5.868×10^{-22}
$4f_{5/2}$	5.030×10^{-12}	1.627×10^{-16}	1.419×10^{-22}
$4f_{7/2}$	4.063×10^{-14}	1.574×10^{-16}	1.723×10^{-22}
$5s_{1/2}$	1.698×10^{-10}	3.845×10^{-16}	8.353×10^{-19}

Continued on next page

Table 4.2.: Deviation of the results obtained from `dish` and GRASP2018 for states of hydrogen up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 1.000 \times 10^{-6}, h = 0.01$ and $r_{\max} = 185$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$5p_{1/2}$	1.699×10^{-10}	4.798×10^{-16}	7.579×10^{-22}
$5p_{3/2}$	1.056×10^{-10}	4.840×10^{-16}	3.103×10^{-22}
$5d_{3/2}$	1.056×10^{-10}	4.832×10^{-16}	3.533×10^{-22}
$5d_{5/2}$	4.272×10^{-11}	4.388×10^{-16}	4.996×10^{-22}
$5f_{5/2}$	4.272×10^{-11}	1.696×10^{-16}	3.605×10^{-22}
$5f_{7/2}$	8.569×10^{-12}	2.125×10^{-16}	1.158×10^{-22}
$5g_{7/2}$	8.569×10^{-12}	1.505×10^{-16}	1.482×10^{-22}
$5g_{9/2}$	1.133×10^{-13}	1.508×10^{-16}	8.352×10^{-23}

Table 4.3.: Deviation of the results obtained from `dish` and GRASP2018 for states of hydrogenic calcium up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 1.000 \times 10^{-6}, h = 0.01$ and $r_{\max} = 50$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$1s_{1/2}$	5.431×10^{-12}	3.199×10^{-10}	1.023×10^{-8}
$2s_{1/2}$	4.373×10^{-12}	4.026×10^{-11}	1.242×10^{-9}
$2p_{1/2}$	1.278×10^{-11}	5.030×10^{-12}	1.195×10^{-13}
$2p_{3/2}$	2.747×10^{-12}	8.469×10^{-15}	1.474×10^{-17}
$3s_{1/2}$	3.495×10^{-12}	1.203×10^{-11}	3.580×10^{-10}
$3p_{1/2}$	2.212×10^{-12}	1.722×10^{-12}	4.080×10^{-14}
$3p_{3/2}$	8.635×10^{-13}	1.194×10^{-14}	8.585×10^{-18}
$3d_{3/2}$	9.253×10^{-13}	7.758×10^{-15}	3.015×10^{-18}
$3d_{5/2}$	9.256×10^{-13}	6.543×10^{-15}	3.051×10^{-18}
$4s_{1/2}$	4.088×10^{-11}	5.131×10^{-12}	1.476×10^{-10}
$4p_{1/2}$	3.666×10^{-11}	7.536×10^{-13}	1.780×10^{-14}
$4p_{3/2}$	1.918×10^{-11}	7.579×10^{-15}	5.541×10^{-18}
$4d_{3/2}$	1.913×10^{-11}	6.637×10^{-15}	3.819×10^{-18}
$4d_{5/2}$	3.907×10^{-12}	5.764×10^{-15}	2.286×10^{-18}
$4f_{5/2}$	3.907×10^{-12}	5.774×10^{-15}	1.616×10^{-18}
$4f_{7/2}$	6.555×10^{-13}	5.187×10^{-15}	1.510×10^{-18}
$5s_{1/2}$	1.643×10^{-10}	2.686×10^{-12}	7.430×10^{-11}
$5p_{1/2}$	1.610×10^{-10}	3.956×10^{-13}	9.174×10^{-15}

Continued on next page

Table 4.3.: Deviation of the results obtained from `dish` and GRASP2018 for states of hydrogenic calcium up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 1.000 \times 10^{-6}, h = 0.01$ and $r_{\max} = 50$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$5p_{3/2}$	1.020×10^{-10}	1.096×10^{-14}	3.906×10^{-18}
$5d_{3/2}$	1.019×10^{-10}	1.032×10^{-14}	3.086×10^{-18}
$5d_{5/2}$	4.089×10^{-11}	1.036×10^{-14}	2.254×10^{-18}
$5f_{5/2}$	4.089×10^{-11}	6.586×10^{-15}	1.577×10^{-18}
$5f_{7/2}$	7.457×10^{-12}	6.613×10^{-15}	2.190×10^{-18}
$5g_{7/2}$	7.456×10^{-12}	1.907×10^{-15}	1.053×10^{-18}
$5g_{9/2}$	4.609×10^{-13}	1.839×10^{-15}	8.720×10^{-19}

Table 4.4.: Deviation of the results obtained from `dish` and GRASP2018 for states of hydrogenic uranium up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 1.000 \times 10^{-8}, h = 0.001$ and $r_{\max} = 5$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$1s_{1/2}$	9.536×10^{-10}	8.085×10^{-4}	1.569×10^{-3}
$2s_{1/2}$	1.073×10^{-9}	1.565×10^{-4}	2.888×10^{-4}
$2p_{1/2}$	5.141×10^{-10}	3.261×10^{-5}	1.484×10^{-5}
$2p_{3/2}$	6.205×10^{-11}	2.460×10^{-12}	1.554×10^{-11}
$3s_{1/2}$	7.883×10^{-10}	4.743×10^{-5}	8.439×10^{-5}
$3p_{1/2}$	4.066×10^{-10}	1.120×10^{-5}	5.090×10^{-6}
$3p_{3/2}$	8.354×10^{-11}	9.702×10^{-13}	6.020×10^{-12}
$3d_{3/2}$	8.357×10^{-11}	1.275×10^{-13}	1.558×10^{-14}
$3d_{5/2}$	2.645×10^{-11}	2.145×10^{-14}	3.394×10^{-16}
$4s_{1/2}$	6.127×10^{-10}	1.990×10^{-5}	3.421×10^{-5}
$4p_{1/2}$	3.258×10^{-10}	4.779×10^{-6}	2.167×10^{-6}
$4p_{3/2}$	7.983×10^{-11}	4.833×10^{-13}	2.720×10^{-12}
$4d_{3/2}$	7.981×10^{-11}	1.351×10^{-13}	9.632×10^{-15}
$4d_{5/2}$	3.460×10^{-11}	6.875×10^{-14}	5.877×10^{-16}
$4f_{5/2}$	3.519×10^{-11}	1.918×10^{-14}	2.731×10^{-16}
$4f_{7/2}$	1.452×10^{-11}	2.122×10^{-14}	1.373×10^{-16}
$5s_{1/2}$	4.986×10^{-10}	1.018×10^{-5}	1.692×10^{-5}
$5p_{1/2}$	2.690×10^{-10}	2.422×10^{-6}	1.095×10^{-6}

Continued on next page

Table 4.4.: Deviation of the results obtained from `dish` and GRASP2018 for states of hydrogenic uranium up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 1.000 \times 10^{-8}, h = 0.001$ and $r_{\max} = 5$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$5p_{3/2}$	7.219×10^{-11}	2.683×10^{-13}	1.415×10^{-12}
$5d_{3/2}$	7.205×10^{-11}	8.065×10^{-14}	5.687×10^{-15}
$5d_{5/2}$	3.616×10^{-11}	4.610×10^{-14}	4.663×10^{-16}
$5f_{5/2}$	3.616×10^{-11}	2.501×10^{-14}	3.252×10^{-16}
$5f_{7/2}$	1.857×10^{-11}	2.748×10^{-14}	1.342×10^{-16}
$5g_{7/2}$	1.934×10^{-11}	1.563×10^{-14}	8.548×10^{-17}
$5g_{9/2}$	9.532×10^{-12}	1.252×10^{-14}	8.068×10^{-17}

JAC

The Jena Atomic Calculator (JAC) is a rather new package for performing atomic calculations published in 2019 and until today under active development. It is written in Julia which allows for a more user-friendly syntax and better code-readability while maintaining a high performance. JAC also offers a wide range of tools to compute various atomic properties, processes cascades. For details refer to the manual [35] or have a look at the tutorials available next to the source code at `https://github.com/OpenJAC/JAC.jl`.
A code example to compute a wave function of a hydrogen-like system is shown in chapter A.3.2.

Analogue to the comparison with GRASP, in tables 4.5, 4.6 and 4.7 the energies and wave functions from `dish` are compared to those obtained using JAC for the three test systems ${}^{1}_{1}\text{H}$, ${}^{40}_{20}\text{Ca}^{19+}$ and ${}^{238}_{92}\text{U}^{91+}$ respectively. Here the consistency for all systems is worse compared to GRASP and no trends are observable. Anyhow, as $\epsilon_{\text{rel}}(E) < 5 \times 10^{-4}$, $\epsilon_{\text{mse}}(f) < 3 \times 10^{-5}$ and $\epsilon_{\text{mse}}(g) < 4 \times 10^{-5}$ the results match reasonable well. The larger discrepancy is probably a result of the different solving method used in JAC which results in small oscillations of both components of the wave function near the origin. Adjusting the parameters of the grid, which is shown in chapter A.3.2, helps to minimize these non-physical oscillations.
Note that here the nucleus for hydrogen is modeled as a homogeneously charged sphere to show that also using this model the agreement of the results is good.

Table 4.5.: Deviation of the results obtained from `dish` and JAC for states of hydrogen up to $5g_{9/2}$ modeled by a uniformly charged sphere-like nucleus with the radius calculated from R_{rms} from table 4.1 using eq. (2.84). The grid for `dish` was constructed using $r_0 = 2.000 \times 10^{-6}$, $h = 0.01$ and $r_{\mathrm{max}} = 615$.

state	$\epsilon_{\mathrm{rel}}(E)$	$\epsilon_{\mathrm{mse}}(f)$	$\epsilon_{\mathrm{mse}}(g)$
$1s_{1/2}$	1.233×10^{-9}	2.335×10^{-11}	3.061×10^{-6}
$2s_{1/2}$	1.248×10^{-7}	4.557×10^{-10}	5.214×10^{-7}
$2p_{1/2}$	7.021×10^{-8}	1.596×10^{-10}	1.934×10^{-16}
$2p_{3/2}$	2.953×10^{-8}	1.663×10^{-10}	1.756×10^{-7}
$3s_{1/2}$	7.365×10^{-7}	6.257×10^{-9}	1.677×10^{-7}
$3p_{1/2}$	1.192×10^{-6}	2.741×10^{-9}	1.097×10^{-15}
$3p_{3/2}$	3.924×10^{-7}	3.222×10^{-9}	6.709×10^{-8}
$3d_{3/2}$	2.711×10^{-7}	4.814×10^{-10}	6.709×10^{-8}
$3d_{5/2}$	2.661×10^{-7}	6.332×10^{-10}	3.354×10^{-8}
$4s_{1/2}$	5.059×10^{-5}	8.009×10^{-8}	7.430×10^{-8}
$4p_{1/2}$	4.309×10^{-5}	5.166×10^{-8}	7.430×10^{-8}
$4p_{3/2}$	2.530×10^{-5}	5.511×10^{-8}	1.159×10^{-14}
$4d_{3/2}$	1.732×10^{-5}	1.931×10^{-8}	8.153×10^{-15}
$4d_{5/2}$	4.441×10^{-6}	2.316×10^{-8}	1.769×10^{-8}
$4f_{5/2}$	2.707×10^{-6}	2.855×10^{-9}	9.756×10^{-16}
$4f_{7/2}$	8.032×10^{-7}	4.199×10^{-9}	1.061×10^{-8}
$5s_{1/2}$	3.451×10^{-4}	4.319×10^{-7}	3.905×10^{-8}
$5p_{1/2}$	3.554×10^{-4}	3.570×10^{-7}	3.909×10^{-8}
$5p_{3/2}$	2.536×10^{-4}	3.497×10^{-7}	1.695×10^{-13}
$5d_{3/2}$	2.231×10^{-4}	2.079×10^{-7}	1.121×10^{-13}
$5d_{5/2}$	1.207×10^{-4}	2.206×10^{-7}	5.127×10^{-14}
$5f_{5/2}$	7.749×10^{-5}	6.862×10^{-8}	1.015×10^{-8}
$5f_{7/2}$	1.461×10^{-5}	9.087×10^{-8}	6.523×10^{-9}
$5g_{7/2}$	9.805×10^{-6}	8.309×10^{-9}	6.521×10^{-9}
$5g_{9/2}$	5.730×10^{-6}	1.555×10^{-8}	4.505×10^{-14}

Table 4.6.: Deviation of the results obtained from `dish` and JAC for states of hydrogenic calcium up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 2.000 \times 10^{-6}, h = 0.05$ and $r_{\max} = 615$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$1s_{1/2}$	4.074×10^{-9}	5.892×10^{-10}	1.010×10^{-8}
$2s_{1/2}$	1.956×10^{-9}	6.842×10^{-9}	1.160×10^{-9}
$2p_{1/2}$	2.033×10^{-8}	2.261×10^{-9}	2.212×10^{-12}
$2p_{3/2}$	8.340×10^{-9}	2.484×10^{-9}	2.616×10^{-12}
$3s_{1/2}$	2.420×10^{-6}	1.183×10^{-7}	3.651×10^{-10}
$3p_{1/2}$	2.010×10^{-6}	6.754×10^{-8}	2.923×10^{-11}
$3p_{3/2}$	1.053×10^{-6}	6.838×10^{-8}	1.888×10^{-11}
$3d_{3/2}$	5.356×10^{-7}	1.666×10^{-8}	6.398×10^{-12}
$3d_{5/2}$	1.785×10^{-7}	1.791×10^{-8}	2.464×10^{-12}
$4s_{1/2}$	1.901×10^{-5}	5.644×10^{-7}	4.992×10^{-10}
$4p_{1/2}$	2.612×10^{-6}	2.882×10^{-7}	1.938×10^{-10}
$4p_{3/2}$	1.816×10^{-5}	4.129×10^{-7}	3.001×10^{-10}
$4d_{3/2}$	4.237×10^{-6}	1.250×10^{-7}	8.394×10^{-11}
$4d_{5/2}$	6.421×10^{-6}	1.504×10^{-7}	1.118×10^{-10}
$4f_{5/2}$	1.564×10^{-6}	3.425×10^{-8}	1.881×10^{-11}
$4f_{7/2}$	3.841×10^{-7}	3.147×10^{-8}	1.458×10^{-11}
$5s_{1/2}$	2.152×10^{-4}	7.436×10^{-6}	1.064×10^{-9}
$5p_{1/2}$	1.184×10^{-5}	3.097×10^{-6}	4.311×10^{-10}
$5p_{3/2}$	2.896×10^{-4}	5.771×10^{-6}	1.090×10^{-9}
$5d_{3/2}$	6.905×10^{-6}	9.209×10^{-7}	2.267×10^{-10}
$5d_{5/2}$	1.987×10^{-4}	2.629×10^{-6}	8.765×10^{-10}
$5f_{5/2}$	1.119×10^{-5}	2.479×10^{-7}	1.217×10^{-10}
$5f_{7/2}$	5.433×10^{-5}	6.129×10^{-7}	3.119×10^{-10}
$5g_{7/2}$	5.203×10^{-6}	8.184×10^{-8}	3.289×10^{-11}
$5g_{9/2}$	2.471×10^{-6}	7.848×10^{-8}	3.096×10^{-11}

Table 4.7.: Deviation of the results obtained from `dish` and JAC for states of hydrogenic uranium up to $5g_{9/2}$ modeled by a Fermi-like nucleus with the parameters from table 4.1. The grid for `dish` was constructed using $r_0 = 2.000 \times 10^{-6}, h = 0.05$ and $r_{\max} = 150$.

state	$\epsilon_{\text{rel}}(E)$	$\epsilon_{\text{mse}}(f)$	$\epsilon_{\text{mse}}(g)$
$1s_{1/2}$	1.202×10^{-7}	2.294×10^{-9}	1.452×10^{-6}
$2s_{1/2}$	1.236×10^{-7}	2.731×10^{-8}	2.383×10^{-7}
$2p_{1/2}$	2.785×10^{-8}	8.618×10^{-9}	8.131×10^{-11}
$2p_{3/2}$	2.126×10^{-8}	1.339×10^{-8}	9.365×10^{-11}
$3s_{1/2}$	2.401×10^{-7}	2.296×10^{-7}	7.046×10^{-8}
$3p_{1/2}$	2.686×10^{-7}	1.037×10^{-7}	1.527×10^{-9}
$3p_{3/2}$	3.542×10^{-7}	2.697×10^{-7}	1.536×10^{-9}
$3d_{3/2}$	1.662×10^{-7}	4.033×10^{-8}	2.063×10^{-10}
$3d_{5/2}$	1.988×10^{-7}	7.024×10^{-8}	4.628×10^{-10}
$4s_{1/2}$	1.882×10^{-5}	5.150×10^{-6}	3.236×10^{-8}
$4p_{1/2}$	1.764×10^{-5}	2.793×10^{-6}	5.003×10^{-9}
$4p_{3/2}$	1.941×10^{-5}	4.468×10^{-6}	7.884×10^{-9}
$4d_{3/2}$	1.330×10^{-5}	1.556×10^{-6}	5.518×10^{-9}
$4d_{5/2}$	4.944×10^{-6}	1.972×10^{-6}	2.158×10^{-9}
$4f_{5/2}$	2.463×10^{-6}	2.709×10^{-7}	9.325×10^{-10}
$4f_{7/2}$	2.022×10^{-7}	4.099×10^{-7}	4.130×10^{-10}
$5s_{1/2}$	3.075×10^{-4}	4.110×10^{-5}	1.147×10^{-7}
$5p_{1/2}$	2.749×10^{-4}	2.896×10^{-5}	9.273×10^{-8}
$5p_{3/2}$	2.018×10^{-4}	2.797×10^{-5}	1.213×10^{-7}
$5d_{3/2}$	1.856×10^{-4}	1.683×10^{-5}	7.865×10^{-8}
$5d_{5/2}$	9.704×10^{-5}	1.649×10^{-5}	4.512×10^{-8}
$5f_{5/2}$	6.906×10^{-5}	5.875×10^{-6}	2.485×10^{-8}
$5f_{7/2}$	2.155×10^{-5}	7.244×10^{-6}	6.640×10^{-9}
$5g_{7/2}$	1.016×10^{-5}	8.496×10^{-7}	2.952×10^{-9}
$5g_{9/2}$	2.787×10^{-6}	1.569×10^{-6}	1.262×10^{-9}

4.1.3. Muonic Atoms

A simple yet exotic atomic system is a hydrogen-like muonic atom where a single negatively charged muon μ^- is bound to a positively charged nucleus. The muon is very similar to an electron as it also is a spin $\frac{1}{2}$ particle with charge $q = -1$ (a.u.) while having a mass of $m_{\mu^-} = 105.658\,\text{MeV} \approx 207$ (a.u.) [17]. Because of this the muonic orbitals are closer to the nucleus, some low orbitals even penetrate the nucleus. Therefore, muonic atoms can be used to investigate nuclear parameters like size and charge distribution [39].

Since μ^- is a spin $\frac{1}{2}$ particle it can be described using the DE eq. (2.70) and the solving toolkit developed in chapter 3 can be used to find muonic wave functions and the energies of the muonic states.
Here, the energies found using `dish` are compared to some analytical calculations from [40] for hydrogenic $^{185}_{75}\mathrm{Re}^{74+}$ which is modeled at first having a point-like and at second a finite sized nucleus. Using a finite-size nucleus has a large impact on the energy on low-lying states which have a high probability density near or even in the nucleus.
The resulting energies along the corresponding errors are shown in tables 4.8 and 4.9. In both cases the results obtained using `dish` align with the analytic results well as $\epsilon_{\mathrm{rel}} < 3 \times 10^{-4}$ for all states up to $3d_{5/2}$.

Table 4.8.: Energies of the states up to $n = 3$ of hydrogen-like muonic $^{185}_{75}\mathrm{Re}^{74+}$ modeled by a point-like nucleus calculated by `dish` and from [40] (E_{ref}). The grid for `dish` was constructed using $r_0 = 10^{-8}, h = 0.01$ and $r_{\mathrm{max}} = 0.01$.

	E_{dish} (in MeV)	E_{ref} (in MeV)	$\epsilon_{\mathrm{abs}}(E)$	$\epsilon_{\mathrm{rel}}(E)$
$1s_{1/2}$	−17.2291	−17.2286	-4.63×10^{-4}	2.69×10^{-5}
$2s_{1/2}$	−4.3988	−4.3988	-3.34×10^{-5}	7.60×10^{-6}
$2p_{1/2}$	−4.3988	−4.3988	-3.34×10^{-5}	7.60×10^{-6}
$2p_{3/2}$	−4.0331	−4.0328	-2.57×10^{-4}	6.38×10^{-5}
$3s_{1/2}$	−1.9130	−1.9129	-6.02×10^{-5}	3.15×10^{-5}
$3p_{1/2}$	−1.9130	−1.9129	-6.02×10^{-5}	3.15×10^{-5}
$3p_{3/2}$	−1.8040	−1.8039	-1.04×10^{-4}	5.77×10^{-5}
$3d_{3/2}$	−1.8040	−1.8039	-1.04×10^{-4}	5.77×10^{-5}
$3d_{5/2}$	−1.7731	−1.7730	-1.38×10^{-4}	7.79×10^{-5}

Table 4.9.: Energies of the states up to $n = 3$ of hydrogen-like muonic $^{185}_{75}\mathrm{Re}^{74+}$ modeled by a sphere-like nucleus calculated by `dish` and from [40] (E_{ref}). The grid for `dish` was constructed using $r_0 = 10^{-8}, h = 0.01$ and $r_{\mathrm{max}} = 0.01$.

	E_{dish} (in MeV)	E_{ref} (in MeV)	$\epsilon_{\mathrm{abs}}(E)$	$\epsilon_{\mathrm{rel}}(E)$
$1s_{1/2}$	−9.2869	−9.2845	-2.43×10^{-3}	2.62×10^{-4}
$2s_{1/2}$	−3.0711	−3.0708	-2.64×10^{-4}	8.58×10^{-5}
$2p_{1/2}$	−4.0352	−4.0355	2.60×10^{-4}	6.44×10^{-5}
$2p_{3/2}$	−3.8896	−3.8896	8.38×10^{-6}	2.15×10^{-6}
$3s_{1/2}$	−1.4934	−1.4934	3.33×10^{-5}	2.23×10^{-5}
$3p_{1/2}$	−1.7904	−1.7905	9.12×10^{-5}	5.09×10^{-5}
$3p_{3/2}$	−1.7525	−1.7525	-1.38×10^{-5}	7.86×10^{-6}
$3d_{3/2}$	−1.8022	−1.8021	-1.18×10^{-4}	6.52×10^{-5}
$3d_{5/2}$	−1.7725	−1.7723	-1.76×10^{-4}	9.94×10^{-5}

Additionally to finite-size effects, energy shifts are induced by QED effects. Due to the larger mass of the muon m_μ, in comparison with the electron's mass m_e, and many QED terms being functions of m/M where M is the mass of the nucleus, these energy shifts are larger in muonic systems than in electronic systems [41]. To quantify the QED influence, the energy difference between the $2s_{1/2}$ and $2p_{1/2}$ levels, induced by finite size and QED effects, is shown in table 4.10 for several hydrogen-like muonic systems. The values for the shift caused by the finite size were calculated by Robert Waltner ($\Delta E^{(a)}_{\text{fns}}$) [42, table 3.6], by Feinberg et al. ($\Delta E^{(b)}_{\text{fns}}$) [43] and using `dish` ($\Delta E^{\texttt{dish}}_{\text{fns}}$). The values from this thesis $\Delta E^{\texttt{dish}}_{\text{fns}}$ agree well with $\Delta E^{(a)}_{\text{fns}}$ that were obtained by solving the DE numerically using a B-Spline approach. The order of magnitude also aligns with $\Delta E^{(b)}_{\text{fns}}$ for all systems and for heavy systems the values correspond very well with $\epsilon_{\text{rel}} < 0.01$. In the last column in table 4.10 shifts induced by QED effects are also incorporated. These values deviate greatly for light systems, as here m_μ/M is not negligible, while for heavy systems $\Delta E_{\text{fns}} \approx \Delta E_{\text{fns + QED}}$. This reflects the fact that the shift for light systems is dominated by QED effects while for heavy systems their impact is smaller than the finite size shift [42, fig. (3.2)].
As calculations performed with `dish` are purely quantum mechanical and do not take QED effects into account, other tools might be preferable to perform computations regarding light muonic atoms with greater accuracy.

Table 4.10.: Energy difference $\Delta E = E(2s_{1/2}) - E(2p_{1/2})$ between the $2s_{1/2}$ and $2p_{1/2}$ states at muonic atoms modeled with a finite size nucleus. In the last column $\Delta E^{(a)}_{\text{fns + QED}}$ additionally QED effects are taken into account. The column marked with $^{\texttt{dish}}$ has been calculated in this thesis, while those marked with $^{(a)}$ were calculated by Robert Waltner [42] and the $^{(b)}$ values are from Feinberg et al. [43].

	$\Delta E^{\texttt{dish}}_{\text{fns}}$	$\Delta E^{(a)}_{\text{fns}}$	$\Delta E^{(b)}_{\text{fns}}$	$\Delta E^{(a)}_{\text{fns + QED}}$
$^{12}_{6}C^{5+}$	5.127×10^{1}	5.159×10^{1}	3.84×10^{1}	1.692×10^{2}
$^{23}_{11}Na^{10+}$	7.572×10^{2}	7.635×10^{2}	7.27×10^{2}	1.159×10^{3}
$^{35}_{17}Cl^{16+}$	4.698×10^{3}	4.744×10^{3}	4.64×10^{3}	5.689×10^{3}
$^{56}_{26}Fe^{25+}$	2.522×10^{4}	2.552×10^{4}	2.60×10^{4}	2.773×10^{4}
$^{72}_{32}Ge^{31+}$	5.725×10^{4}	5.825×10^{4}	5.76×10^{4}	6.159×10^{4}
$^{79}_{35}Br^{34+}$	7.963×10^{4}	8.063×10^{4}	8.08×10^{4}	8.464×10^{4}
$^{92}_{42}Mo^{41+}$	1.500×10^{5}	1.553×10^{5}	1.54×10^{5}	1.610×10^{5}
$^{120}_{50}Sn^{49+}$	2.772×10^{5}	2.809×10^{5}	2.81×10^{5}	2.891×10^{5}
$^{142}_{60}Nd^{59+}$	4.947×10^{5}	5.016×10^{5}	4.99×10^{5}	5.134×10^{5}
$^{184}_{74}W^{73+}$	9.189×10^{5}	9.315×10^{5}	9.23×10^{5}	9.494×10^{5}
$^{208}_{82}Pb^{81+}$	1.202×10^{6}	1.219×10^{6}	1.21×10^{6}	1.241×10^{6}

4.1.4. Isotope Shift

Modeling the nucleus of the system having a finite size, results in an energy shift for the s-orbitals which have a non-zero probability density at the origin. This was already observed in the last results. Following [44] the energy corrections are expanded in orders of αZ. The lowest order contribution is given by

$$E^{(4)}_{\text{nucl}} = \frac{2}{3} c^2 \frac{(Z\alpha)^4}{n^3} m_r^3 \left(\frac{r_{\text{rms}}}{\lambda\!\!\!^{-}_C} \right)^3 \delta_{l0}\,, \tag{4.5}$$

where m_r is the reduced mass, $\lambda\!\!\!^{-}_C = \hbar/c$ the reduced Compton wavelength and r_{rms} the root-mean-square radius of the nucleus. This order is induced only by the finite nuclear size expressed in r_{rms}. The next order in $(\alpha Z)^5$ has contributions of both, the finite size and nuclear polarizability, while `dish` does not take into account the latter one. Because of that here the effect is calculated only for hydrogen, where $\alpha Z \ll 1$, and, therefore, only the first order will be considered.

The finite size contributions are calculated numerically by solving the DE for a point-like nucleus and a nucleus that is modeled as a homogeneously charged sphere. Then the energy difference $\Delta E_{FS} = E_{\text{point-like}} - E_{\text{sphere-like}}$ is calculated. In table 4.11 these results and the reference values calculated from eq. (4.5) are displayed for hydrogen s-states up to $n = 30$. The results match good as for all these states the relative error is $\epsilon_{\text{rel}}(\Delta E_{FS}) < 6.6 \times 10^{-4}$. The relative error being very similar for all the states, indicates that the dominant error is a systematic one and not yet numerical. Possible reasons are that higher order terms are not included and that `dish` only solves the DE for a static nucleus and, therefore, the reduced mass is 1.

Table 4.11.: Finite size energy shift ΔE_{FS} for the s states up to $n = 30$ of hydrogen modeled by a uniformly charged sphere-like nucleus calculated by `dish` and calculated from eq. (4.5). The grid for `dish` was constructed using $r_0 = 10^{-8}, h = 0.01$ and $r_{\text{max}} = 0.01$.

state	$\Delta E_{\text{FS, dish}}$	$\Delta E_{\text{FS, ref}}$	$\epsilon_{\text{rel}}(\Delta E_{FS})$
$1s_{1/2}$	1.8376×10^{-10}	1.8365×10^{-10}	5.7300×10^{-4}
$2s_{1/2}$	2.2971×10^{-11}	2.2956×10^{-11}	6.3043×10^{-4}
$3s_{1/2}$	6.8061×10^{-12}	6.8019×10^{-12}	6.2725×10^{-4}
$4s_{1/2}$	2.8713×10^{-12}	2.8695×10^{-12}	6.2559×10^{-4}
$5s_{1/2}$	1.4701×10^{-12}	1.4692×10^{-12}	6.2967×10^{-4}
$6s_{1/2}$	8.5077×10^{-13}	8.5023×10^{-13}	6.3133×10^{-4}
$7s_{1/2}$	5.3577×10^{-13}	5.3542×10^{-13}	6.4755×10^{-4}
$8s_{1/2}$	3.5892×10^{-13}	3.5869×10^{-13}	6.2801×10^{-4}
$9s_{1/2}$	2.5208×10^{-13}	2.5192×10^{-13}	6.1157×10^{-4}

Continued on next page

Table 4.11.: Finite size energy shift ΔE_{FS} for the s states up to $n = 30$ of hydrogen modeled by a uniformly charged sphere-like nucleus calculated by `dish` and calculated from eq. (4.5). The grid for `dish` was constructed using $r_0 = 10^{-8}, h = 0.01$ and $r_{\max} = 0.01$.

state	$\Delta E_{\text{FS, dish}}$	$\Delta E_{\text{FS, ref}}$	$\epsilon_{\text{rel}}(\Delta E_{FS})$
$10s_{1/2}$	1.8377×10^{-13}	1.8365×10^{-13}	6.3439×10^{-4}
$11s_{1/2}$	1.3807×10^{-13}	1.3798×10^{-13}	6.2612×10^{-4}
$12s_{1/2}$	1.0635×10^{-13}	1.0628×10^{-13}	6.3541×10^{-4}
$13s_{1/2}$	8.3645×10^{-14}	8.3592×10^{-14}	6.3861×10^{-4}
$14s_{1/2}$	6.6971×10^{-14}	6.6928×10^{-14}	6.3783×10^{-4}
$15s_{1/2}$	5.4448×10^{-14}	5.4415×10^{-14}	6.0252×10^{-4}
$16s_{1/2}$	4.4866×10^{-14}	4.4837×10^{-14}	6.4735×10^{-4}
$17s_{1/2}$	3.7404×10^{-14}	3.7381×10^{-14}	6.3067×10^{-4}
$18s_{1/2}$	3.1510×10^{-14}	3.1490×10^{-14}	6.1501×10^{-4}
$19s_{1/2}$	2.6793×10^{-14}	2.6775×10^{-14}	6.5232×10^{-4}
$20s_{1/2}$	2.2971×10^{-14}	2.2956×10^{-14}	6.2967×10^{-4}
$21s_{1/2}$	1.9843×10^{-14}	1.9831×10^{-14}	6.2123×10^{-4}
$22s_{1/2}$	1.7258×10^{-14}	1.7247×10^{-14}	6.0727×10^{-4}
$23s_{1/2}$	1.5103×10^{-14}	1.5094×10^{-14}	6.1783×10^{-4}
$24s_{1/2}$	1.3293×10^{-14}	1.3285×10^{-14}	6.3133×10^{-4}
$25s_{1/2}$	1.1761×10^{-14}	1.1754×10^{-14}	6.4428×10^{-4}
$26s_{1/2}$	1.0455×10^{-14}	1.0449×10^{-14}	6.1786×10^{-4}
$27s_{1/2}$	9.3361×10^{-15}	9.3304×10^{-15}	6.0554×10^{-4}
$28s_{1/2}$	8.3712×10^{-15}	8.3660×10^{-15}	6.2487×10^{-4}
$29s_{1/2}$	7.5349×10^{-15}	7.5301×10^{-15}	6.4049×10^{-4}
$30s_{1/2}$	6.8062×10^{-15}	6.8019×10^{-15}	6.3439×10^{-4}

4.2. Matrix-Elements and Expectation Values

By knowledge of the wave functions of electronic states $|n_i\kappa_i\rangle$ one can calculate matrix elements

$$\langle n_1\kappa_1 m|\hat{o}|n_2\kappa_2 m\rangle = \int \psi^*_{n_1\kappa_1 m_1}\hat{o}(r,\theta,\phi)\psi_{n_2\kappa_2 m_2}\mathrm{d}\mathbf{r}\,, \tag{4.6}$$

where $\hat{o}$ is an operator. And by constructing the wave functions from the radial parts and the spinors, one could calculate this for any operator, but often the problem can be separated into radial and spherical integrals or can rephrase the problem, e.g. using Wigner-Eckart's theorem [14, ch. 3], such that a radial matrix element of the form

$$\langle n_1\kappa_1|\hat{o}_r|n_2\kappa_2\rangle = \int_0^\infty \begin{pmatrix} -if_{n_1\kappa_1}(r) & g_{n_1\kappa_1}(r)\end{pmatrix} \hat{o}_r(r) \begin{pmatrix} if_{n_2\kappa_2}(r) \\ g_{n_2\kappa_2}(r)\end{pmatrix} \mathrm{d}r \tag{4.7}$$

is to be calculated. The integral needs to be evaluated numerically since the radial wave functions usually have no analytic form, the calculation would be computationally very expensive, or there is no analytic solution at all. To achieve the best precision, the DE should be solved on a `dish.dirac.radial.grid.grid.RombergIntegrationGrid` as discussed in chapter 3.7.

Table 4.12.: Radial Moments $\langle (2Zr)^s \rangle$ for hydrogenic bound states [45].

s	non-relativistic	relativistic
2	$2n^2\left[5n^2+1-3l(l+1)\right]$	$2\Big[N^2\left(5N^2-2\kappa^2\right)\left(1-\frac{Z^2}{N^2c^2}\right)$ $+N^2\left(1-\gamma^2\right)-3\kappa N^2\sqrt{1-\frac{Z^2}{N^2c^2}}\Big]$
1	$3n^2-l(l-1)$	$\left(3N^2-\kappa^2\right)\sqrt{1-\frac{Z^2}{N^2c^2}}-\kappa$
-1	$\frac{1}{2n^2}$	$n\gamma+\left(\lvert\kappa\rvert-\gamma\right)\lvert\kappa\rvert/\left(2\gamma N^3\right)$

As a benchmark the results for some scaled radial moments $\langle (2Zr)^s \rangle$ are compared to the analytic solutions for a point-like nucleus shown in table 4.12. In `dish` the wave functions from which the expectation values are evaluated are also calculated for a point-like nucleus. The used grid was a `RombergIntegrationGrid` which was constructed from a `DistanceGrid` (see chapter 5.2.3 for more details) with the parameters `r0=1e-6` and `h=1e-3`.

In tables 4.13 and 4.14 the analytic values calculated from table 4.12 and the results calculated with `dish` are shown next to the relative errors. The states $1s_{1/2}$, $3p_{1/2}$ and $10d_{5/2}$ are calculated for ${}^{1}_{1}\mathrm{H}$ and ${}^{238}_{92}\mathrm{U}^{91+}$ as examples to show the behavior for low-lying states with zero or just a few nodes as well as for a higher state for both, a lightweight and a heavy nucleus. For the low states of both nuclei the results coincide very well with the analytical results, as for hydrogen $\epsilon_{\mathrm{rel}}(\langle (2Zr)^s \rangle) < 10^{-14}$ and for uranium $\epsilon_{\mathrm{rel}}(\langle (2Zr)^s \rangle) < 10^{-10}$. For higher states like the shown $10d_{5/2}$, the values agree not as good but are still a reasonable good approximation with $\epsilon_{\mathrm{rel},H}(\langle (2Zr)^s \rangle) < 2 \times 10^{-2}$ and $\epsilon_{\mathrm{rel},U}(\langle (2Zr)^s \rangle) < 2 \times 10^{-3}$.

In the appendix in chapter A.4 in tables A.1 and A.2 the scaled radial moments for the same systems for the corresponding non-relativistic states are shown. The findings are similar, except for the coincidence of the radial moments of the $1s$ state is about five orders of magnitude worse for both systems. With $\epsilon_{\mathrm{rel},H}(\langle (2Zr)^s \rangle) < 10^{-9}$ and $\epsilon_{\mathrm{rel},U}(\langle (2Zr)^s \rangle) < 2 \times 10^{-4}$ one should keep this in mind for non-relativistic calculations especially for heavier nuclei.

Table 4.13.: Scaled radial moments $\langle (2Zr)^s \rangle$ of the relativistic states $1s_{1/2}, 3p_{1/2}$ and $10d_{5/2}$ of point-like hydrogen on a `RombergIntegrationGrid` similar to a `DistanceGrid` defined by `r0=1e-6`, `h=1e-3` and `r_max=250`.

state	s	$\langle r^s \rangle_{\text{ref}}$	$\langle r^s \rangle_{\text{dish}}$	$\epsilon_{\text{rel}}(\langle r^s \rangle)$
$1s_{1/2}$	2	1.200×10^{1}	1.200×10^{1}	3.701×10^{-15}
	1	3.000	3.000	1.776×10^{-15}
	-1	5.000×10^{-1}	5.000×10^{-1}	3.109×10^{-15}
$3p_{1/2}$	2	7.200×10^{2}	7.200×10^{2}	6.000×10^{-15}
	1	2.500×10^{1}	2.500×10^{1}	2.984×10^{-15}
	-1	5.556×10^{-2}	5.556×10^{-2}	3.372×10^{-15}
$10d_{5/2}$	2	9.660×10^{4}	9.523×10^{4}	1.435×10^{-2}
	1	2.940×10^{2}	2.919×10^{2}	7.049×10^{-3}
	-1	5.000×10^{-3}	5.040×10^{-3}	7.990×10^{-3}

Table 4.14.: Scaled radial moments $\langle (2Zr)^s \rangle$ of the relativistic states $1s_{1/2}, 3p_{1/2}$ and $10d_{5/2}$ of hydrogen-like uranium modeled by a point-like nucleus on a `RombergIntegrationGrid` similar to a `DistanceGrid` defined by `r0=1e-6`, `h=1e-3` and `r_max=5`.

state	s	$\langle r^s \rangle_{\text{ref}}$	$\langle r^s \rangle_{\text{dish}}$	$\epsilon_{\text{rel}}(\langle r^s \rangle)$
$1s_{1/2}$	2	8.644	8.644	2.424×10^{-12}
	1	2.482	2.482	1.416×10^{-12}
	-1	6.746×10^{-1}	6.746×10^{-1}	1.678×10^{-11}
$3p_{1/2}$	2	5.291×10^{2}	5.291×10^{2}	1.824×10^{-13}
	1	2.124×10^{1}	2.124×10^{1}	9.887×10^{-14}
	-1	7.450×10^{-2}	7.450×10^{-2}	1.291×10^{-12}
$10d_{5/2}$	2	9.417×10^{4}	9.401×10^{4}	1.722×10^{-3}
	1	2.901×10^{2}	2.899×10^{2}	8.078×10^{-4}
	-1	5.121×10^{-3}	5.125×10^{-3}	8.417×10^{-4}

5. Using dish

The source code for dish is open source (distributed under MIT license) and is available on GitHub at https://github.com/suppetia/qm-dish. Here a short introduction will be given on how to get started using dish. A more extensive documentation is available online at https://suppetia.github.io/qm-dish/ which is also distributed next to the software as a PDF at https://github.com/suppetia/qm-dish/blob/main/manual_dish.pdf.

dish is implemented in Python due to its popularity in the scientific community and easy to read source code. Python offers a great flexibility as there are many packages for various applications, and it has a very active community. The language was designed with simplicity for the user in mind and allows for an easy entrance into scripting/programming because of the simple syntax. As Python is an interpreted language the execution speed is rather slow in comparison with compiled languages. For increasing the performance the standard numeric and scientific packages numpy and scipy are mainly implemented in C and Fortran and offer Python bindings. [46]
In dish, the most time-consuming part, which is the Adams-Moulton procedure, is implemented in two versions. A version in pure Python (actually already utilizing numpy and scipy) exists for compatibility, but as discussed in chapter 4.1.1 it is highly recommended to use the version that implemented the AM procedure in Fortran.
In principle, Python is object-oriented which allows implementing structures in a way that is natural to our thinking as objects are assigned properties and functionalities. On the other hand, Python also allows using functional programming where instead the objects are passed to functions which handle their behavior. In dish this combination was used to generate a script flow that is close to human thinking: At first one defines the system via the nuclear parameters, the grid on which the calculations are performed and the states which should be calculated. And at second one passes this information to a function which actually solves the SE/DE. Having calculated some results using dish, one can make use of the rich ecosystem of Python packages to continue processing these results like e.g. use one's favorite plotting package to display the findings.

5.1. Installation

It should be possible to install dish under all operating systems but only Windows and Unix-based systems are tested, and at the moment an automated installation is only possible on Unix-based systems. Here, just the installation process for the Fortran-accelerated version on Unix-based systems will be covered but a short instruction on how to install dish without compiling the Fortran code (e.g. on Windows) can be found in chapter A.5.

First make sure to have a working Python installation based on CPython 3.7 or above, but below 3.12 as `gmpy2` currently does not support Python 3.12. Next to this, also the python virtual environment package `venv` and the python packages `pip`, `build` and `numpy` are required for the building process. To compile the Fortran source code a Fortran compiler like `gfortran`[1] needs to be available.
If all the prerequirements are fulfilled, clone the repository from GitHub using

```
git clone https://github.com/suppetia/qm-dish.git
# or via ssh
git clone git@github.com:suppetia/qm-dish.git
```

or download the source code.
Then change into the repository and run in a shell

```
make
```

to build the package and compile the Fortran extension. To install the package run

```
make install
```

from the same location. To use a specific Python executable or Fortran compiler, modify the settings in the first few lines of the *Makefile* which is located in the root directory of the repository.
If the installation was successful running

```
python3 -c "import dish; print(dish.__version__)"
```

should yield a similar output to `0.1.0`.

5.2. Getting Started

In this section the most important parts of `dish` are covered. In chapter A.7 in the appendix two full examples using `dish` for finding wave functions will be discussed while more examples and more details to the respective classes/functions can be found in the manual or online.
Throughout this chapter the pseudo-Python notation

```
function(argument1: type,
         argument2: type = <optional:default_value>)
```

will be used to illustrate the usage of the functions and classes. If for optional arguments a default value is given, usually it is a value that has proven to work reliably, if not the use of this argument is discussed explicitly.

[1] For the automated install `gfortran` is expected, but this can be changed in the *Makefile*.

5.2.1. Conversion of Units

All calculations in `dish` are performed dimensionless in Hartree atomic units and, hence, all classes and functions expect input parameters in atomic units. To convert from and to atomic units `dish` provides the utility function

```
from dish import convert_units
convert_units(old_unit: str|float,
              new_unit: str|float,
              value: float = <optional:1.>,
              old_unit_exp: float = <optional:1>,
              new_unit_exp: float = <optional:1>
             )
```

which calculates

$$\text{new\_value} = \frac{(\text{value} \cdot \text{new\_unit})^{\text{old\_unit\_exp}}}{\text{new\_unit}^{\text{new\_unit\_exp}}} .$$

The fields `old_unit` and `new_unit` expect the alias of the old/new unit or alternatively a conversion factor to convert to the corresponding SI units of the same dimension. There are no checks of the dimension performed! Make sure to only convert units of matching dimensions.

5.2.2. Defining the Hydrogenic System

The hydrogenic system is defined by the properties of the nucleus and the model used to describe its potential. The nucleus has a charge Z and can be modeled static, when its mass is infinitely heavy, or having a finite mass M. As discussed in chapter 2.5 in `dish` the nucleus can be modeled point-like, sphere-like or its charge density can be described using a Fermi distribution. The latter ones have a finite size which is characterized by the radius of the sphere R_0 in the sphere-like model or by the parameter c and the diffuseness a for the Fermi model. These parameters are related with the root-mean-square radius by eq. (2.84) and eq. (2.85) respectively. To offer flexibility, one can pass either the root-mean-square radius R_{rms} to the parameter `R_rms`, the sphere radius to `R0` or the Fermi parameter c to the identically named parameter when defining the system by constructing a `dish.util.atom.Nucleus` object:

```
from dish import Nucleus

nuc = Nucleus(Z:float, M:float=<optional>,
              R_rms:float=<optional>,
              R0:float=<optional>,
              c:float=<optional>, a:float=<optional>,
              system_charge:float=<optional:Z-1>)
```

The other two parameters are calculated[2]. If no value is passed explicitly for the diffuseness parameter a, the default value $2.3\,\mathrm{fm}/a_0/(4\ln 3)$ is used, which is a good approximation for most stable nuclei [29]. The nuclear charge Z is required while all other parameters can be omitted if the nucleus should be modeled static and point-like as by default the nucleus is modeled infinitely heavy. Keep in mind that the mass of the nucleus is only used for non-relativistic calculations as in the relativistic framework there are no corrections due to finite masses implemented. The parameter `system_charge` describes the system's charge for $r \to \infty$ and is set to $Z - 1$ by default as a hydrogen-like system is calculated. The charge density $\rho(r)$ and the potential $V(r)$ can be evaluated at specific points r by calling

```
rho = nuc.charge_density(r: float|numpy.ndarray|Distancegrid,
                         model:str=<optional>)

V = nuc.potential(r: float|numpy.ndarray|Distancegrid,
                  model:str=<optional>))
```

To the argument `r` a `numpy.ndarray` of floats, a `dish.util.grid.grid.DistanceGrid` or a single `float` can be passed to evaluate the charge density or potential at the given points. The argument `model` is optional and expects a string that specifies, which model for the nucleus should be used. Valid options are (case-insensitive):

1. For a point-like nuclear model: `"point"`, `"point-like"`, `"p"`, `"coulomb"`, `"c"`
2. For a homogeneously charged sphere: `"u"`, `"uniform"`, `"ball"`, `"s"`, `"sphere"`
3. For a Fermi charge distribution: `"f"`, `"fermi"`

The potential models and their charge distributions are implemented in the module `dish.util.potential`. To each model there are corresponding classes for the potential and the charge density distribution which are subclasses of `potential.PotentialModel` and `potential.ChargeDistribution`, respectively.

To use another potential model to describe your systems, you can write a subclass of `dish.util.atom.Nucleus` and implement the `potential` and `charge_density` methods as shown below:

```
from dish.util.atom import Nucleus

class MyCustomNucleus(Nucleus):

    def potential(self, r, model):
        if model.lower() == "my-custom-model-name":
            # my implementation
```

[2]For small values of R_{rms} the Fermi model is not applicable with the default value a. If c can't be calculated from R_0 or R_{rms} an error will be thrown if a Fermi-like model is requested.

```
            return result
        else:
            return super().potential(r, model)

    def charge_density(self, r, model):
        if model.lower() == "my-custom-model-name":
            # my implementation
            return result
        else:
            return super().charge_density(r, model)
```

The implementation of the `charge_density`-method is optional as only the `potential`-method is used in the solving process shown in 5.3. It is anyway recommended to achieve uniformity with the implemented nuclear potentials. In A.6 as a short example an implementation for an *Yukawa* potential is given.

5.2.3. Defining the Grid

As discussed in chapter 3.2 all operations are performed on a finite grid of the form (cf. eqs. (3.1))

$$r[i] = r_0 \left(\mathrm{e}^{t[i]} - 1 \right) , \tag{5.1a}$$

where t is a linear grid

$$t[i] = i \cdot h, \quad i = 0, 1, \ldots, N-1 . \tag{5.1b}$$

To construct an instance, pass the parameters `h`, `r0` and `N` to the constructor:

```
from dish.util.radial.grid import DistanceGrid
grid = DistanceGrid(h:float, r0:float, N:int)
```

Alternatively, you can also pass the limiting value of r to `r_max` instead of `N` in which case `N` will be calculated internally:

```
grid = DistanceGrid(h:float, r0:float, r_grid:float=<value>)
```

The grid points $r[i]$ and $t[i]$ and the points of the derivative are stored in `numpy.ndarrays` and can be accessed using the attributes `r`, `t` and `rp` respectively.

To calculate matrix elements and for the precision to not be limited by the integration method it might be useful to use a `RombergIntegrationGrid`. For details see chapter 3.7. `dish.util.radial.grid.RombergIntegrationGrid` is a class that assures that the number of grid points allows for integration using Romberg's method. An instance can be constructed using

```
from dish.util.radial.grid import RombergIntegrationGrid
grid = RombergIntegrationGrid(r0:float, h:float, N:float)
# alternatively a keyworded parameter k:int can be passed
```

```
# from which N is calculated:
grid = RombergIntegrationGrid(r0:float, h:float, k:int = <value>)
```

This does not allow for passing the maximum distance, but a `RombergIntegrationGrid` can be constructed from a `DistanceGrid` using `RombergIntegrationGrid`'s *classmethod* `construct_similar_grid_from_distance_grid(grid:DistanceGrid)`. To construct it h is lowered so that the N is increased as $N \leftarrow 2^{\lceil \log_2(N-1) \rceil} + 1$.

5.2.4. Defining Electronic States

A relativistic electronic bound state in a hydrogenic system is defined by the four quantum numbers n, l, j and m. The orbital angular projection quantum number m is important only for the spherical part of the wave function. The radial part of the wave function is therefore defined by n, l and j or equally by n and $\kappa := \mp(j + 1/2)$ for $j = l \pm 1/2$. To store these values use an instance of `dish.util.atom.QuantumNumberSet`:

```
from dish.util.atom import QuantumNumberSet

state = QuantumNumberSet(n:int, l:int, j:float)
state = QuantumNumberSet(n:int, kappa:float = <value>)
```

Note that `kappa` is required to be a keyworded argument. All four quantum numbers can be accessed by the corresponding attributes `state.n` etc.
In the following, "state" will refer only to the radial part of a bound state.

Often a state is written not in terms of n, l, j or n, κ but it is given in spectroscopic notation

$$n\langle l\text{-alias}\rangle_j\,, \tag{5.2a}$$

where the alias for l is given by $0 : s, 1 : p, 2 : d, 3 : f, 4 : g, \ldots$ and the brackets mean that the inner part should be replaced. Since for a hydrogenic system only $j = l \pm 1/2$ is possible and therefore also the notation

$$n\langle l\text{-alias}\rangle_{\pm} \tag{5.2b}$$

is common. Some people also use the notation

$$n[l]_j\,, \tag{5.2c}$$

where the value for l is given directly. In `dish` all these versions can be parsed from a string using the function `dish.util.atom.parse_atomic_term_symbol`, also combinations are supported:

```
from dish.util.atom import parse_atomic_term_symbol

state: QuantumNumberSet = parse_atomic_term_symbol(state_repr:str)
```

```
# e.g.
parse_atomic_term_symbol("1s+")
parse_atomic_term_symbol("4d3/2")
parse_atomic_term_symbol("4[2]-")
```

A non-relativistic state is only given by the two quantum numbers n and l. These states can also be represented using the `QuantumNumberSet`-class by only passing these two values. The term symbol is written as

$$n\langle l\text{-alias}\rangle \quad \text{or} \quad n[l]\,, \tag{5.3}$$

and can also be parsed using `parse_atomic_term_symbol`.

5.3. Calculating the States Energy and Wave Function

Having defined the state, the nucleus and the grid, one can solve the DE/SE to find the energy and the wave function of the given state. In `dish`, this is done by calling the function `dish.dirac.solver.solve`/`dish.schrodinger.solver.solve`:

```
from dish.dirac.solver import solve
# or importing it directly from the main namespace:
from dish import solve

result = solve(nucleus: Nucleus,
               state: QuantumNumberSet,
               r_grid: DistanceGrid = <optional:dict(r0=1e-6, h=1e-4)>,
               potential_model: str = <optional:"Fermi">,
               m: float = <optional:1>,
               E_guess: float = <optional:"auto">,
               order_AM: int = <optional:5>,
               order_indir: int = <optional:7>,  # for SE: 'order_insch'
               max_number_of_iterations: int = <optional:20>
              )
```

The objects which were constructed in the last few sections should be passed to the parameters `nucleus`, `state` and `r_grid`.
The model of the nuclear potential can be specified by passing a corresponding alias (cf. chapter 5.2.2) to the parameter `potential_model`. By default, a Fermi-like model is used as specified by `"Fermi"`.
To `m` the mass of the particle for which the equation should be solved can be passed. The default value is 1 (a.u.) which is the mass of an electron. As discussed in chapter 4.1.3, muons are, except for the mass, similar to electrons and therefore muonic wave functions can be retrieved by passing the muons mass $m \approx 207$ (a.u.) to `m`.

The value passed to `E_guess` is used as an initial guess for the states energy. This usually expects a `float` but the default `string` value `"auto"` uses the analytic solution for a Coulomb potential.
The order of the AM procedure (cf. chapter 3.3) can be specified via `order_AM`. The default value of 5 has been proven to be reliable and yields good results as discussed in chapter 4.1.1.
The parameter `order_indir` is unique to the DE solver as the procedure to find the starting values is called `indir` for the DE. The equivalent procedure for the SE is called `insch`, and, therefore, it is called `order_insch` for `dish.schrodinger.solver.solve`. As the name implies, the value passed to these parameters determines the order of these procedures which are discussed in chapter 3.6 and chapter 3.5 respectively. The default value of 7 has been proven to be reliable.
The `solve` function returns a `dish.util.misc.SolvingResult`-object in which all important information about the solving process are stored. In particular the energy and the wave function can be accessed via

```
E = result.energy  # actually E-mc^2 in the relativistic case
wf = result.wave_function
```

5.3.1. Wave Functions

The wave function, calculated using the respective `solve`-function from the previous section, is stored in a `dish.util.radial.wave_function.RadialDiracWaveFunction` instance for relativistic calculations while solutions of the SE are stored using an instance of `dish.util.radial.wave_function.RadialSchrodingerWaveFunction`.
Bound solutions of the DE are of the form

$$\psi_{n\kappa m}(r,\theta,\varphi) = \frac{1}{r}\begin{pmatrix} i f_{n\kappa}(r)\Omega_{\kappa,m}(\theta,\varphi) \\ g_{n\kappa}(r)\Omega_{-\kappa,m}(\theta,\varphi)\end{pmatrix} \quad \text{(cf. eq. (2.52))}. \tag{5.4}$$

The radial parts f and g evaluated on the `DistanceGrid` `grid` can be accessed using the corresponding fields:

```
wf.f     # large component values -> stored in a numpy array
wf.g     # small component values -> stored in a numpy array
```

Additionally, information about the grid and the state to which the wave function belongs are stored:

```
wf.r     # grid points -> stored in a numpy array
wf.grid  # the DistanceGrid instance itself
wf.state # the QuantumNumberSet instance of the state
```

Having found the wave function on a particular grid, the wave function can be interpolated on an arbitrary grid or even at arbitrary distance using cubic spline interpolation:

```
wf.interpolate_at(r: DistanceGrid)        # new RadialDiracWaveFunction
wf.interpolate_values(r: numpy.ndarray)   # values in an numpy array
```

More details on spline interpolation can be found in [37, ch. 3].
To use the wave functions from another tool the values can be exported in a plain text file using the `write_to_file`-method:

```
wf.write_to_file(filename: str)
```

In very close analogy to the relativistic wave functions the solutions of the SE which are of the form

$$\psi_{nlm}(r,\theta,\varphi) = \frac{1}{r} R_{nl} Y_{lm}(\theta,\varphi) \quad \text{(cf. eq. (2.22))} \tag{5.5}$$

are stored in a `RadialSchrodingerWaveFunction`. The radial component R can be accessed as well as its derivative Q:

```
wf.R        # radial function's values -> stored in a numpy array
wf.Q        # dR/dr -> stored in an numpy array
```

The other features are the same as for the relativistic wave functions.

5.3.2. Calculating Matrix Elements

As discussed in chapter 4.2, having the relativistic radial wave functions $|n_1\kappa_1\rangle$, $|n_2\kappa_2\rangle$ of two states, one can calculate radial matrix elements

$$\langle n_1\kappa_1|\hat{o}_r|n_2\kappa_2\rangle = \int_0^\infty \begin{pmatrix} -if_{n_1\kappa_1}(r) & g_{n_1\kappa_1}(r) \end{pmatrix} \hat{o}_r(r) \begin{pmatrix} if_{n_2\kappa_2}(r) \\ g_{n_2\kappa_2}(r) \end{pmatrix} \mathrm{d}r \quad \text{(cf. eq. (4.7))}. \tag{5.6}$$

In this formula the Jacobian determinant r^2 cancels with the factor $1/r$ of the wave functions. Because of this, the radial wave functions are stored without this factor to avoid unnecessary loss of precision due to numerical errors when dividing by numbers very close to zero.
To calculate these matrix elements `dish` provides two interfaces. The low-level interface just performs the evaluation of a radial integral on a `DistanceGrid` or its subclass `RombergIntegrationGrid`:

```
from dish.util.radial.integration import integrate_on_grid

integrate_on_grid(y:numpy.ndarray, grid:DistanceGrid)
```

This offers the most flexibility as `y` can be any array of values at the grid points while taking the integration mathematics off the user. Passing a `RombergIntegrationGrid` to the parameter `grid`, Romberg's method will be used for integration, and if a `DistanceGrid` instance is passed, it will simply fall back to the trapezoidal rule. As discussed in chapter 3.7 a `RombergIntegrationGrid` should be used if possible.
As an example, calculating $\langle n_1\kappa_1|\gamma_5|n_2\kappa_2\rangle$ where $\gamma_5 = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ is the fifth Dirac-matrix for the radial part results in the following code:

```
from dish import solve, integrate_on_grid
# the details of the solving process are spared out
# refer to the previous sections
wf1 = solve(...).wave_function
wf2 = solve(...).wave_function

# you should assure wf1 and wf2 are evaluated on the same grid
integrate_on_grid(-wf1.f*wf2.g + wf1.g * wf2.f, grid=wf1.grid) * 1j
```

The same can be achieved using the high-level interface implemented in the module `dish.util.radial.operator`[3]. It enables a syntax very similar to the Bra-ket notation. You can define operators which extend the base-class `AbstractOperator` and apply them on instances of subclasses of `dish.util.radial.wave_function.RadialWaveFunction`. In the case of scalar or matrix operators, a new wave function is returned with values modified by the effect of the operator. Also, a `BraOperator` can be constructed from a `RadialWaveFunction`-object which performs the integration when it is applied on a `RadialWaveFunction`, and therefore this returns a scalar value.
Applying an operator is done by multiplying the operator from the left to a wave function:

```
from dish.util.radial.operator import BraOperator, AbstractOperator

# this is actually not valid but a subclass should be used
op = AbstractOperator()

op * wf1  # returns a new wave function
BraOperator(wf1) * op * wf2  # returns a scalar value
# e.g. calculating an expectation value becomes
BraOperator(wf1) * op * wf2
```

Operators can be chained and algebraic rules hold. The evaluation is performed from the right to the left:

```
# let op1, op2, op3 be instances of subclasses of AbstractOperator

op1 * op2 * wf1 # == op1 * (op2 * wf1)
op1 * (op2 + op3) * wf1 # == op1 * (op2 * wf1 + op3 * wf1)
```

The discussed syntax can be used for both, relativistic and non-relativistic wave functions. Note, that in the case of non-relativistic case the wave function is scalar and hence no matrix operators can be used. If a scalar operator $\hat{a}_r$ is applied on a relativistic wave function,

[3] The high-level interface is still being tested at the time of release 0.1.0 (along the thesis) and hence not stable. If you notice a bug please create an issue on GitHub describing the problem.
Note that fixes of such may have impacts on details of the implementation. Therefore, refer to the online documentation of this module.

it is interpreted as $\hat{a}_r \cdot \mathbb{1}$.
There are two types of operators that extend the base-class `AbstractOperator`. The first one are operators that are given as actual numbers at each grid point, i.e. the values are stored in an array and when applied on a wave function point-wise multiplied by it. The more interesting kind are the symbolic operators which accept functions that take the wave function as an argument. This allows for very versatile usage like the implementation of the differential operator $\mathrm{d}/\mathrm{d}r$ using numeric differentiation:

```
from dish import DifferentialOperator
DifferentialOperator() * wf
```

Also operators that are dependent of r can be easily constructed using a `RadialOperator` which is a subclass of `SymbolicScalarOperator`. To calculate e.g. the expectation value $\langle r \rangle$ one can write:

```
# let wf_s be a RadialSchrodingerWaveFunction
# and wf_d a RadialDiracWaveFunction
from dish import RadialOperator

BraOperator(wf_s) * RadialOperator(lambda r: r) * wf_s
# the same syntax can be used for the relativistic wave functions
# where actually the scalar operator is multiplied by a unity matrix
BraOperator(wf_d) * RadialOperator(lambda r: r) * wf_d
```

Matrix versions can be constructed by passing scalar operators in a nested list:

```
from dish import SymbolicMatrixOperator

op = SymbolicMatrixOperator([[RadialOperator(lambda r: r), 5],
                             1j, RadialOperator(lambda r: r**2+50)]])
# physically useless but here for demonstration purposes
BraOperator(wf_d) * op * wf_d
```

Instances of `RadialOperator` evaluate the function's argument r on the grid of the wave function. For better readability, it is useful to use anonymous functions using the `lambda`-syntax, like shown in the example above, when constructing `RadialOperator`-instances. Calculating $\langle n_1\kappa_1|\gamma_5|n_2\kappa_2\rangle$ using this syntax, results in

```
# assume wf1, wf2 to be RadialDiracWaveFunctions like above
gamma_5 = SymbolicMatrixOperator([[0,1], [1,0]])
BraOperator(wf1) * gamma_5 * wf2
```

As can be seen from the above examples, these symbolic operators allow for a syntax very similar to handwritten matrix elements. More examples are shown in the appendix in chapter A.7.2.

6. Summary and Outlook

In this thesis, it is shown how one can numerically solve the Schrödinger and Dirac equation (SE and DE) for hydrogen-like systems described by an arbitrary central potential. We focus on giving an introduction to numerics used for atomic physics suitable for students who want to deepen their knowledge in this area. To supply an implementation of the introduced numerical approach for first hands-on calculations, the atomic toolkit `dish` is provided with the thesis.

After covering the analytic solution to these equations when describing the nucleus point-like in the first part, the procedure to find numerical solutions used in this thesis is described: Initial values, found from asymptotic assumptions, are used to integrate the radial SE or DE from the origin outward, and from $a_\infty \gg 1$ inward using an Adams-Moulton multistep method. If the wave function found in this way does not fulfill the physical properties, the energy is adjusted, and the process is repeated. Otherwise, the state's energy and wave function have been found.
Then, the numerical methods used to perform this method are introduced, and their implementation in `dish` is referenced. The conducted comparison of results obtained from `dish` with analytical results and such from the existing atomic codes GRASP and JAC shows that `dish` is capable of providing wave functions and energies of bound states with high precision in a runtime that is competitive with existing codes. Also, the calculation of matrix elements using the wave functions yields high-accuracy results. To perform the computation of wave functions, `dish` provides a simple, user-friendly interface very similar for both, relativistic and non-relativistic description. For the calculation of matrix elements, next to a very versatile low-level interface to calculate radial integrals, a high-level interface is available allowing easy translation of the mathematical representation of operators into code.
As `dish` is implemented as a Python package, it can be integrated effortlessly into your ecosystem and easily extended to the needs of your application. For example, it is easily possible to solve the SE or DE for a custom nuclear potential. To work with `dish`, rich documentation with several examples is available online and supports the introduction given in this thesis. With that, `dish` helps to find a smooth entrance into numerical atomic calculations but is, because of the shown performance, not just a tool for beginners in atomic physics as `dish` is also a suitable tool for carrying out calculations requiring wave functions of hydrogenic systems with high efficiency and precision.

We only gave an introduction to the numerics and demonstrated the usage and performance of `dish`, but an actual application was not shown as it would exceed the limits this thesis. Nonetheless, we already used `dish` to calculate the weak-interaction mixing of the hydrogenic states $2s_{1/2}$ and $2p_{1/2}$ for Ca^{19+}, and with that we verified the results from [47].

Currently, we use the tool to continue the investigation on parity non-conserving effects based on [48].
A possible and fruitful extension for `dish` would be to solve the DE for continuum states, which have a positive energy. Considering the different asymptotic behavior, the same approach used for the bound states can be applied which would offer a straight-forward implementation in `dish`.

A. Appendix

A.1. Bra-ket Notation

In this thesis often the Dirac or Bra-ket notation is used to label a wave function using its quantum numbers

$$\psi_\alpha \equiv |\alpha\rangle \tag{A.1}$$

where α is a tuple of quantum numbers, e.g. the eigenfunctions $\varphi_{j,m}$ of an angular momentum $\hat{\mathbf{J}}$ operator will be written as $|jm\rangle$.
The above object is called a ket and is just another notation for the wave function ψ_α which is an element from the underlying Hilbert space $\mathbb{H}$. The bra-ket notation allows a simple (and unified) syntax for the scalar product $(\cdot,\cdot)$ of two elements a, b of the vector space:

$$(a,b) \equiv \langle a|b\rangle \tag{A.2}$$

Therefore the so-called *bra* $\langle a|$ is introduced which is mathematically a linear operator (from the dual space $\mathbb{H}^*$ of $\mathbb{H}$) and is for an L^2-function ψ defined as

$$\langle\psi|\cdot := \int_{\mathbb{R}^3} \psi^* \cdot \mathrm{d}\mathbf{r}$$

and for vectors $\mathbf{v}$ from the $\mathbb{C}^n$

$$\langle\mathbf{v}|\cdot := \mathbf{v}^\dagger \cdot \ .$$

An advantage of using the bra-ket notation is, one does not have to think about in which Hilbert space the objects are and which scalar product to use until the final evaluation.

A wave function that is a (tensor) product of two functions φ_{α_1} and φ_{α_2} from two different Hilbert subspaces will be written in short as

$$|\alpha_1\rangle \otimes |\alpha_2\rangle \equiv |\alpha_1, \alpha_2\rangle \ .$$

A.2. Auxiliary Calculations

A.2.1. Separation of Variables of the Dirac Equation

Here the calculations which allow for a separation of variables for the time-independent DE for a spherical symmetric potential eq. (2.51) are carried out in more detail. It is based on [10, 18].

First, we will study the influence of the parity operator

$$P : \mathbb{R}^3 \to \mathbb{R}^3 : \mathbf{r} \mapsto -\mathbf{r} \tag{A.3}$$

which is in spherical coordinates equivalent to

$$r \mapsto r, \quad \varphi \mapsto \varphi + \pi, \quad \theta \mapsto \theta\,.$$

One finds for the spherical harmonics

$$PY_{lm}(\theta,\varphi) = (-1)^l Y_{lm}(\theta,\varphi) \tag{A.4}$$

and therefore, that the spherical spinors $\Omega_{\kappa m}$ are also eigenfunctions of P with eigenvalue $(-1)^l$. $\Omega_{\kappa m}$ and $\Omega_{-\kappa m}$ have, because of the definition of κ, the same value of j but differ in l by 1 and therefore have opposite parity. We then note that

$$\left[\sigma\cdot\hat{\mathbf{r}}, \hat{\mathbf{J}}\right], \tag{A.5}$$

where $\hat{\mathbf{r}} := \mathbf{r}/r$, and hence the value of j is unchanged by $\hat{\sigma}\cdot\hat{\mathbf{r}}$. Applying the parity operator yields

$$P\hat{\sigma}\cdot\hat{\mathbf{r}} = -\sigma\cdot\hat{\mathbf{r}}$$

from which follows

$$\sigma\cdot\hat{\mathbf{r}}\,\Omega_{\kappa m}(\theta,\varphi) = a\Omega_{-\kappa m}(\theta,\varphi)\,, \tag{A.6}$$

where $a = -1$ can be easily found by evaluating the above equation for $\theta = 0$.
For the following calculations, we also need the property

$$\sigma\cdot\mathbf{a}\sigma\cdot\mathbf{b} = \mathbf{a}\cdot\mathbf{b} + i\sigma\left[\mathbf{a}\times\mathbf{b}\right] \tag{A.7}$$

for two vectors $\mathbf{a}$ and $\mathbf{b}$. From this follows directly the identity

$$\sigma\cdot\hat{\mathbf{r}}\sigma\cdot\hat{\mathbf{r}} = 1\,. \tag{A.8}$$

We now apply $\sigma\cdot\mathbf{p}f(r)$ on $\Omega_{\kappa m}(\theta,\varphi)$ where $f(r)$ is an arbitrary radial function. This yields

$$\begin{aligned}
\sigma\cdot\mathbf{p}f(r)\Omega_{\kappa m}(\theta,\varphi) &= \sigma\cdot\hat{\mathbf{r}}\sigma\cdot\hat{\mathbf{r}}\sigma\cdot\mathbf{p}f\Omega_{\kappa m}\\
&= -i\sigma\cdot\hat{\mathbf{r}}\left(i\hat{\mathbf{r}}\cdot\mathbf{p} - \frac{\sigma\cdot[\mathbf{r}\times\mathbf{p}]}{r}\right)f\Omega_{\kappa m}\\
&= -i\frac{\sigma\cdot\hat{\mathbf{r}}}{r}\left(i\mathbf{r}\cdot\mathbf{p} - \sigma\cdot\mathbf{L}\right)f\Omega_{\kappa m}\\
&= -i\frac{\sigma\cdot\hat{\mathbf{r}}}{r}\left(\underbrace{\mathbf{r}\cdot\boldsymbol{\nabla}}_{=r\hat{e}_r\cdot\partial_r\hat{e}_r} + 1\underbrace{-1-\sigma\cdot\mathbf{L}}_{=\kappa}\right)f\Omega_{\kappa m}\\
&= -i\frac{\sigma\cdot\hat{\mathbf{r}}}{r}\left(r\partial_r f + (1+\kappa)f\right)\Omega_{\kappa m}\\
&= -i\left(\partial_r f + \frac{\kappa+1}{r}f\right)\sigma\cdot\hat{\mathbf{r}}\,\Omega_{\kappa m}\\
&= i\left(\partial_r f + \frac{\kappa+1}{r}f\right)\Omega_{-\kappa m}\,.
\end{aligned} \tag{A.9}$$

Here $\mathbf{L}$ is the orbital angular momentum and the definition of κ as an eigenvalue was used. From eq. (A.7) also follows

$$\sigma \cdot \mathbf{L}\sigma \cdot \hat{\mathbf{r}} = \underbrace{\mathbf{L} \cdot \hat{\mathbf{r}}}_{=0} + i\sigma \left[\mathbf{L} \times \hat{\mathbf{r}}\right] = -\hat{\mathbf{r}} \cdot \mathbf{L} - i\sigma \left[\hat{\mathbf{r}} \times \mathbf{L}\right] = -\sigma \cdot \hat{\mathbf{r}}\sigma \cdot \mathbf{L}\,, \tag{A.10}$$

which will be used in the following calculation which is very similar to the derivation of eq. (A.9). Applying $\sigma \cdot \mathbf{p}\sigma \cdot \hat{\mathbf{r}} f(r)$ on a spherical spinor yields

$$\begin{aligned}
\sigma \cdot \mathbf{p}\sigma \cdot \hat{\mathbf{r}} f(r)\Omega_{\kappa m}(\theta, \varphi) &= \sigma \cdot \hat{\mathbf{r}}\sigma \cdot \hat{\mathbf{r}}\sigma \cdot \mathbf{p}\sigma \cdot \hat{\mathbf{r}} f\Omega_{\kappa m} \\
&= -i\frac{\sigma \cdot \hat{\mathbf{r}}}{r}\left(\mathbf{r} \cdot \boldsymbol{\nabla} + 1 - 1 - \sigma \cdot \mathbf{L}\right)\sigma \cdot \hat{\mathbf{r}} f\Omega_{\kappa m} \\
&= -i\sigma \cdot \hat{\mathbf{r}}\left[\hat{\mathbf{r}} \cdot \sigma\boldsymbol{\nabla}\left(\hat{\mathbf{r}} f\Omega_{\kappa m}\right) - \sigma \cdot \hat{\mathbf{r}}\left(1 - 1 - \sigma \cdot \mathbf{L}\right) f\Omega_{\kappa m}\right] \\
&= -i\sigma \cdot \hat{\mathbf{r}}\sigma \cdot \hat{\mathbf{r}}\left[\left(\boldsymbol{\nabla} \cdot \mathbf{r}\right) f + \hat{\mathbf{r}}\left(\boldsymbol{\nabla} f\right) - \frac{\kappa + 1}{r} f\right]\Omega_{\kappa m} \\
&= -i\left(\frac{2}{r} f + \partial_r f - \frac{\kappa + 1}{r} f\right)\Omega_{\kappa m} \\
&= -i\left(\partial_r f - \frac{\kappa - 1}{r} f\right)\Omega_{\kappa m}\,,
\end{aligned} \tag{A.11}$$

where the identity

$$\boldsymbol{\nabla} \cdot \hat{\mathbf{r}} = \boldsymbol{\nabla} \cdot \frac{\mathbf{r}}{r} = \frac{3}{r} - \frac{\mathbf{r} \cdot \mathbf{r}}{r} = \frac{2}{r}$$

was used. Writing the time-independent DE $H_D\psi = E\psi$, in matrix form using

$$H_D = \begin{pmatrix} mc^2 + V & c\sigma \cdot \mathbf{p} \\ c\sigma \cdot \mathbf{p} & -mc^2 + V \end{pmatrix} \tag{A.12}$$

and for the wave functions the *ansatz*

$$\psi = \frac{1}{r}\begin{pmatrix} if_\kappa \Omega_{\kappa m} \\ g_\kappa \Omega_{-\kappa m} \end{pmatrix} = \frac{1}{r}\begin{pmatrix} if_\kappa \Omega_{\kappa m} \\ -g_\kappa\left(\sigma \cdot \hat{\mathbf{r}}\right)\Omega_{\kappa m} \end{pmatrix} \tag{A.13}$$

one obtains

$$
\begin{aligned}
\text{I.}\quad 0 &= \left(mc^2+V-E\right)\frac{1}{r}if_\kappa\Omega_{\kappa m}+c\boldsymbol{\sigma}\cdot\mathbf{p}\frac{1}{r}\left(-g_\kappa\left(\boldsymbol{\sigma}\cdot\hat{\mathbf{r}}\right)\Omega_{\kappa m}\right)\\
&= \left(mc^2+V-E\right)\frac{1}{r}if_\kappa\Omega_{\kappa m}+\frac{c}{i}\boldsymbol{\sigma}\cdot\frac{\mathbf{r}}{r^3}\left(-g_\kappa\left(\boldsymbol{\sigma}\cdot\hat{\mathbf{r}}\right)\Omega_{\kappa m}\right)+\frac{c}{r}\boldsymbol{\sigma}\cdot\mathbf{p}\left[\left(\boldsymbol{\sigma}\cdot\hat{\mathbf{r}}\right)\cdot\left(-g_\kappa\right)\Omega_{\kappa m}\right]\\
&= \left(mc^2+V-E\right)\frac{1}{r}if_\kappa\Omega_{\kappa m}+\frac{ic}{r^2}\left(\boldsymbol{\sigma}\cdot\hat{\mathbf{r}}\right)\left(\boldsymbol{\sigma}\cdot\hat{\mathbf{r}}\right)g_\kappa\Omega_{\kappa m}+\frac{ic}{r}\left(\partial_r g_\kappa-\frac{\kappa-1}{r}g_\kappa\right)\Omega_{\kappa m}\\
&= \frac{i}{r}\left[\left(mc^2+V-E\right)f_\kappa\Omega_{\kappa m}+\frac{c}{r}\left(1-\kappa-1\right)g_\kappa\Omega_{\kappa m}+\frac{\mathrm{d}}{\mathrm{d}r}g_\kappa\Omega_{\kappa m}\right]\quad\text{and}\\
\text{II.}\quad 0 &= c\boldsymbol{\sigma}\cdot\mathbf{p}\frac{1}{r}if_\kappa\Omega_{\kappa m}+\left(-mc^2+V-E\right)\frac{1}{r}\left(g_\kappa\Omega_{-\kappa m}\right)\\
&= c\left(\boldsymbol{\sigma}\cdot\frac{1}{i}\left(\boldsymbol{\nabla}\frac{1}{r}\right)if_\kappa\Omega_{\kappa m}+\frac{1}{r}\boldsymbol{\sigma}\cdot\mathbf{p}if_\kappa\Omega_{\kappa m}\right)+\frac{1}{r}\left(V-mc^2-E\right)g_\kappa\Omega_{-\kappa m}\\
&= c\left(\boldsymbol{\sigma}\cdot\hat{\mathbf{r}}\left(-\frac{1}{r^2}\right)f_\kappa\Omega_{\kappa m}+\frac{i}{r}i\left(\partial_r f_\kappa+\frac{\kappa+1}{r}f_\kappa\right)\Omega_{-\kappa m}\right)+\frac{1}{r}\left(V-mc^2-E\right)g_\kappa\Omega_{-\kappa m}\\
&= \frac{c}{r}\left(\frac{1}{r}\left(1-\kappa-1\right)f_\kappa-\frac{\mathrm{d}}{\mathrm{d}r}f_\kappa\right)\Omega_{-\kappa m}+\frac{1}{r}\left(V-mc^2-E\right)g_\kappa\Omega_{-\kappa m}\,.
\end{aligned}
$$

From these two results we find

$$\left(V+mc^2\right)f_\kappa+c\left(\frac{\mathrm{d}}{\mathrm{d}r}-\frac{\kappa}{r}\right)g_\kappa=Ef_\kappa\quad\text{and}\tag{A.14a}$$

$$-c\left(\frac{\mathrm{d}}{\mathrm{d}r}+\frac{\kappa}{r}\right)f_\kappa+\left(V-mc^2\right)g_\kappa=Eg_\kappa\,,\tag{A.14b}$$

which are the equations defining the radial components f_κ and g_κ eqs. (2.53).

A.2.2. Root-Mean-Squared Radius of a Fermi Charge Density Distribution

Here is shown how the root-mean-squared radius $R_{\mathrm{rms},F}$ of a charge density $\rho_F(r)$ which is given by a Fermi distribution eq. (2.82) is calculated in `dish`. It is based on the implementation in JAC. For this we will make use of the fact, that the Fermi-Dirac integral

$$F_n(x)=\frac{1}{\Gamma(n+1)}\int_0^\infty\frac{x^n}{1+\exp(t-x)}\mathrm{d}t\tag{A.15}$$

can be expressed in terms of the polylogarithm

$$\mathrm{Li}_s(z)=\sum_{k=1}^{\infty}\frac{z^k}{k^s},\quad|z|<1\tag{A.16}$$

as

$$F_n(x)=-\mathrm{Li}_{n+1}(-\mathrm{e}^x)\,.\tag{A.17}$$

For positive x the argument $-\mathrm{e}^x<-1$ and the polylogarithm is continued analytically.

Using these identities, we find:

$$\begin{aligned} R^2_{\text{rms},F} &= \frac{\int \rho_F(r) r^2 \mathrm{d}\mathbf{r}}{\int \rho_F(r) \mathrm{d}\mathbf{r}} \\ &= \frac{\int_0^\infty \frac{\rho_0 r^4}{1+\exp\left(\frac{r-c}{a}\right)} \mathrm{d}r}{\int_0^\infty \frac{\rho_0 r^2}{1+\exp\left(\frac{r-c}{a}\right)} \mathrm{d}r}, \quad u := \frac{r}{a}, \mu := \frac{c}{a} \\ &= a^2 \frac{\int_0^\infty \frac{u^4}{1+\exp(u-\mu)} \mathrm{d}u}{\int_0^\infty \frac{u^2}{1+\exp(u-\mu)} \mathrm{d}u} \\ &\overset{(A.15)}{=} a^2 \cdot \frac{\Gamma(2+1)}{\Gamma(4+1)} \frac{F_4(u)}{F_2(u)} \\ &\overset{(A.17)}{=} 12a^2 \frac{\mathrm{Li}_5\left(-\exp\left(\frac{c}{a}\right)\right)}{\mathrm{Li}_3\left(-\exp\left(\frac{c}{a}\right)\right)} \end{aligned} \tag{A.18}$$

With this we found a way to write the $R_{\text{rms},F}$ in terms of polylogarithms which are implemented in numeric packages like `gympy2` and can be used to evaluate $R_{\text{rms},F}$ with high precision.

We can also use this function which determines $R_{\text{rms},F}$ in dependence of c and a to calculate the parameter c from a given value for $R_{\text{rms},F}$ by using the simple root finding algorithm 1.

Algorithm 1: Root-finding algorithm to calculate the Fermi-parameter c

input : root-mean-squared radius R_{rms}, Fermi-parameter a, precision ε, maximum number of iterations MAX_IT
output: approximated Fermi-parameter c

$c \leftarrow \frac{5}{3}R^2_{\text{rms}} - \frac{7}{3}\pi^2 a^2$ `// Initial guess from eq. (2.85) for` c^2
if $c < 0$ **then**
    `// the Fermi model can't be applied for small values of` R_{rms} `if` a `is too large`
    return NaN
else
    $c \leftarrow \sqrt{c}$
    for $i \leftarrow 1$ **to** MAX_IT **do**
        diff $\leftarrow$ `Rrms(`c, a`)` $- R_{\text{rms}}$
        if $|c| < \varepsilon$ **then break**
        $c \leftarrow c -$ diff
    end
    return c
end

A.3. Calculating Wave Functions

A.3.1. Using GRASP2018

In GRASP the parts of the calculations are spread over several scripts. There is a manual [49] where some examples are discussed which scripts to use for certain calculations. Nevertheless, getting started with GRASP has a high entrance barrier since it is not easy to find out what exactly the scripts are doing since the amount of scripts and overall complexity has grown over the time. Also, the program is optimized for execution speed and not for user experience since it is meant for significantly more complex and computationally expensive calculations than finding hydrogenic wave functions.
The following snippet shows the shell in- and output to calculate the energy and wave function of the $1s_{1/2}$ state of Hydrogen. The lines where the scripts and shell commands are executed start with a "$".

```
$ rnucleus
 Enter the atomic number:
1
 Enter the mass number (0 if the nucleus is to be modelled as a point
 ↪  source:
1
 The default root mean squared radius is   0.87830001115798950      fm;
 ↪  (Angeli)
   the default nuclear skin thickness is    2.2999999999999998      fm;
 Revise these values?
y
 Enter the root mean squared radius of the nucleus (in fm):
0.8783
 Enter the skin thickness of the nucleus (in fm):
1
 Enter the mass of the neutral atom (in amu) (0 if the nucleus is to be
 ↪  static):
0
 Enter the nuclear spin quantum number (I) (in units of h / 2 pi):
0.5
 Enter the nuclear dipole moment (in nuclear magnetons):
2.792847
 Enter the nuclear quadrupole moment (in barns):
0.00286
Note: The following floating-point exceptions are signalling:
 ↪  IEEE_INVALID_FLAG IEEE_OVERFLOW_FLAG
```

```
$ rcsfgenerate

 RCSFGENERATE
 This program generates a list of CSFs

 Configurations should be entered in spectroscopic notation
 with occupation numbers and indications if orbitals are
 closed (c), inactive (i), active (*) or has a minimal
 occupation e.g. 1s(2,1)2s(2,*)
 Outputfiles: rcsf.out, rcsfgenerate.log

 Default, reverse, symmetry or user specified ordering? (*/r/s/u)
*

 Select core
        0: No core
        1: He (       1s(2)                  =  2 electrons)
        2: Ne ([He] + 2s(2)2p(6)             = 10 electrons)
        3: Ar ([Ne] + 3s(2)3p(6)             = 18 electrons)
        4: Kr ([Ar] + 3d(10)4s(2)4p(6)       = 36 electrons)
        5: Xe ([Kr] + 4d(10)5s(2)5p(6)       = 54 electrons)
        6: Rn ([Xe] + 4f(14)5d(10)6s(2)6p(6) = 86 electrons)
0

 Enter list of (maximum 100) configurations. End list with a blank line
 ↪  or an asterisk (*)

 Give configuration           1
1s(1,i)
 Give configuration           2

 Give set of active orbitals, as defined by the highest principal
 ↪  quantum number
 per l-symmetry, in a comma delimited list in s,p,d etc order, e.g.
 ↪  5s,4p,3d
1s
 Resulting 2*J-number? lower, higher (J=1 -> 2*J=2 etc.)
1,1
 Number of excitations (if negative number e.g. -2, correlation
 orbitals will always be doubly occupied)
0
```

```
 Generate more lists ? (y/n)
n
  Excitationdata file opened

 THE LINES BELOW ARE DEBUG OUTPUT FROM THE
 CSF GENERATOR:   PLEASE IGNORE !!!

     *  : new list
     e  : expand existing list
     q  : quit
 Default, reverse, symmetry or user specified ordering? (*/r/s/u)
 Highest principal quantum number, n? (1..15)
 Highest orbital angular momentum, l? (s..s)
 Are all these nl-subshells active? (n/*)
 Highest n-number in reference configuration? (1..1)
 Highest n-number in reference configuration? (1..1)
 Predefine open, closed or no core? (o/c/*)
 Number of electrons in 1s? (0..2)
 Inactive or active? (i/*)
 Resulting 2*J-number? lower, higher (J=1 -> 2*J=2 etc.)
 Number of excitations = ? (0..0)
 One configuration state has been generated.
 One configuration state in the final list.
 Generate another list? (y/*)

 Group CSFs into symmetry blocks

 1 blocks were created

       block  J/P            NCSF
           1  1/2+              1
$ cp rcsfgenerate.log 1s+.exc
$ cp rcsf.out rcsf.inp
$ rangular

 RANGULAR
 This program performs angular integration
 Input file:  rcsf.inp
 Outputfiles: mcp.30, mcp.31, ....
 rangular.log
```

```
 Full interaction?  (y/n)
y
 Block            1 ,  ncf =            1
 Loading CSF file ... Header only
 There are/is            1  relativistic subshells;

 Block            1 ,  ncf =            1
 Loading CSF File for block            1
 There are            1  relativistic CSFs... load complete;
 Analysing sparse matrix array definition file ...          30
  ... complete; density of non-zero elements of H(DC):           1 /
↪  1
 Sorting 0 T coefficients ...          31
 Sorting 0 V(k=0) coefficients ...          32
 Sorting 0 V(k=1) coefficients ...          33

 Wall time:
        1 seconds

 Finish Date and Time:
   Date (Yr/Mon/Day): 2024/05/01
   Time (Hr/Min/Sec): 16/51/38.612
   Zone: +0200

 RANGULAR: Execution complete.
$ rwfnestimate
 RWFNESTIMATE
 This program estimates radial wave functions
 for orbitals
 Input files: isodata, rcsf.inp, optional rwfn file
 Output file: rwfn.inp
 Default settings ?
y
 Loading CSF file ... Header only
 There are/is            1  relativistic subshells;
 The following subshell radial wavefunctions remain to be estimated:
 1s

 Read subshell radial wavefunctions. Choose one below
     1 -- GRASP92 File
     2 -- Thomas-Fermi
```

```
     3 -- Screened Hydrogenic
     4 -- Screened Hydrogenic [custom Z]
3
 Enter the list of relativistic subshells:
*
 Orbital Z_eff for hydrogenic orbitals
 1s       1.00

 All required subshell radial wavefunctions  have been estimated:
Shell      e           p0        gamma        <r>      MTP  SRC

  1s   0.5000D+00  0.2000D+01  0.1000D+01  0.1500D+01  327  Hyd
 RWFNESTIMATE: Execution complete.
Note: The following floating-point exceptions are signalling:
 ↪  IEEE_UNDERFLOW_FLAG IEEE_DENORMAL
$ rmcdhf

 RMCDHF
 This program determines the radial orbitals
 and the expansion coefficients of the CSFs
 in a self-onsistent field proceedure
 Input file:  isodata, rcsf.inp, rwfn.inp, mcp.30, ...
 Outputfiles: rwfn.out, rmix.out, rmcdhf.sum, rmcdhf.log

Default settings?  (y/n) y
 Loading CSF file ... Header only
 There are/is            1  relativistic subshells;
 Loading CSF File for ALL blocks
 There are            1  relativistic CSFs... load complete;
 Loading Radial WaveFunction File ...
 There are            1  blocks  (block   J/Parity   NCF):
  1  1/2+       1

 Enter ASF serial numbers for each block
 Block            1    ncf =            1  id =  1/2+
1
 Radial functions
 1s
 Enter orbitals to be varied (Updating order)
*
 Which of these are spectroscopic orbitals?
```

```
*
 Enter the maximum number of SCF cycles:
100

 Average energy =  -5.0000665564D-01 Hartrees

Optimise on the following level(s):

Level  1    Energy = -5.000066556402D-01    Weight =  1.00000D+00

Weights of major contributors to ASF:

Block Level J Parity      CSF contributions

  1     1   1/2 +       1.0000
                              1

Generalised occupation numbers:

   1.0000D+00

 Iteration number   1
 --------------------

Lagrange multipliers are not required
                                          Self-            Damping
Subshell    Energy    Method   P0    consistency  Norm-1  factor  JP MTP
↪  INV NNP

  1s    5.0000666D-01  2  2.000D+00  7.99D-08  0.00D+00 0.000   278
↪  328  0  0

 Average energy =  -5.0000665564D-01 Hartrees

Optimise on the following level(s):

Level  1    Energy = -5.000066556402D-01    Weight =  1.00000D+00

Weights of major contributors to ASF:

Block Level J Parity      CSF contributions
```

```
  1     1   1/2 +       1.0000
                            1

Generalised occupation numbers:

   1.0000D+00

 Iteration number   2
 --------------------
                                        Self-            Damping
Subshell    Energy    Method   P0    consistency  Norm-1  factor  JP MTP
↪  INV NNP

  1s    5.0000666D-01  2  2.000D+00  4.65D-10 -1.11D-16 0.000   278
↪  328  0  0

 Average energy =  -5.0000665564D-01 Hartrees

Optimise on the following level(s):

Level  1    Energy = -5.000066556402D-01    Weight =  1.00000D+00

Weights of major contributors to ASF:

Block Level J Parity      CSF contributions

  1     1   1/2 +       1.0000
                            1

Generalised occupation numbers:

   1.0000D+00

 Wall time:
       14 seconds

 Finish Date and Time:
   Date (Yr/Mon/Day): 2024/05/01
   Time (Hr/Min/Sec): 16/52/11.301
   Zone: +0200
```

```
 RMCDHF: Execution complete.
Note: The following floating-point exceptions are signalling:
 ↪  IEEE_UNDERFLOW_FLAG IEEE_DENORMAL
$ rsave 1s+
Created 1s+.w, 1s+.c, 1s+.m, 1s+.sum 1s+.alog and 1s+.log
$ rwfntotxt

  RWFNTOTXT - Jon Grumer, Uppsala University, 2022
  Converts binary GRASP wavefunctions to ASCII file format

     Input: rwfn.inp      Output: rwfn.plot

     Note: rwfnpyplot can be used to conveniently investigate orbitals
     from the .plot file.
           E.g. >> rwfnpyplot rwfn.plot P 2p- 0.50 y
           plots the large P(2p-) component to screen up to r = 0.50 au.

                              max       max
    i   nl    E[a.u]       grid-n    r[a.u]    a0
    1   1s    5.000067E-01    327     23.99    1.999999E+00

  Done! Radial wavefunctions printed to rwfn.plot.
```

Since this is a huge effort to extract the wave functions it was automated using a shell script:

```
#!/bin/bash
source /path/to/GRASP/enable_environment

export state="1s"
export parity="+"
export doublej=1

rnucleus < rnucleus_input
envsubst < rcsfgenerate_input | rcsfgenerate
cp rcsfgenerate.log "$state$parity.exc"
cp rcsf.out rcsf.inp
rangular < rangular_input
rwfnestimate < rwfnestimate_input
envsubst < rmcdhf_input | rmcdhf
rsave $state$parity
```

```
rwfntotxt
cp rwfn.plot "$state$parity.plot"
cp "$state$parity.plot" "/path/where/to/store/the/data/$state$parity.plot"
cp "$state$parity.sum" "/path/where/to/store/the/data/$state$parity.sum"
```

From the script it can be seen way easier what scripts are used: First the nucleus is defined using `rnucleus`[1]. Then `rcsfgenerate` generates the list of atomic state functions required for the calculation. In this case this is just a single one. Then the program `rangular` sets up the Hamiltonian with the requested interaction. Because we are only interested in one-electron atoms this is only the single particle Hamiltonian. `rwfnestimate` generates the radial wave function using a screened hydrogenic wave function. If there were more than just a single state, the combined orbital would be estimated. Then the radial part and mixing coefficients of the configuration state function are determined using a relativistic self-consistent-field procedure by the program `rmcdhf`. Finally, the output is saved using `rsave` and the values of the wave function are extracted from a binary format and stored in a plain text file using `rwfntotxt`. The energy can be extracted from the *1s+.sum* file.
In the *xxx_input* files the input for the programs is stored and using some environment variables the current state is input. To generate the wave functions for comparing them with the `dish` results, running this script and reading in the wave functions and energies from the output files has also been automated using Python.

[1] In the current version there is a bug where the script fails to calculate the Fermi charge density distribution parameters from the defaults values for small nuclei like hydrogen. No appropriate error/warning is thrown in this case but other scripts fail and produce `NaN`-values. See this issue for more details.

A.3.2. Using JAC

In JAC a computation can be performed in a single script or even using Julia's REPL and without the need of files to transfer data between different parts of the calculation. Here a simple script is given to calculate wave function of the $1s_{1/2}$ state of hydrogen where the nucleus is modeled by a uniformly charged sphere. Afterward, both components of the wave function are plotted.

```
using JAC
using Plots

Z = 1
M = Inf
radius = .8783 # radius must ge given in fm
nuclear_model = "uniform"
nuc = Nuclear.Model(Z, nuclear_model, M, radius, AngularJ64(1//2), 2.792847, 0)
Defaults.setDefaults("unit: energy", "Hartree")
grid_ = Radial.Grid(false)  # use an exponential grid
setDefaults("standard grid", grid_)

nuc_orb = HydrogenicIon.radialOrbital(SubShell("1s_1/2"), nuc, grid_)
plot("radial orbitals: both", Orbital[nuc_orb], grid_; N=350)
```

Similar to `dish` where the resulting wave function is stored in a `RadialWaveFunction` instance in JAC it is stored in an `Orbital` instance along with information about the grid. Using one of Julia's core features *multiple dispatch* JAC offers new methods of the `Plots.plot`-function to easily plot the wave functions.
Because the resulting wave functions are highly dependent on the parameters of the B-Splines which are defined by the `Radial.Grid` one might need to try different parameters for the grid to get good results. The grid can be adjusted by tuning the following parameters:

```
nth = orderL = 7
orderS = 9
orderGL = 7
meshType = Radial.MeshGL()
NoPoints = 392 - rem(NoPoints,orderGL)
grid_ = Radial.Grid(2e-6, 5e-2, 0., NoPoints, Float64[], Float64[], 0,0,
          orderL, orderS, 0,0,orderGL, meshType,
          Float64[], Float64[], Float64[], Float64[])
grid_ = Radial.determineGrid(grid_)
```

A.4. Additional Results

Table A.1.: Scaled radial moments $\langle (2Zr)^s \rangle$ of the non-relativistic states $1s, 3p$ and $10d$ of point-like hydrogen on a `RombergIntegrationGrid` similar to a `DistanceGrid` defined by `r0=1e-6`, `h=1e-3` and `r_max=250`.

state	s	$\langle r^s \rangle_{\text{ref}}$	$\langle r^s \rangle_{\text{dish}}$	$\epsilon_{\text{rel}}(\langle r^s \rangle)$
1s	2	1.200×10^{1}	1.200×10^{1}	6.734×10^{-10}
	1	3.000	3.000	3.673×10^{-10}
	-1	5.000×10^{-1}	5.000×10^{-1}	5.509×10^{-10}
3p	2	7.200×10^{2}	7.200×10^{2}	1.263×10^{-15}
	1	2.500×10^{1}	2.500×10^{1}	5.684×10^{-16}
	-1	5.556×10^{-2}	5.556×10^{-2}	4.996×10^{-16}
10d	2	9.660×10^{4}	9.641×10^{4}	1.954×10^{-3}
	1	2.940×10^{2}	2.938×10^{2}	8.344×10^{-4}
	-1	5.000×10^{-3}	5.003×10^{-3}	6.701×10^{-4}

Table A.2.: Scaled radial moments $\langle (2Zr)^s \rangle$ of the non-relativistic states $1s, 3p$ and $10d$ of point-like hydrogenic uranium on a `RombergIntegrationGrid` similar to a `DistanceGrid` defined by `r0=1e-6`, `h=1e-3` and `r_max=5`.

state	s	$\langle r^s \rangle_{\text{ref}}$	$\langle r^s \rangle_{\text{dish}}$	$\epsilon_{\text{rel}}(\langle r^s \rangle)$
1s	2	1.200×10^{1}	1.200×10^{1}	2.491×10^{-4}
	1	3.000	3.000	1.359×10^{-4}
	-1	5.000×10^{-1}	4.999×10^{-1}	2.031×10^{-4}
3p	2	7.200×10^{2}	7.200×10^{2}	4.011×10^{-10}
	1	2.500×10^{1}	2.500×10^{1}	2.139×10^{-10}
	-1	5.556×10^{-2}	5.556×10^{-2}	4.449×10^{-10}
10d	2	9.660×10^{4}	9.659×10^{4}	1.506×10^{-4}
	1	2.940×10^{2}	2.940×10^{2}	5.983×10^{-5}
	-1	5.000×10^{-3}	5.000×10^{-3}	4.145×10^{-5}

A.5. Installing Pure-Python dish

1. Make sure to have the following prerequirements installed:
   a) A working Python solution based on CPython 3.7 or above but below 3.12[2].
   b) The Python packages `pip` and `build`.

   Then clone the repository from GitHub using

   ```
   git clone https://github.com/suppetia/qm-dish.git
   # or via ssh
   git clone git@github.com:suppetia/qm-dish.git
   ```

   or download the source code.

2. Change into the repositories directory and run from a shell

   ```
   python -m build --outdir dist
   ```

   The package will be built in the *dist* subdirectory.

3. To install the package and its dependencies run

   ```
   python -m pip install qm-dish --find-links dist/
   ```

   from the same directory.
   Not that the package is named `qm-dish` to distinguish it from a package available on pypi named `dish`.

A.6. Implementation of a Custom Nuclear Potential

As shown in chapter 5.2.2, a custom nuclear potential model can be used by implementing a subclass of `dish.util.atom.Nucleus`. Here is a short example given on how to do this for a *Yukawa* (also called *screened Coulomb*) potential

$$V_{\text{Yukawa}}(r) = -g^2 \frac{\mathrm{e}^{-mr}}{r}\,, \tag{A.19}$$

where g is a scaling constant for the magnitude of the potential and m is a scaling constant for the range [50].

```
1 from dish.util.atom import Nucleus
2 import numpy as np
3
4 class YukawaNucleus(Nucleus):
5
6     def __init__(self, Z, g, m):
```

[2] Currently `gympy2` does not support Python 3.12.

```
        self.g = g
        self.m = m

        # note that the following line is required
        # as this is a subclass of Nucleus
        # it passes the nuclear charge Z which is required for asymptotics
        super().__init__(Z)

    def potential(self, r, model):
        if model.lower() in ["yukawa", "y"]:
            return -self.g**2 * np.exp(-self.m * r) / r
        else:
            # this case can be omitted
            # it enables correct error handling and calling the default potentials
            return super().potential(r, model)
```

Construct an instance of your custom nucleus and pass it to `dish.dirac.solver.solve`/ `dish.schrodinger.solver.solve`, and you retrieve the wave functions for the potential. Note, that you might need to try other grid parameters to retrieve good results.

A.7. Computation Examples

A.7.1. Example 1: Plotting Radial Wave Functions

In this short example it is shown how to compute radial wave functions and plot them using the Python plotting library `matplotlib`. First the wave functions of the $2p_{1/2}$ and $2p_{3/2}$ states of hydrogenic calcium are calculated. The system is modeled by a Fermi-like nucleus. Then the large components and the small components of both states are plotted in a figure with two axes stacked above each other. In the upper one both large components are plotted and in the lower one the small components. The resulting plot is shown in fig. A.1.

```
from dish import (
    Nucleus,
    DistanceGrid,
    parse_atomic_term_symbol,
    convert_units,
    solve
)
import matplotlib.pyplot as plt
import scienceplots  # scientific styles of the plots

```

```
# define the hydrogenic system
Ca = Nucleus(Z=20,
             c=convert_units("m", "a_0", 3.629e-15),
             a=convert_units("m", "a_0", 0.552e-15)
             )

# define the grid to work on
r_grid = DistanceGrid(r0=1e-6, h=1e-3, r_max=2)

# initialize the plotting style and set up the figure
plt.style.use(["science", "pgf", "high-vis"])
fig, ax = plt.subplots(nrows=2, sharex=True, figsize=(6, 4))

# iterate over the states and solve the DE
results = {}
wf = {}
states = ["2p-", "2p+"]
for s in states:
    # store the solving results in the dictionary 'results'
    results[s] = solve(nucleus=Ca,
                       state=parse_atomic_term_symbol(s),
                       r_grid=r_grid,
                       potential_model="Fermi")
    # store the wave functions in the dictionary 'wf'
    wf[s] = results[s].wave_function

    # plot the radial components
    ax[0].plot(wf[s].r, wf[s].f, label=f"${s}$")
    ax[1].plot(wf[s].r, wf[s].g, label=f"${s}$")

# modify the properties of the plot
ax[1].set_xlabel("$r$ (in a.u.)")
ax[0].set_ylabel("$f(r)$")
ax[1].set_ylabel("$g(r)$")
ax[0].legend()

# save the resulting plot to a file
fig.savefig(f"wavefunctions-example.eps",
                dpi=300, bbox_inches="tight")
```

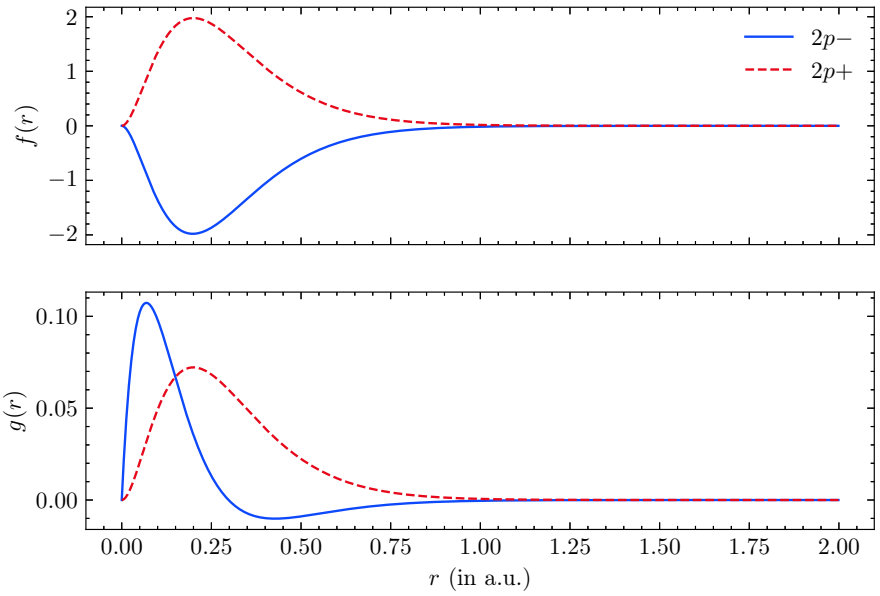

Figure A.1.: The resulting plot of Example 1.

A.7.2. Example 2: More on Operators

In the second example it is shown that also more complex operators can be implemented easily using the high-level operator interface. We will calculate the energy expectation value by implementing the radial Dirac-Hamiltonian. From eqs. (2.53) we find as the matrix representation of the Hamiltonian

$$H_D = \begin{pmatrix} V + c^2 & c\left(\frac{\mathrm{d}}{\mathrm{d}r} - \frac{\kappa}{r}\right) \\ -c\left(\frac{\mathrm{d}}{\mathrm{d}r} + \frac{\kappa}{r}\right) & V - c^2 \end{pmatrix} . \tag{A.20}$$

Calculating the expectation value $E = \langle H_D | a | H_D \rangle\, a$ using `dish` can be done in the following way:

```
from dish import (
    Nucleus,
    DistanceGrid,
    RombergIntegrationGrid,
    parse_atomic_term_symbol,
    convert_units,
    solve
)
from dish.util.radial.operator import (
    BraOperator,
    SymbolicMatrixOperator,
    DifferentialOperator
```

```
)
from dish.util.radial.operator import RadialOperator as RO
from dish.util.atomic_units import c

import numpy as np

# define the hydrogenic system
nuc = Nucleus(Z=1,
              c=convert_units("m", "a_0", .69975e-15),
              a=convert_units("m", "a_0", 1e-15)/(4*np.log(3))
              )

r_grid = DistanceGrid(r0=1e-6, h=1e-3, r_max=250)
r_grid = RombergIntegrationGrid.construct_similar_grid_from_distance_grid(r_grid)

# calculate the wave functions
state_a = parse_atomic_term_symbol("1s1/2")
r_a = solve(nucleus=nuc, state=state_a, r_grid=r_grid,
            potential_model="Fermi")
a = r_a.wave_function

# implement H_D
H_D = SymbolicMatrixOperator([
     [RO(lambda r: nuc.potential(r, "f") + c**2),
      RO(c) * DifferentialOperator() - RO(lambda r: c*state_a.kappa/r)
      ],
     [RO(-c) * DifferentialOperator() + RO(lambda r: c*state_a.kappa/r),
      RO(lambda r: nuc.potential(r, "f") - c**2)]
    ])

# calculate the energy expectation value <a|H_D|a>
E = BraOperator(a) * H_D * a
```

Bibliography

[1] I.P. Grant et al. "An atomic multiconfigurational Dirac-Fock package". In: *Computer Physics Communications* 21.2 (1980), pp. 207–231. ISSN: 0010-4655. DOI: https://doi.org/10.1016/0010-4655(80)90041-7. URL: https://www.sciencedirect.com/science/article/pii/0010465580900417.

[2] F.A. Parpia, C.Froese Fischer, and I.P. Grant. "GRASP92: A package for large-scale relativistic atomic structure calculations". In: *Computer Physics Communications* 94.2 (1996), pp. 249–271. ISSN: 0010-4655. DOI: https://doi.org/10.1016/0010-4655(95)00136-0. URL: https://www.sciencedirect.com/science/article/pii/0010465595001360.

[3] P. Jönsson et al. "New version: Grasp2K relativistic atomic structure package". In: *Computer Physics Communications* 184.9 (2013), pp. 2197–2203. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2013.02.016. URL: https://www.sciencedirect.com/science/article/pii/S0010465513000738.

[4] C. Froese Fischer et al. "GRASP2018—A Fortran 95 version of the General Relativistic Atomic Structure Package". In: *Computer Physics Communications* 237 (2019), pp. 184–187. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2018.10.032. URL: https://www.sciencedirect.com/science/article/pii/S0010465518303928.

[5] Charlotte Froese Fischer. "The MCHF atomic-structure package". In: *Computer Physics Communications* 128.3 (2000), pp. 635–636. ISSN: 0010-4655. DOI: https://doi.org/10.1016/S0010-4655(00)00009-6. URL: https://www.sciencedirect.com/science/article/pii/S0010465500000096.

[6] Charlotte Froese Fischer et al. "An MCHF atomic-structure package for large-scale calculations". In: *Computer Physics Communications* 176.8 (2007), pp. 559–579. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2007.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0010465507000446.

[7] Ming Feng Gu. "The flexible atomic code". In: *Canadian Journal of Physics* 86.5 (2008), pp. 675–689. DOI: 10.1139/p07-197.

[8] Stephan Fritzsche. "A fresh computational approach to atomic structures, processes and cascades". In: *Computer Physics Communications* 240 (2019), pp. 1–14. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2019.01.012. URL: https://www.sciencedirect.com/science/article/pii/S0010465519300189.

[9] I. P. Grant, ed. *Relativistic Quantum Theory of Atoms and Molecules*. 1st ed. Springer New York, 2007. DOI: 10.1007/978-0-387-35069-1.

[10] Walter R. Johnson. *Atomic Structure Theory*. 1st ed. Springer Berlin, Heidelberg, 2007. DOI: 10.1007/978-3-540-68013-0.

[11] Paul A. Tipler and Gene Mosca. *Physik*. Ed. by Jenny Wagner. 7th ed. Springer Berlin, Heidelberg, 2015. ISBN: 978-3-642-54165-0. DOI: 10.1007/978-3-642-54166-7.

[12] M. G. Kozlov et al. "Highly charged ions: Optical clocks and applications in fundamental physics". In: *Rev. Mod. Phys.* 90 (4 Dec. 2018), p. 045005. DOI: 10.1103/RevModPhys.90.045005. URL: https://link.aps.org/doi/10.1103/RevModPhys.90.045005.

[13] P. Beiersdorfer et al. "The magnetic trapping mode of an electron beam ion trap: New opportunities for highly charged ion research". In: *Review of Scientific Instruments* 67.11 (Nov. 1996), pp. 3818–3826. ISSN: 0034-6748. DOI: 10.1063/1.1147276. eprint: https://pubs.aip.org/aip/rsi/article-pdf/67/11/3818/8810204/3818_1_online.pdf. URL: https://doi.org/10.1063/1.1147276.

[14] J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. 2nd ed. Pearson Education, 2010. ISBN: 978-0805382914.

[15] Franz Schwabl. *Quantenmechanik (QM I)*. 7th ed. Springer Berlin, Heidelberg, 2007. ISBN: 978-3-540-73674-5. DOI: 10.1007/978-3-540-73675-2.

[16] Paul A. Tipler and Ralph A. Llewellyn. *Modern Physics*. Ed. by Clancy Marshall. 5th ed. W.H. Freeman New York, 2008. ISBN: 978-0-7167-7550-8.

[17] NIST. *CODATA Internationally recommended 2018 values of the Fundamental Physical Constants*. 2019. URL: https://physics.nist.gov/cuu/Constants/archive2018.html (visited on 05/09/2024).

[18] Franz Schwabl. *Quantenmechanik für Fortgeschrittene (QM II)*. 5th ed. Springer Berlin, Heidelberg, 2008. ISBN: 978-3-540-85075-5. DOI: 10.1007/978-3-540-85076-2.

[19] Paul Adrien Maurice Dirac and Ralph Howard Fowler. "The quantum theory of the electron". In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 117.778 (1928), pp. 610–624. DOI: 10.1098/rspa.1928.0023. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1928.0023. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1928.0023.

[20] John David Jackson. *Klassische Elektrodynamik*. 4th ed. Walter de Gruyter, Berlin, 2006. ISBN: 978-3-11-018970-4.

[21] B Cagnac. "Hydrogen metrology: up to what limit?" In: *Physica Scripta* 1997.T70 (Jan. 1997), p. 24. DOI: 10.1088/0031-8949/1997/T70/004. URL: https://dx.doi.org/10.1088/0031-8949/1997/T70/004.

[22] Willis E. Lamb and Robert C. Retherford. "Fine Structure of the Hydrogen Atom by a Microwave Method". In: *Phys. Rev.* 72 (3 Aug. 1947), pp. 241–243. DOI: 10.1103/PhysRev.72.241. URL: https://link.aps.org/doi/10.1103/PhysRev.72.241.

[23] K. Gottfried and V.F. Weisskopf. *Concepts of Particle Physics*. 2. Oxford University Press, 1986. ISBN: 9780195365276. URL: https://books.google.de/books?id=KXvoI-m9-9MC.

[24] E. E. Salpeter. "Mass Corrections to the Fine Structure of Hydrogen-Like Atoms". In: *Phys. Rev.* 87 (2 July 1952), pp. 328–343. DOI: 10.1103/PhysRev.87.328. URL: https://link.aps.org/doi/10.1103/PhysRev.87.328.

[25] V. M. Shabaev. "QED theory of the nuclear recoil effect in atoms". In: *Phys. Rev. A* 57 (1 Jan. 1998), pp. 59–67. DOI: 10.1103/PhysRevA.57.59. URL: https://link.aps.org/doi/10.1103/PhysRevA.57.59.

[26] Krzysztof Pachucki and Vladimir A. Yerokhin. "QED Theory of the Nuclear Recoil with Finite Size". In: *Physical Review Letters* 130.5 (Feb. 2023). ISSN: 1079-7114. DOI: 10.1103/physrevlett.130.053002. URL: http://dx.doi.org/10.1103/PhysRevLett.130.053002.

[27] C. Froese Fischer O. Zatsarinny. "DBSR_HF: A B-spline Dirac-Hartree-Fock program". In: *Computer Physics Communications* 202 (2016), pp. 287–303.

[28] Walter R. Johnson. *Potential for a Fermi Charge Distribution*. 2018. URL: https://www3.nd.edu/~johnson/Publications/poten.pdf (visited on 04/16/2024).

[29] F. A. Parpia and A. K. Mohanty. "Relativistic basis-set calculations for atoms with Fermi nuclei". In: *Phys. Rev. A* 46 (7 Oct. 1992), pp. 3735–3745. DOI: 10.1103/PhysRevA.46.3735. URL: https://link.aps.org/doi/10.1103/PhysRevA.46.3735.

[30] Martin G.H. Gustavsson and Ann-Marie Mårtensson-Pendrill. "Four Decades of Hyperfine Anomalies". In: ed. by Per-Olov Löwdin. Vol. 30. Advances in Quantum Chemistry. Academic Press, 1998, pp. 343–360. DOI: https://doi.org/10.1016/S0065-3276(08)60516-X. URL: https://www.sciencedirect.com/science/article/pii/S006532760860516X.

[31] E. Rutherford. "LXXIX. The scattering of α and β particles by matter and the structure of the atom". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 21.125 (1911), pp. 669–688. DOI: 10.1080/14786440508637080. URL: https://doi.org/10.1080/14786440508637080.

[32] R. Hofstadter, H. R. Fechter, and J. A. McIntyre. "High-Energy Electron Scattering and Nuclear Structure Determinations". In: *Phys. Rev.* 92 (4 Nov. 1953), pp. 978–987. DOI: 10.1103/PhysRev.92.978. URL: https://link.aps.org/doi/10.1103/PhysRev.92.978.

[33] C.W. De Jager, H. De Vries, and C. De Vries. "Nuclear charge- and magnetization-density-distribution parameters from elastic electron scattering". In: *Atomic Data and Nuclear Data Tables* 14.5 (1974). Nuclear Charge and Moment Distributions, pp. 479–508. ISSN: 0092-640X. DOI: https://doi.org/10.1016/S0092-640X(74)80002-1. URL: https://www.sciencedirect.com/science/article/pii/S0092640X74800021.

[34] I. Angeli and K.P. Marinova. "Table of experimental nuclear ground state charge radii: An update". In: *Atomic Data and Nuclear Data Tables* 99.1 (2013), pp. 69–95. ISSN: 0092-640X. DOI: https://doi.org/10.1016/j.adt.2011.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0092640X12000265.

[35] Stephan Fritzsche. *JAC: Jena Atomic Calculator. User Guide, Compendium & Theoretical Background*. Helmholtz-Institut Jena. Fröbelstieg 3, Jena, Germany, 2023.

[36] Germund Dahlquist and Åke Björk. *Numerical Methods in Scientific Computing Vol. 1*. Society for Industrial and Applied Mathematics, 2008. ISBN: 978-0-898716-44-3.

[37] Hans-Rudolf Schwarz and Norbert Köckler. *Numerische Mathematik*. 5th ed. Teubner Verlag, Wiesbaden, 2005. ISBN: 978-3-519-42960-8.

[38] Germund Dahlquist and Åke Björk. *Numerical Methods*. Dover Publications, Inc., Mineaola, New York, 2003. ISBN: 978-0-486428-07-9.

[39] E. Borie and G. A. Rinker. "The energy levels of muonic atoms". In: *Rev. Mod. Phys.* 54 (1 Jan. 1982), pp. 67–118. DOI: 10.1103/RevModPhys.54.67. URL: https://link.aps.org/doi/10.1103/RevModPhys.54.67.

[40] Abu Saleh Musa Patoary and Natalia S. Oreshkina. *Finite Nuclear Size Effect to the Fine Structure of Heavy Muonic Atoms*. 2017. arXiv: 1711.06440 [physics.atom-ph].

[41] Savely G. Karshenboim et al. "Theory of Lamb Shift in Muonic Hydrogen". In: *Journal of Physical and Chemical Reference Data* 44.3 (July 2015), p. 031202. ISSN: 0047-2689. DOI: 10.1063/1.4921197. eprint: https://pubs.aip.org/aip/jpr/article-pdf/doi/10.1063/1.4921197/15982436/031202_1_online.pdf. URL: https://doi.org/10.1063/1.4921197.

[42] Robert Waltner. "Paritätsverletzungen in myonischen Atomen". Bachelor's Thesis. Technische Universität Braunschweig, 2022.

[43] G. Feinberg and M. Y. Chen. "$2S_{\frac{1}{2}} \to 1S_{\frac{1}{2}}$ + one − photon decay of muonic atoms and parity-violating neutral-current interactions". In: *Phys. Rev. D* 10 (1 July 1974), pp. 190–203. DOI: 10.1103/PhysRevD.10.190. URL: https://link.aps.org/doi/10.1103/PhysRevD.10.190.

[44] Peter J. Mohr et al. "CODATA Recommended Values of the Fundamental Physical Constants: 2022". Manuscript in preparation.

[45] V M Burke and I P Grant. "The effect of relativity on atomic wave functions". In: *Proceedings of the Physical Society* 90.2 (Feb. 1967), p. 297. DOI: 10.1088/0370-1328/90/2/301. URL: https://dx.doi.org/10.1088/0370-1328/90/2/301.

[46] Oliver Natt. *Physik mit Python*. 2nd ed. Springer Spektrum Berlin, Heidelberg, 2022. ISBN: 978-3-662-66454-4. DOI: 10.1007/978-3-662-66454-4.

[47] Jan Richter et al. “Parity-Violation Studies with Partially Stripped Ions”. In: *Annalen der Physik* 534.3 (2022), p. 2100561. DOI: https://doi.org/10.1002/andp.202100561. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.202100561. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.202100561.

[48] V. A. Dzuba, V. V. Flambaum, and Y. V. Stadnik. “Probing Low-Mass Vector Bosons with Parity Nonconservation and Nuclear Anapole Moment Measurements in Atoms and Molecules”. In: *Phys. Rev. Lett.* 119 (22 Nov. 2017), p. 223201. DOI: 10.1103/PhysRevLett.119.223201. URL: https://link.aps.org/doi/10.1103/PhysRevLett.119.223201.

[49] Per Jönsson et al. “GRASP Manual for Users”. In: *Atoms* 11.4 (2023). ISSN: 2218-2004. DOI: 10.3390/atoms11040068. URL: https://www.mdpi.com/2218-2004/11/4/68.

[50] Hideki Yukawa. “On the Interaction of Elementary Particles. I”. In: *Progress of Theoretical Physics Supplement* 1 (Jan. 1955), pp. 1–10. ISSN: 0375-9687. DOI: 10.1143/PTPS.1.1. eprint: https://academic.oup.com/ptps/article-pdf/doi/10.1143/PTPS.1.1/5310694/1-1.pdf. URL: https://doi.org/10.1143/PTPS.1.1.