UiT The Arctic University of Norway

Faculty of science and technology

Unleashing the VAMPyR

A Python Journey into the Realm of Multiwavelets and Quantum Chemistry

Magnar Bjørgve

A dissertation for the degree of Philosophiae Doctor November 2023

Abstract

This thesis presents `VAMPyR`, an advancement in quantum chemistry that significantly mitigates the challenges inherent in high-precision computational methods. By providing a Python interface to the multiwavelet-based functionalities of `MRCPP`, `VAMPyR` enables a more intuitive, accessible, and efficient approach to quantum chemical calculations, reducing the complexity traditionally encountered with basis sets and lower-level programming languages.

The development and capabilities of `VAMPyR` are underscored, illustrating its application in simplifying the prototyping and implementation of new methods in quantum chemistry. The software's usability is evidenced through successful deployment in educational settings and research projects, thus confirming its potential to enhance productivity and foster innovation in the field.

Acknowledgements

I want to express my sincere gratitude to my supervisors, Luca Frediani, Stig Rune Jensen, and Tor Flå, for their unwavering support and confidence in my abilities. Luca, for the liberty to drive VAMPyR's development, Stig, for the countless invaluable Zoom sessions that kept my research on track, and Tor, for your impactful advice and expertise in guiding this project forward.

A big thank you goes to my family and in-laws for their dedication to our children during the many weekends this project took over. Your support has been a cornerstone, not just for the completion of my thesis but also for maintaining our family's well-being.

I owe a great deal of appreciation to the PhD student community and to my colleagues in the Chemistry Department. Our shared experiences – the laughter during lunches and the joy in social events – have provided a delightful counterweight to the rigors of research.

Similarly, my colleagues at the IT department warrant a special mention. Thank you for your flexibility and understanding, especially when I needed to concentrate on my thesis. Your willingness to accommodate my work schedule has been tremendously helpful. Additionally, providing me access to chat.uit.no that has offered constructive feedback on my writing which has been instrumental.

Lastly, my heartfelt thanks to my wife and children. Your unwavering love and patience have been instrumental to this journey. This achievement is as much a testament to your sacrifices as it is to my own efforts. We can all share in this success together.

List of papers

This thesis is based on the following scientific papers:

1. *"VAMPyR – A high level Python library for mathematical operations in a Multiwavelets representation"*; Magnar Bjørgve, Christian Tantardini, Stig Rune Jensen, Gabriel A. Gerez S., Peter Wind, Roberto Di Remigio Eikås, Evgueni Dinvay and Luca Frediani; *Manuscript in preparation.*

2. *"Kinetic Energy-free Hartree–Fock equations: an integral formulation"*; Stig Rune Jensen, Antoine Durdek, Magnar Bjørgve, Peter Wind, Tor Flå and Luca Frediani; Published in *Journal of Mathematical Chemistry*, 61(2), 343-361 (2023), Springer.

3. *"Future Perspective for Core-Electron Spectroscopy: Breit Hamiltonian in the Multiwavelets Framework"*; Christian Tantardini, Roberto Di Remigio Eikås, Magnar Bjørgve, Stig Rune Jensen and Luca Frediani; Submitted to *Journal of Chemical Theory and Computation.*

4. *"Cavity-free continuum solvation: implementation and parametrization in a multiwavelet framework"*; Gabriel A. Gerez S., Roberto Di Remigio Eikås, Stig Rune Jensen, Magnar Bjørgve and Luca Frediani; Published in *Journal of Chemical Theory and Computation*, 19(7), 1986-1997 (2023), ACS Publications.

other scientific contributions:

1. *"The MRChem multiresolution analysis code for molecular electronic calculations: performance and scaling properties"*; Peter Wind, Magnar Bjørgve, Anders Brakestad, Gabriel A. Gerez S, Stig Rune Jensen, Roberto Di Remigio Eikås and Luca Frediani; Published in *Journal of Chemical Theory and Computation*, 19(1), 137-146 (2022), ACS Publications.

Contents

Abstract **3**

Acknowledgements **5**

1 Introduction **11**

1.1 Why `VAMPyR`? 13

1.2 The Schrödinger Equation: The Essentials 14

1.3 Non-Rigorous and Simplified: Unpacking Multiwavelets 15

1.3.1 Specialized Basis 17

1.3.2 Small Pieces 18

1.3.3 A Bit of Both 20

1.3.4 Separation of Scales and Adaptive Representation . . . 21

2 Partial differential equations and Multiwavelets **23**

2.1 Multiresolution Analysis 23

2.1.1 Extension to d-dimensions 24

2.1.2 Projection onto Scaling and Wavelet Spaces 25

2.1.3 Generalization to d-dimensions 26

2.2 Numerical Grid 26

2.3 Addition of Functions 27

2.4 Multiplication of Functions 27

2.5 Function and Operator Thresholding 28

2.6 Operator representation 29

2.7 Derivative Operators 30

2.8 Convolution Operators 31

2.8.1 The Poisson Equation 32

2.8.2 The Bound State Helmholtz Equation 33

2.8.3 Kernel Separation Using Gaussians 33

3 Introduction to Quantum Chemistry **35**

3.1 The Schrödinger Equation 36

3.2 The Time Independent Schrödinger Equation 36

3.3 The Born–Oppenheimer Approximation 38

3.4 The Variational Principle 39

3.5 Hartree–Fock 40
3.5.1 The Hartree–Fock Equations 41
3.6 Density Functional Theory 41
3.6.1 The Kohn–Sham Equations 42
3.7 Solving the Hartree–Fock and Kohn–Sham Equation in Integral Form 44
3.8 Solvation 45
3.8.1 The Solvation Hamiltonian 46
3.8.2 The Generalized Poisson Equation 47
3.8.3 Integrating and Solving the Reaction Potential in Hartree–Fock and Kohn–Sham Equations 48
3.9 Relativity in Multiwavelets 49
3.9.1 The Dirac Equation 50
3.9.2 Solving the Dirac Equation Using Green's Function and Integral Form 54

4 Very Accurate MultiResolution Python Routines 57
4.1 Installation and Documentation 58
4.2 Key Data Structures 58
4.2.1 Multiresolution Analysis (`MultiResolutionAnalysis<D>`) 59
4.2.2 MultiWavelet Node (`MWNode<D>`) and MultiWavelet Tree (`MWTree<D>`) 59
4.2.3 `FunctionTree<D>` and `OperatorTree<D>` 59
4.3 Intuitive Multiresolution Analysis with `VAMPyR` 60
4.3.1 Interactive Design: `VAMPyR` vs. `MRCPP` 61
4.3.2 Function Projection in `VAMPyR` 61
4.3.3 Arithmetic Operations 63
4.3.4 Derivative Operators 64
4.3.5 Convolution Operators 66
4.4 Implementation 67
4.4.1 Modifying the `FunctionTree<D>` for `VAMPyR` 67
4.4.2 Modifying `MRCPP` Functions for `VAMPyR` 70

5 Papers 73
5.1 Paper 1: `VAMPyR` – A High-Level Python Library for Mathematical Operations in a Multiwavelets Representation 73
5.2 Paper II: Kinetic Energy-free Hartree–Fock equations: an integral formulation 121
5.3 Paper III: Future Perspective for Core-Electron Spectroscopy: Breit Hamiltonian in the Multiwavelets Framework 141
5.4 Paper IV: Cavity-free continuum solvation: implementation and parametrization in a multiwavelet framework 168

Bibliography 181

Chapter 1

Introduction

Quantum chemistry methods are essential tools for understanding the intricate structures and behaviors of molecules, contributing to various scientific domains. The primary contribution of the `MRChem` group lies in computational methodologies, particularly in relation to the precision of calculations.

To avoid confusion, it is important to clarify the term *precision*, which is often mistaken for accuracy. Consider the analogy of throwing darts at a dartboard: accuracy refers to how close the darts land to the bullseye, while precision pertains to how tightly grouped the darts are, regardless of their distance from the bullseye. In the context of quantum chemistry, accuracy is the closeness of a computational model to the Schrödinger equation, while precision relates to the numerical reliability of these calculations. The `MRChem` group addresses precision through the choice of basis.

Historically, the most successful and impactful basis set type has arguably been atom-centered Gaussian functions[1]. However, these functions have limitations, particularly in high-precision calculations, due to the linear dependence of the basis set[2]. High-precision calculations have often been performed using real-space numerical grids. However, the computational cost of these real-space methods has traditionally been prohibitively high, limiting their primary use to benchmarking studies.

Now more computationally feasible real space alternative exist[3–6]. One such approach is based on Alpert's multiwavelets[7]. Multiwavelets provide rigorous error control and are adaptive, meaning they create tight grids where functions are difficult to represent and computational efforts are required, while areas that do not require much effort represented sparcely.

Recent advancements in computing infrastructure, along with the paper by Wind et al.[8], which demonstrates near-linear scaling for molecular electronic structure calculations with the multiwavelet-based code `MRChem`, suggest that real-space methods have potential for larger systems, not just for benchmarking.

The use of multiwavelets in quantum chemistry was first initiated by Harrison et al.[9–11] with their code, `MADNESS`[12–14]. Their success inspired Tor Flå, Luca Frediani and Kenneth Ruud to start the `MRChem` project from UiT. As of this writing, the `MRChem` project includes three code contributions: `MRChem`, `MRCPP`, and `VAMPyR`. These contributions have resulted in various publications related to the basis set precision. For instance benchmarks on total energy [15, 16], linear response properties[17, 18], on scalar relativistic effects[19] and magnetic properties[17].The relationships and functionalities of these codes are depicted in Figure 1.1.

> `MRCPP` (MultiResolution Computation Program Package) is a numerical mathematics library based on multi-resolution analysis and the multiwavelet basis. It offers mathematical operations such as arithmetic, differentiation, integration, and integral operators, all with low-scaling algorithms and rigorous error control. The library is implemented in modern C++ and parallelized using OpenMP, a multi-platform shared-memory parallel programming model[20, 21].
>
> `MRChem` (MultiResolution CHEMistry) builds upon `MRCPP` and is a real-space code for molecular electronic structure calculations such as Hartree–Fock and Density Functional Theory. It is written in modern C++ and parallelized using MPI, making it suitable and scalable for large-scale HPC facilities.
>
> `VAMPyR` (Very Accurate Multiresolution Python Routines), which is the main contribution of this work, binds the functionality of `MRCPP` into a Python library. Its purpose is to reduce the barrier of entry to writing multiwavelet applications.

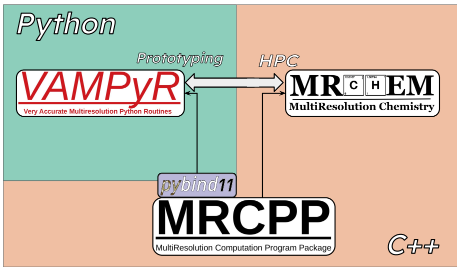

Python
Prototyping
HPC
VAMPyR
Very Accurate Multiresolution Python Routines
MRCHEM
MultiResolution Chemistry
pybind11
MRCPP
MultiResolution Computation Program Package
C++

Figure 1.1: Connections between `MRChem`, `MRCPP` and `VAMPyR`. `MRCPP` implements a high-performance MRA framework with Multiwavelets (MW). `MRChem` uses `MRCPP` as an external library to solve electronic structure systems. `VAMPyR` imports features from `MRCPP` using Pybind11 and brings intuitive design and easy prototyping through Python. Figure adapted from [22].

The thesis begins with the 'Why `VAMPyR`?' section, examining the challenges of quantum chemistry computations and the role that `VAMPyR` plays in addressing these complexities. This setting of context provides insight into the importance and motivations behind `VAMPyR`.

What follows next is a brief introduction to foundational Quantum Chemistry concepts, focusing on the Schrödinger equation, which forms the basis for the methods and problem-solving techniques discussed in the upcoming chapters.

This leads into the next section on Multiwavelets, designed to present fundamental mathematical concepts related to multiwavelets in a non-technical manner. Presenting these concepts in a layman's language makes them accessible to readers who may not have a strong background in mathematics.

Having established the basic content and context, the thesis then progresses into specific chapters:

> Chapter 2 provides an introduction to the mathematical concepts central to multiwavelets, setting the stage for computational methodologies reliant on these principles.
>
> Chapter 3 offers an introduction to mainstay quantum chemistry methods. This includes a discussion on the Schrödinger equation, along with simplifications such as the Born–Oppenheimer approximation, Hartree–Fock, and Density Functional Theory. Additionally, the chapter outlines how these problems are solved within a multiwavelet framework. The chapter also covers essential elements such as solvation and relativity, providing an overview of their corresponding equations and how those are addressed within the multiwavelet framework.
>
> Chapter 4 gives an overview of the advantages with `VAMPyR`, highlighting its design and usability. It also presents some details on important datatypes shared between `MRCPP` and `VAMPyR` along with some implementation details.

1.1 Why `VAMPyR`?

Developing new methods in quantum chemistry codes is a significant challenge. The inherent complexity intrinsic to quantum chemistry itself stems from its intricate equations and their interactions. Accidental complexity arises not only from the use of low-level programming languages, but also from the use of basis sets in quantum chemistry. Traditionally used programming languages, such as C, C++, and Fortran, introduce an additional layer of complexity. These complexities exist before even considering aspects of optimizing code performance and efficiency. Moreover, when working with basis sets, making choices on the kinds of basis sets to use and the optimization of these basis sets can add further accidental complexity.

Multiwavelet-based codes, namely `MRCPP` and by extension, `MRChem`, which builds upon `MRCPP`, offer a distinct advantage. They provide a more direct and intuitive correspondence between the mathematical operations of quantum chemistry equations and their coded forms. This approach effectively mitigates many of the accidental complexities traditionally associated with using basis sets.

However, the use of multiwavelet-based codes does introduce some accidental complexities of their own, particularly related to techniques for solving partial differential equations. Even so, these complexities are considerably less extensive compared to those imposed by other basis sets.

When developing new methods using `MRCPP` or `MRChem`, further accidental complexities emerge. The intricacies of the C++ language, performance optimization challenges, and the navigation of an expansive existing codebase present considerable barriers. For new developers, and even experts of the field, the question of "Where should I start?" when looking to add new features or enhancements, can be daunting. `VAMPyR` seeks to address these challenges by providing a more accessible Python interface to the mathematical functionality of `MRCPP`, simplifying the process and contributing to the ongoing efforts to make computational quantum chemistry more user-friendly.

Addressing the accidental complexities associated with quantum chemistry computations, `VAMPyR` offers an accessible suite of mathematical tools based on `MRCPP`, made available in Python. This not only simplifies its usage but also ensures a close link between equations and code, providing a more intuitive understanding of mathematical operations. One significant advantage of `VAMPyR` lies in its counteraction to the accidental complexity that large codebases can introduce. It achieves this through its simplicity and modularity. In `VAMPyR`, new features can be prototyped independently, effectively lowering both computational and cognitive barriers. This approach is especially helpful for those less experienced in the field, promoting more manageable and efficient experimentation.

The utility and simplified application of `VAMPyR` have been practically exemplified in several contexts. Its applicability has been demonstrated in a Bachelor's project and a Master's project[23], not to mention its deployment in producing a solvation paper[24]. Initial implementations of `ReMRChem`[22, 25] and time propagation methods using `VAMPyR`[22] further testify to its versatility. Its educational value has been acknowledged through its inclusion in a tutorial session for NMQC[26] and the 2021 Hylleraas School[27].

1.2 The Schrödinger Equation: The Essentials

To understand complex systems like airplanes, engineers break them down into smaller components—wings, engines, cockpit—and analyze each to understand its function and behavior. Similarly, scientists dissect nature to comprehend the laws that govern it. In chemistry, this analytical approach extends to the level of atoms, electrons, and nuclei.

Traditional experimental chemistry relies on beakers and test tubes. However, advancements in computational power have paved the way for virtual experiments, enabling detailed scrutiny of each particle's behavior within a computer. This virtual laboratory is the realm of quantum chemistry, anchored by the Schrödinger equation,

$$H\Psi = E\Psi, \tag{1.1}$$

which serves as the rule book for the behavior of atoms and electrons at the quantum level[1]. The wavefunction, Ψ, contains all information about the system it represents.

All physical observable properties, such as position x, momentum p, and energy E, can be extracted from the wavefunction Ψ. This extraction is achieved using their corresponding operators $x \to \hat{x}$, $p \to \hat{p}$, and $E \to \hat{H}$. The expectation value, which represents the mean value of these physical properties, can then be calculated as

$$\langle \Omega \rangle = \langle \Psi | \hat{\Omega} | \Psi \rangle. \tag{1.2}$$

It's important to note that the physical information extracted from the wavefunction represents the expectation or mean value of these properties. This means that the values we compute provide an average behavior of the system under study, rather than precise measurements of individual particles at specific moments.

The real challenge here is solving the Schrodinger equation, determine Ψ. Despite its simple appearance, the Schrödinger equation is analytically solvable only for systems with two or fewer particles. For larger systems, numerical computational methods are required. Yet even advanced numerical techniques are often insufficient for tackling complex molecules. Therefore, simplifications are commonly used, traditionally branching into two main approaches: wave function methods and density functional theory (DFT).

Wave function methods range from low-accuracy techniques like Hartree–Fock to high-accuracy methods such as coupled cluster. On the other hand, DFT uses the density of the system to extract the same information as the wavefunction, albeit based on certain approximations due to unknown exact functional forms. These methods primarily pertain to the accuracy of the model. For the precision of the solution, another essential component comes into play: basis sets.

1.3 Non-Rigorous and Simplified: Unpacking Multiwavelets

Some chemical experiments are either impossible or, at the very least, impractical to do in a laboratory. Luckily, we have an amazing tool in our arsenal—computers. Unfortunately, computers can't do chemistry, or at least not directly, as they cannot think. Computers compute and humans think, so to use

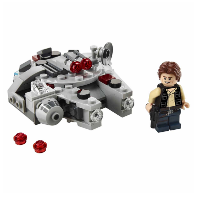
[1]Ignoring relativistic effects which we'll touch upon in section 3.9

the computer, we humans need to reformulate the chemical problems into computational problems. We start this process by figuring out the mathematics that best describe the chemical systems we wish to explore, which usually end up being some set of partial differential equations. For chemists, the partial differential equations typically boil down to either the Hartree–Fock equations (see section 3.5.1) or the Kohn–Sham equations (see section 3.6.1)

Once we have some partial differential equations we wish to solve, we need to figure out how to convert the problem with a partial differential equation into a computable problem. There are many approaches. One approach is to introduce a basis set. The basis set helps us represent mathematical functions as a set of values with rules that can be implemented into a computer program. The choice of basis set is not arbitrary; some basis sets are great for quick and dirty calculations while others are great for precise but slower calculations. So with a basis set, we can transform the mathematical problem into numbers, we can then write computer programs that tell the computer what to compute using those numbers, and this is how we solve the partial differential equations. I like to think about basis sets as a set of Lego blocks, and the programs we write to solve the mathematical equations as the building instructions.

Lego blocks can be used to build more or less whatever you like, depending only on the pieces available and the building instructions you choose to follow. In 1.2, the first Figure 1.2a is a picture of the Millennium Falcon and in Figures 1.2b and 1.2c, the Millennium Falcon is replicated in Lego. In Figure 1.2b, the copy uses few pieces with simple building instructions, and in Figure 1.2c, there are thousands of pieces, with elaborate building instructions, and building it takes a significantly longer time than the crude copy in Figure 1.2b.

(a) Exact


(b) Crude

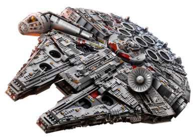

(c) Precise

Figure 1.2: The Millennium Falcon and replicas in Lego

Now if we think about Lego or some type of generic building block, there are three ways to make precise builds. First, we can use specialized pieces, pieces that to begin with are similar to what we wish to build. Another way is to use a fixed size cubic piece. If we want something crude, we can use a few big cubes, and if we want something accurate, we use more, only smaller, pieces. The third option is to combine the two approaches to use some special pieces but also have the option to use smaller pieces if we need higher precision. We can think of basis sets in a similar manner. In the following sections, we will explore each approach, with multiwavelet glasses on, to approximate the

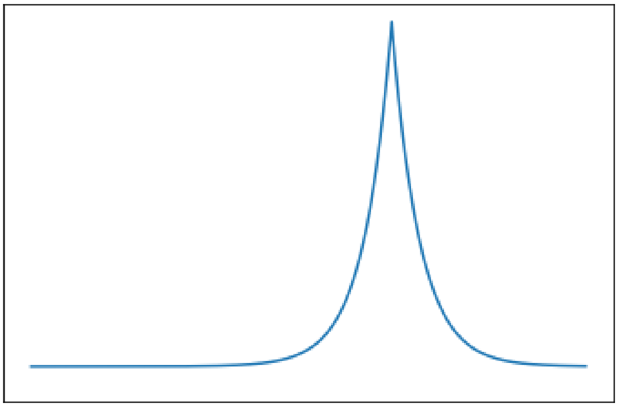


Figure 1.3: The Slater Function

Slater function in Figure 1.3.

1.3.1 Specialized Basis

First, let's build a representation of the Slater function using a set of $k+1$ basis functions. We'll call these basis functions *scaling functions* $\{\varphi_i\}_{i=0}^k$. The scaling functions we'll use are linearly independent, orthonormal polynomial functions defined between $[0,1]$. Linear independence means it's *impossible* to replicate any of the basis functions in the set using the other functions in the set. This means each basis function adds something unique to the set. Then, using the basis set to approximate a function, the approximation can be improved by adding basis functions to the basis set.

The Slater function (Figure 1.3), or any function $f(x)$, can be approximated in terms of the scaling functions $\{\varphi_i\}_{i=0}^k$

$$f(x) \approx f_k(x) = \sum_{i=0}^{k} s_i \varphi_i(x), \tag{1.3}$$

where the expansion coefficients, s_i, are calculated through the projection integral

$$s_i = \int f(x)\varphi_i(x)\,\mathrm{d}x. \tag{1.4}$$

In the projection integral (1.4), we calculate the overlap between the basis function and the function we wish to approximate. For the sake of this discussion we assume that the function $f(x)$ is normalized. That is, $\int |f(x)|^2 = 1$. Then, the expansion coefficient, s_i, has a value between -1 and 1, where 0 means there is no overlap between the function $f(x)$ and the scaling function $\varphi_i(x)$. When it is 1 it is exactly the same and when it is -1 the functions are the same but with opposite signs

A set of functions that satisfy our requirements for scaling functions are the Legendre Polynomials—see Figure 1.4 for an example. In Figure 1.5, we see how 10, 20 and 40 Legendre polynomials approximate the Slater function from Figure 1.3. It's clear that the more scaling functions we use, the better the approximation becomes. But even with 40 Legendre polynomials, we see some oscillations around the edges of the figure, and the sharp part of the Slater function isn't well represented. So if we need a better approximation, we need to use more than 40 scaling functions, which increases the cost of the representation. Moreover most of this effort is required to get correctly zero away from the cusp.

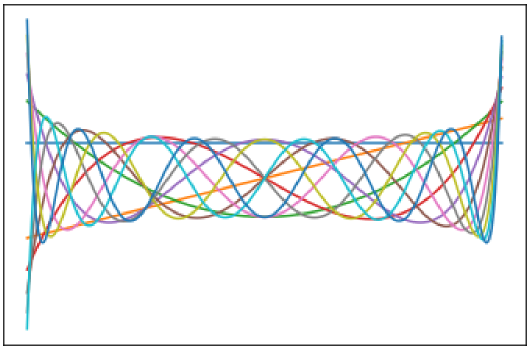


Figure 1.4: 10 Legendre Polynomials

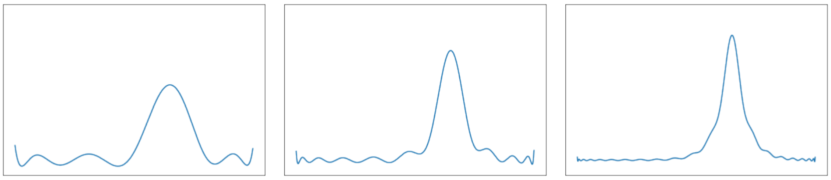


Figure 1.5: The Slater function from Figure 1.3 represented by 10, 20 and 40 Legendre Polynomials

1.3.2 Small Pieces

The second way of using building blocks is where we use only one simple cube, then increase the precision by using smaller cubes. In the basis set world, the equivalent is to use a single scaling function $\varphi(x) = \varphi_0(x)$ and set it to a constant, then increase the precision of the representation by limiting the area in which the scaling function is defined. If the scaling function we use to approximate the Slater function is defined between $[0, 1]$, we will only get a line, which in the case of a constant scaling function, represents the average of the function in the interval. To improve the approximation, we split the area into two parts, $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$, effectively doubling the number of scaling functions. Then we use the constant to approximate the function within each area, and then we'll get two lines—still a bad approximation, but better than a single line. To improve the approximation, we continue this doubling recursively, where the approximation improves at each step. In principle, this can be done infinitely many times, and in that case, the representation will be exact. Next we'll introduce some terminology for this approach.

When one scaling function cover the entire area $[0, 1]$, we say the approximation is on scale $n = 0$. When we split the area into two, with one scaling function in each part, $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$, we say the approximation is on scale $n = 1$. Then if continue this process n times, we say that the approximation is on scale n. The scaling function at scale n, φ_l^n, then takes the form

$$\varphi_l^n(x) = 2^{n/2}\varphi(2^n x - l), \tag{1.5}$$

where the factor 2^n in front of the argument, x, compresses the function, the factor $2^{n/2}$ conserves the norm and, l, is the translation index $0 \leq l \leq 2^n - 1$. Translates the function within the interval $[0, 1]$, $l = 0$ the function is placed all the way to the left and $l = 2^n - 1$ it is all the way to the right. The functions, $f(x)$, is then expressed in terms of our scaling function on scale n as

$$f(x) \approx f^n(x) = \sum_{l=0}^{2^n-1} s_l^n \varphi_l^n(x), \tag{1.6}$$

where l is the translation index and the expansion coefficients, s_l^n, are obtained by projection integral

$$s_l^n = \int f(x)\varphi_l^n(x)\,\mathrm{d}x. \tag{1.7}$$

In Figure 1.6 we see how the approximation of the Slater function 1.3 improves the further up in scale we go. Here, the approximation is plotted on scales $n = 2, 3$ and 4, and we must go a lot further up in scale to accurately represent the Slater function.

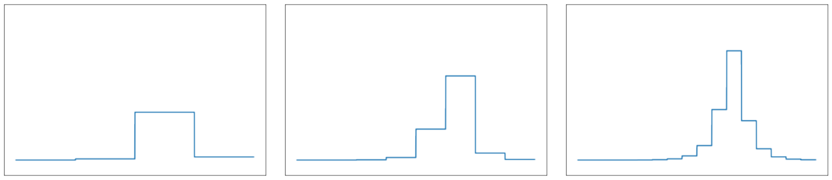


Figure 1.6: 1 scaling function (a constant) with scales 2, 3 and 4.

1.3.3 A Bit of Both

A third alternative is to mix the two approaches above by using, $k+1$, scaling functions, $\{\varphi_i\}_{i=0}^{k}$. Then improve the approximation by increasing the scale, n. The function approximation then looks like this,

$$f(x) \approx f_k^n(x) = \sum_{l=0}^{2^n-1} \sum_{i=0}^{k} s_{i,l}^n \varphi_{i,l}^n(x), \tag{1.8}$$

here we have introduced the notation, f_k^n, which means we have projected a function onto a basis of order k on scale n. The scaling functions $\varphi_{i,l}^n$ are obtained by translation and dilation of the $k+1$ scaling functions, $\{\varphi_i\}_{i=0}^{k}$,

$$\varphi_{i,l}^n(x) = 2^{n/2}\varphi_i(2^n x - l), \tag{1.9}$$

and the expansion coefficients, $s_{i,l}^n$, are calculated using the projection integral

$$s_{j,l}^n = \int f(x)\varphi_{j,l}^n(x)\,\mathrm{d}x. \tag{1.10}$$

In Figure 1.8 we approximate the Slater function from figure 1.3. We use 5 scaling functions then improve the approximation by increasing the scales one level at a time with $n = 2, 3$ and 4. This approach appears to offer a sweet spot when it comes to representing a function using basis sets. For all approaches discussed this far, a clear limitation is that when we want to do an accurate representation of a sharp function, like the Slater function, we see that we need to go to very high scales to represent the cusp accurately, which increases the cost of the representation significantly. The solution to this problem starts at separating the scales.

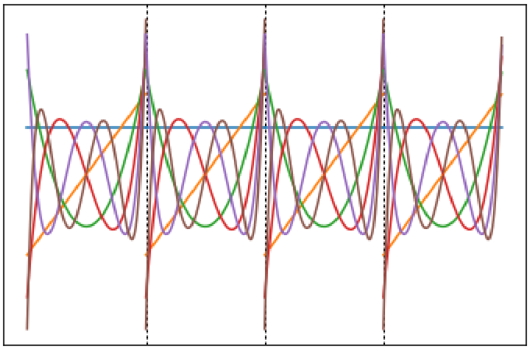


Figure 1.7: 6 scaling functions $\varphi_{i,l}^n$ at scale $n = 2$

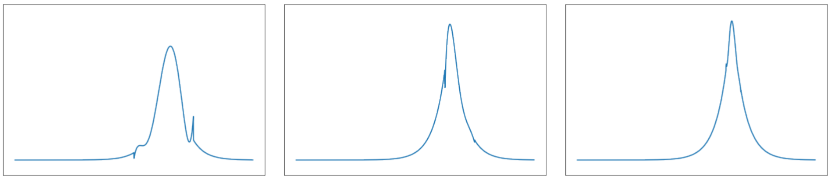


Figure 1.8: Slater function approximated using 5 scaling functions at scales 2, 3 and 4

1.3.4 Separation of Scales and Adaptive Representation

Above we illustrated how a fixed number of basis functions can be used to approximate a function to an arbitrarily high precision by increasing the scale, n, of the projection. Next, we'll look into achieving the same approximation as above, but with a slightly different approach. Namely, by improving the approximation one layer (scale) at a time. We begin by rewriting the function approximation, f_k^n, from equation (1.8) as

$$f_k^n(x) = f_k^0(x) + \sum_{n'=0}^{n} df_k^{n'}(x), \tag{1.11}$$

where f_k^0 is the function projected onto the scaling basis (1.9) at scale $n = 0$. df_k^n is the difference between the projection onto the scaling basis at scales $n+1$ and n,

$$df_k^n(x) = f_k^{n+1}(x) - f_k^n(x). \tag{1.12}$$

The difference term df^n can be calculated directly by projecting the function onto what we call a *wavelet* basis $\{\psi_{i,l}^n\}$. The wavelet basis on scale n, similar to the scaling functions in equation (1.9), are obtained from a set of $k+1$ wavelet functions $\{\psi_i\}_{i=0}^k$ as

$$\psi_{i,l}^n(x) = 2^{n/2}\psi_i(2^n x - l). \tag{1.13}$$

then df^n is defined as

$$df^n(x) = \sum_{l=0}^{2^n-1} \sum_{i=0}^{k} w_{i,l}^n \psi_{i,l}^n(x), \tag{1.14}$$

and $w_{i,l}^n$ is calculated from the projection integral

$$w_{j,l}^n = \int f(x)\psi_{j,l}^n(x)\,\mathrm{d}x. \tag{1.15}$$

The result of this is that functions projected onto the scaling basis in figure 1.7 and in figure 1.9 are equal. We call this a multiresolution representation.

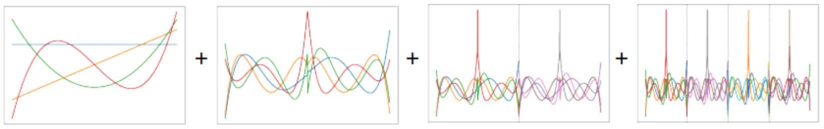


Figure 1.9: High-resolution basis

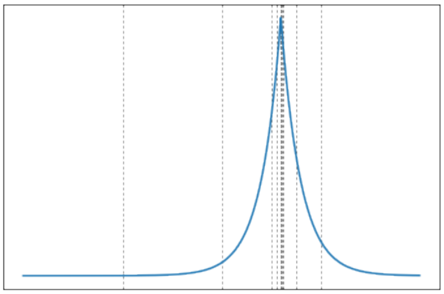


Figure 1.10: Slater function projected onto an adaptive multiwavelet basis, with numerical precision 1.0×10^{-3}.

The advantage of this approach lies in its capability to enhance precision incrementally, one layer at a time. This process is facilitated by measuring the wavelet coefficients. It is indicated that if the magnitude of the coefficients is smaller than a certain threshold, ϵ, the magnitude of the wavelet coefficients on the subsequent scale will correspondingly diminish. This triggers a stop in the refinement process along that branch.

$$|w_l^n| < 2^{-n/2}\epsilon|f| \tag{1.16}$$

For a visual representation of this process, see Figure 1.10. The vertical lines depict the grid used to embody the function, with the tightest grid situated around the cusp of the Slater function. Simultaneously, the smoother portions of the function are represented on a sparser grid.

Chapter 2

Partial differential equations and Multiwavelets

2.1 Multiresolution Analysis

We first define the scaling space V_k^n as outlined in Alpert et al. [28]. Specifically, V_k^n is a space of piecewise polynomial functions characterized as follows:

$$\begin{aligned} V_k^n = \{f :\ & \text{the restriction of } f \text{ to the interval } (2^{-n}l, 2^{-n}(l+1)) \\ & \text{is a polynomial of degree less than } k+1, \text{ for } l = 0, \dots, 2^n - 1, \\ & \text{and } f \text{ vanishes elsewhere}\}. \end{aligned} \tag{2.1}$$

The space V_k^n has dimension $2^n(k+1)$ and is characterized by an infinite, ordered set of nested subspaces,

$$V_k^0 \subset V_k^1 \subset \cdots \subset V_k^n \subset \cdots , \tag{2.2}$$

Defining the space $V_k = \cup_{n=0}^{\infty} V_k^n$, we observe that V_k is dense in $L^2([0,1])$.

Building on this foundation, we introduce the scaling functions that span V_k^0, denoted by $\{\phi_i\}_{i=0}^k$. These functions serve as a basis for constructing the multiresolution analysis, with scaling functions at higher resolutions V_k^n for $n > 0$ generated through dilations and translations:

$$\phi_{i,l}^n(x) = 2^{n/2}\phi_i(2^n x - l), \quad i = 0, \dots, k, \quad l = 0, \dots, 2^n - 1. \tag{2.3}$$

They satisfy the orthogonality relation,

$$\langle \phi_{i,l}^n, \phi_{j,m}^n \rangle = \delta_{i,j}\delta_{l,m}, \tag{2.4}$$

To capture features not represented by the scaling functions, we introduce the 'wavelet space' W_k^n, defined as the orthogonal complement of V_k^n in V_k^{n+1}:

$$W_k^n = V_k^{n+1} \ominus V_k^n. \tag{2.5}$$

Similar to the scaling functions, W_k^n is spanned by a set of wavelet functions $\{\psi_{i,l}^n\}_{i=0}^k, l = 0, \dots, 2^n - 1$, generated through dilations and translations:

$$\psi_{i,l}^n(x) = 2^{n/2}\psi_i(2^n x - l). \tag{2.6}$$

These wavelet functions are not only mutually orthogonal but also orthogonal to the scaling functions at all scales lower or equal their own:

$$\langle \psi_{i,l}^n, \psi_{j,m}^{n'} \rangle = \delta_{i,j}\delta_{l,m}\delta_{n,n'}, \text{ if } n' \leq n. \tag{2.7}$$

An essential feature of these wavelet functions is that they have 'vanishing moments', meaning they are orthogonal to the polynomials in the corresponding scaling space. This is crucial for the numerical efficacy of approximations and operations.

In summary, the multiresolution structure is recursively defined as follows:

$$V_k^N = V_k^n \oplus W_k^n \oplus W_k^{n+1} \oplus \cdots \oplus W_k^{N-1}. \tag{2.8}$$

2.1.1 Extension to d-dimensions

The theory of multiresolution analysis naturally extends to d-dimensions through tensor products. In this framework, the d-dimensional scaling space $V_k^{n,d}$ is defined as:

$$V_k^{n,d} = \bigotimes^d V_k^n, \tag{2.9}$$

and the basis functions for this space are tensor products of the one-dimensional bases:

$$\phi_{\mathbf{j},\mathbf{l}}^{n,d}(\mathbf{x}) = \prod_{p=1}^d \phi_{j_p,l_p}^n(x_p). \tag{2.10}$$

To facilitate a uniform notation that accommodates both wavelet and scaling functions, we introduce a generalized wavelet function $\phi_{j,l}^{\alpha,n}$:

$$\phi_{j_p,l_p}^{\alpha_p,n} = \begin{cases} \phi_{j_p,l_p}^n & \text{if } \alpha_p = 0, \\ \psi_{j_p,l_p}^n & \text{if } \alpha_p = 1. \end{cases} \tag{2.11}$$

This generalized function allows us to write the d-dimensional wavelet functions as:

$$\Psi_{\boldsymbol{j},\boldsymbol{l}}^{n,\boldsymbol{\alpha}}(\boldsymbol{x}) = \prod_{p=1}^d \phi_{j_p,l_p}^{\alpha_p,n}(x_p), \tag{2.12}$$

where α encodes the 2^d different combinations of wavelet and scaling functions.

Upon expanding the tensor product in the definition of the wavelet space, the pure scaling term is recognized as $V_k^{n,d}$. This makes the wavelet space $W_k^{n,d}$ consist of all remaining terms that contain at least one wavelet space. Consequently, the dimensionality of the wavelet space is $2^d - 1$ times higher than the corresponding scaling space.

An immediate implication of extending to d-dimensions is the exponential growth in computational complexity, primarily due to the increase in basis functions. Each hypercube at scale n contain $(k+1)^d$ basis functions, and the total number scales as 2^{dn}.

2.1.2 Projection onto Scaling and Wavelet Spaces

Projecting functions onto the scaling and wavelet spaces allows us to approximate arbitrary functions within the multiresolution framework. Consider a function $f(x)$ in one dimension.

The scaling projector P^n allows us to project $f(x)$ onto the scaling space V_k^n:

$$P^n f(x) = f^n(x) = \sum_{l=0}^{2^n-1} \sum_{j=0}^{k} s_{j,l}^{n,f} \phi_{j,l}^n(x), \tag{2.13}$$

where $s_{j,l}^{n,f}$ are the scaling coefficients, computed as:

$$s_{j,l}^{n,f} = \int f(x)\phi_{j,l}^n(x)\, dx. \tag{2.14}$$

Here, l serves as the translation parameter ranging from 0 to $2^n - 1$, and j denotes the polynomial order parameter ranging from 0 to k.

Similarly, the wavelet projector Q^n allows us to project $f(x)$ onto the wavelet space W_k^n:

$$Q^n f(x) = df^n(x) = \sum_{l=0}^{2^n-1} \sum_{j=0}^{k} w_{j,l}^{n,f} \psi_{j,l}^n(x), \tag{2.15}$$

with wavelet coefficients $w_{j,l}^{n,f}$ defined as:

$$w_{j,l}^{n,f} = \int f(x)\psi_{j,l}^n(x)\, dx. \tag{2.16}$$

Again, l and j have the same roles and limitations as in the scaling projection.

In equations (2.13) and (2.15) we see how the function f are projected onto the scaling space V_k^n and W_k^n respectively. As we can do with scaling and wavelet spaces in (2.8). The function f^n can also be represented in multiresolution components as

$$f^n(x) = P^n f(x) = \left(P^{n-1} + Q^{n-1}\right) f(x) = \left(P^0 + \sum_{n'=0}^{n-1} Q^{n'}\right) f(x) \tag{2.17}$$

$$= f^0(x) + \sum_{n'=0}^{n-1} df^{n'}(x) \tag{2.18}$$

2.1.3 Generalization to d-dimensions

The principles can be extended to d-dimensions, where the function $f(\mathbf{x})$ can be approximated within the d-dimensional scaling space $V_k^{n,d}$ using the scaling projector P^n. This is expressed as:

$$P^n f(\mathbf{x}) = f^n(\mathbf{x}) = \sum_{\mathbf{l},\mathbf{j}} s_{\mathbf{j},\mathbf{l}}^{n,f} \Phi_{\mathbf{j},\mathbf{l}}^n(\mathbf{x}), \tag{2.19}$$

where $\mathbf{l} = (l_1, \ldots, l_d)$ represents the translation vectors with $0 \leq l_p \leq 2^n - 1$ for each dimension p, and $\mathbf{j} = (j_1, \ldots, j_d)$ are the polynomial order parameters with $0 \leq j_p \leq k$. The d-dimensional scaling coefficients $s_{\mathbf{j},\mathbf{l}}^{n,f}$ are obtained by:

$$s_{\mathbf{j},\mathbf{l}}^{n,f} = \int f(\mathbf{x}) \Phi_{\mathbf{j},\mathbf{l}}^n(\mathbf{x})\, d\mathbf{x}. \tag{2.20}$$

Similarly, the d-dimensional wavelet projector Q^n allows us to project $f(\mathbf{x})$ onto the wavelet space $W_k^{n,d}$:

$$Q^n f(\mathbf{x}) = df^n(\mathbf{x}) = \sum_{\mathbf{l},\mathbf{j}} \sum_{\alpha=1}^{2^d-1} w_{\mathbf{j},\mathbf{l}}^{\alpha,n,f} \Psi_{\mathbf{j},\mathbf{l}}^{\alpha,n}(\mathbf{x}), \tag{2.21}$$

The d-dimensional wavelet coefficients are computed as:

$$w_{\mathbf{j},\mathbf{l}}^{\alpha,n,f} = \int f(\mathbf{x}) \Psi_{\mathbf{j},\mathbf{l}}^{\alpha,n}(\mathbf{x})\, d\mathbf{x}. \tag{2.22}$$

The multiresolution representation is given by:

$$f^n(\mathbf{x}) = f^0(\mathbf{x}) + \sum_{n'=0}^{n-1} df^{n'}(\mathbf{x}) \tag{2.23}$$

2.2 Numerical Grid

The scaling coefficients are calculated numerically using a Gauss–Legendre quadrature. This means the integral in (2.14) is approximated as the following sum:

$$s_{j,l}^{n,f} \approx 2^{-n/2} \sum_{q=0}^{k_q-1} w_q f(2^{-n}(x_q + l))\phi_j(x_q) \tag{2.24}$$

In d-dimensions, the approximation is extended as follows:

$$s_{\mathbf{j},\mathbf{l}}^{n,f} = 2^{-nd/2} \sum_{\mathbf{q}} f(2^{-n}(\mathbf{x}_q + \mathbf{l})) \prod_{d=1}^{D} \omega_{q_d} \phi_{j_p}(x_{q_d}) \tag{2.25}$$

In practical terms, this means that when a function is projected onto V_k^n, the function f is sampled at grid points x_q. We denote this vector as f_l. To obtain the scaling coefficients, a function-to-scaling transformation matrix $T_{v\leftarrow f}$ is applied:

$$s_l^n = T_{v\leftarrow f} f_l \tag{2.26}$$

Conversely, we can retrieve the function values at the grid points by applying the inverse of the transformation matrix to the scaling coefficients. This operation can be represented as:

$$f_l = T_{f\leftarrow v} f_l = T_{v\leftarrow f}^{-1} s_l^n \tag{2.27}$$

2.3 Addition of Functions

The addition of functions is straightforward, represented by the following mappings:

$$V_k^n + V_k^n \to V_k^n \tag{2.28}$$

$$W_k^n + W_k^n \to W_k^n \tag{2.29}$$

For instance, consider adding two functions f^n and g^n to obtain a new function h^n. The function h^n is then simply expressed as:

$$h^n(x) = \sum_{l=0}^{2^n-1} \sum_{j=0}^{k} (s_{j,l}^{n,f} + s_{j,l}^{n,g}) \phi_{j,l}^n(x) \tag{2.30}$$

The same principle applies to dh^n.

2.4 Multiplication of Functions

The product of functions is represented by the following mapping:

$$V_k^n \times V_k^n \to V_{2k}^n \tag{2.31}$$

This implies that the new function at scale n should be represented by double the number of scaling functions, requiring a projection back to the scaling space. Following Beylkin[29], we propose that the product spills over onto the finer scales:

$$V_k^n \times V_k^n \to \otimes \bigoplus_{n'=n}^{\infty} W_k^{n'} \tag{2.32}$$

This is truncated to:

$$V_k^n \times V_k^n \to V_k^n \otimes \bigoplus_{n'=n}^{N} W_k^{n'} = V_k^N \tag{2.33}$$

where the scale N is generally higher than n. To multiply the functions f^n and g^n, we adopt the following general method. The first step invokes upsampling the scaling coefficients of f^n and g^n:

$$\uparrow^N s_{j,l}^{n,f} = s_{j,l}^{N,f} \qquad\qquad \uparrow^N s_{j,l}^{n,g} = s_{j,l}^{N,g} \tag{2.34}$$

Next, the scaling-to-function transformation (as per equation (2.27)) is performed for both functions, leading to the upsampled grid points:

$$f_l^N = T_{f\leftarrow v} s_{j,l}^{N,f} \qquad\qquad g_l^N = T_{f\leftarrow v} s_{j,l}^{N,g} \tag{2.35}$$

Carrying this forward, a pointwise multiplication is executed between these grid points:

$$h_l^N = f_l^N \times g_l^N \tag{2.36}$$

The final step in this procedure involves obtaining the scaling coefficients via a function-to-scaling transformation. This sequence of operations effectively multiplies the functions.

2.5 Function and Operator Thresholding

A crucial aspect of achieving fast algorithms is thresholding the functions and operators. Specifically, the input function f, output function g, and operator $\hat{O}$ are thresholded to a predetermined relative precision ϵ. We require

$$||f - f_\epsilon||_2 < \epsilon ||f||_2, \tag{2.37}$$

and it can be shown that such a precision is guaranteed when we have the following bound on the wavelet coefficients ω:

$$||\omega_{\mathbf{l}}^n||_2 < 2^{-dn/2} \epsilon ||f||_2. \tag{2.38}$$

Numerical studies have shown that

$$||\omega_1^n||_2 < 2^{-n/2}\epsilon||f||_2 \tag{2.39}$$

is generally sufficient, as long as the polynomial order k is sufficiently large, with higher precision requiring larger k.

A similar kind of thresholding is done for operator applications to calculate

$$\hat{g}^n = \hat{O}^n f^n. \tag{2.40}$$

The main difference here lies in determining which terms to include in the output $\hat{g}^n$. Detailed information, along with numerical tests, is well explained in [30] and in the supplementary information to Harrison et. al.[10].

2.6 Operator representation

We write the operator T projected onto the scaling basis at scale n, as, T^n. T^n can then be applied onto the function f^n which yields us $\hat{g}^n$.

$$\hat{g}^n = T^n f^n \tag{2.41}$$

Here we differentiate between $\hat{g}^n$ and g^n. The former, $\hat{g}^n$, is calculated by applying an operator represented in a multiwavelet basis onto a function represented in a multiwavelet basis. Whereas, the latter, g^n, would be the convolution performed analytically then projected onto the multiwavelet basis.

As for functions projected onto a scaling basis, operators projected onto a scaling basis can also be expanded from a pure scaling representation to a scaling plus wavelet representation. The first step to this is,

$$T^{n+1} = T^n + A^n + B^n + C^n. \tag{2.42}$$

Here A^n, B^n and C^n are the kernel represented by various combinations of scaling and wavelet functions. The pure scaling term T^n is rather expensive to apply, and they are applied onto f^n and df^n and gives us

$$\hat{g}^n = T^n f^n \tag{2.43}$$

$$\tilde{g}^n = C^n df^n \tag{2.44}$$

$$d\tilde{g}^n = A^n df^n + B^n f^n \tag{2.45}$$

which we use to construct $\hat{g}^{n+1}$ as

$$\hat{g}^{n+1} = \hat{g}^n + \tilde{g}^n + d\tilde{g}^n. \tag{2.46}$$

The T^n part of the operator is represented purely by scaling functions, which makes it significantly more costly to represent and apply than the A^n, B^n and C^n parts. The latter parts are partly or fully represented in the wavelet basis, which have the property of vanishing moments. This property leads to sparse representations of smooth operators such as potentials and differential operators. So to speed up the operator application do we focus on decomposing, T^n. Which can be expanded recursively until we hit the coarsest scale, doing that we get what is called the *non-standard form* of the operator

$$T^{n+1} = T^0 + \sum_{n'=0}^{n} \left(A^{n'} + B^{n'} + C^{n'} \right) \tag{2.47}$$

This leads to $\hat{g}^{n+1}$ being calculated as

$$\hat{g}^{n+1} = \hat{g}^0 + \sum_{n'=0}^{n} \left(\hat{g}^{n'} + d\tilde{g}^{n'} \right). \tag{2.48}$$

2.7 Derivative Operators

Due to the discontinuous nature of the multiwavelet basis, derivative operators do not exist in conventional sense[28]. We employ two methods suited for different situations in computing the derivatives:

1. A method based on the concept of a weak derivative through integration by parts, ideal for handling non-continuous functions when a single derivative is needed. This method is discussed in detail in [28].

2. A B-spline derivative operator introduced in [31], which is designed to differentiate continuous functions. It works well for higher order derivatives.

The first method, following [28], is designed for non-continuous functions and is based on the concept of a weak derivative through integration by parts. The scaling coefficients of $\frac{df}{dx}$ are calculated as:

$$s_{j,l}^{n,\frac{df}{dx}} = f(x)\phi_{j,l}^n(x)\Big|_{2^{-n}l}^{2^{-n}(l+1)} - \int_{2^{-n}l}^{2^{-n}(l+1)} f(x)\phi_{j,l}^n(x)dx \tag{2.49}$$

The integral here (right hand side of (2.49)) is manageable if we substitute $f(x)$ with $f^n(x)$:

$$\int_{2^{-n}l}^{2^{-n}(l+1)} f^n(x)\phi_{j,l}^n(x)dx = 2^n \sum_{i=0}^{k} s_{i,l}^{n,f} K_{j,i} \tag{2.50}$$

$$\tag{2.51}$$

where $K_{j,i}$ can be computed as the following integral:

$$K_{j,i} = \int_0^1 \phi_i(x) \frac{d}{dx} \phi_j(x) dx \tag{2.52}$$

The challenging part here is evaluating:

$$f(x)\phi^n_{j,l}(x)\big|_{2^{-n}l}^{2^{-n}(l+1)} = f(2^{-n}(l+1))\phi^n_{j,l}(2^{-n}(l+1)) - f(2^{-n}l)\phi^n_{j,l}(2^{-n}l) \tag{2.53}$$

since we do not have the value of $f(x)$ in the points $x = 2^{-n}(l+1)$ and $x = 2^{-n}l$ which represent discontinuities of our basis and is not sampled in our numerical grid. So the values are approximated as:

$$f(2^{-n}(l+1)) \approx 2^{n/2}\sum_{i=0}^{k}\Big[(1-a)s^{n,f}_{i,l}\phi_i(1) + as^{n,f}_{i,l}\phi_i(0)\Big] \tag{2.54}$$

$$f(2^{-n}l) \approx 2^{n/2}\sum_{i=0}^{k}\Big[(1-b)s^{n,f}_{i,l}\phi_i(0) + bs^{n,f}_{i,l}\phi_i(1)\Big] \tag{2.55}$$

where $0 \leq a, b \leq 1$. Selecting the parameters as $a = b = 1/2$ can be seen as a central difference. $a = 1$, $b = 0$ can be viewed as a forward difference, and $a = 0$, $b = 1$ can be viewed as a backward difference. More details on this can be found in [28].

The other method, introduced in [31], employs what we call a B-spline derivative operator. The derivative $\frac{d^p}{dx^p}f^n$ is calculated by applying the operator:

$$D^{(p)} = B^{(p)}(A^*A)^{-1}A^* \tag{2.56}$$

This operator maps the coefficients $s^{n,f}_{i,l}$ onto a b-spline basis, a basis where the derivative is better defined. It then differentiates and transforms the coefficients back to the scaling space. Therefore, the scaling coefficients of $\frac{d^{(p)}f}{dx^{(p)}}$ are calculated as:

$$s^{n,\frac{d^pf}{dx^p}}_{\mathbf{i},l} = D^{(p)}s^{n,f}_{\mathbf{i},l} \tag{2.57}$$

2.8 Convolution Operators

A convolution operation between a kernel K and a function f is defined as follows:

$$g(\mathbf{r}) = (K * f)(\mathbf{r}) = \int K(\mathbf{r}-\mathbf{r}')f(\mathbf{r}')d\mathbf{r}' \tag{2.58}$$

This operation can be written as:

$$g(\mathbf{r}) = \hat{K}[f](\mathbf{r}) \tag{2.59}$$

In this context, we introduce the notation $\hat{K}$ to denote a convolution operator.

In our framework, we work with convolution kernels that are a sum of Gaussians:

$$K(\mathbf{r}-\mathbf{r}') = \sum_{\kappa=1}^{M} \alpha_\kappa e^{-\beta_\kappa \|\mathbf{r}-\mathbf{r}'\|^2} \tag{2.60}$$

$$= \sum_{\kappa=1}^{M} \alpha_\kappa \prod_{i}^{d} e^{-\beta_\kappa \|r_i - r_i'\|^2} \tag{2.61}$$

This results in convolutions of the form:

$$g(\mathbf{r}) = \sum_{\kappa=1}^{M} \sum_{\mathbf{j},\mathbf{m}} s^{n,f}_{\mathbf{j},\mathbf{m}} \prod_{i}^{d} K(x_p - y_p)\phi_{j_p,m_p}(y)dy \tag{2.62}$$

This transformation allows us to convert a single d-dimensional convolution into $M \times d$ 1-dimensional convolutions. Within the multiwavelet framework, this reduces the number of operations required for a convolution from $O((k+1)^{2d})$ to $O(dM(k+1)^{d+1})$. For detailed information on this, refer to [32], where this topic is discussed extensively.

2.8.1 The Poisson Equation

The Poisson equation is traditionally expressed as:

$$\nabla^2 V(\mathbf{r}) = -4\pi\rho(\mathbf{r}) \tag{2.63}$$

In this equation, ∇^2 is the Laplacian operator, $\rho(\mathbf{r})$ traditionally represents the charge density, and $V(\mathbf{r})$ is the electric potential. The solution to this equation can be formulated as a convolution integral involving Poisson kernel $P(\mathbf{r}-\mathbf{r}')$:

$$V(\mathbf{r}) = \int P(\mathbf{r}-\mathbf{r}')\rho(\mathbf{r}')\,\mathrm{d}\mathbf{r}' \tag{2.64}$$

The Poisson kernel, is a Green's function and satisfies the equation

$$\nabla^2 P(\mathbf{r}-\mathbf{r}') = -\delta(\mathbf{r}-\mathbf{r}') \tag{2.65}$$

and it is given by:

$$P(\mathbf{r}-\mathbf{r}') = \frac{1}{4\pi||\mathbf{r}-\mathbf{r}'||} \tag{2.66}$$

We denote the convolution in 2.64 using the operator $\hat{P}$:

$$V(\mathbf{r}) = \hat{P}[\rho](\mathbf{r}) \tag{2.67}$$

2.8.2 The Bound State Helmholtz Equation

The Bound State Helmholtz equation has a central role in quantum chemistry, specifically in the context of the Hartree–Fock equations and the Kohn–Sham equations, detailed further in Sections 3.5.1 and 3.6.1 respectively. The Bound State Helmholtz Equation is given by:

$$\left(\nabla^2 - \mu^2\right) \phi(\mathbf{r}) = s(\mathbf{r}) \tag{2.68}$$

In this equation, μ is a positive number with $\mu > 0$. The Helmholtz equation can be solved through the convolution integral:

$$\phi(\mathbf{r}) = \int G^{\mu}(\mathbf{r} - \mathbf{r}')s(\mathbf{r}')\ \mathrm{d}\mathbf{r}' \tag{2.69}$$

The Helmholtz kernel, denoted as $G^{\mu}(\mathbf{r} - \mathbf{r}')$, satisfies:

$$\left(\nabla^2 - \mu^2\right) G^{\mu}(\mathbf{r} - \mathbf{r}') = -\delta(\mathbf{r} - \mathbf{r}') \tag{2.70}$$

The Helmholtz kernel G is given by:

$$G^{\mu}(\mathbf{r} - \mathbf{r}') = \frac{e^{-\mu||\mathbf{r}-\mathbf{r}'||}}{4\pi||\mathbf{r} - \mathbf{r}'||} \tag{2.71}$$

We denote the convolution in 2.64 using the operator $\hat{P}$:

$$\phi(\mathbf{r}) = \hat{G}^{\mu}[\rho](\mathbf{r}) \tag{2.72}$$

2.8.3 Kernel Separation Using Gaussians

In the beginning of this section, we outlined the use of convolution operators in a separable form, as detailed in equation (2.62). This approach leans toward reducing the number of operations required to apply the operator. However, intrinsic to the Poisson (equation (2.66)) and Helmholtz (equation (2.71)) kernels we examined is a non-separable nature. To handle this, we approximate these kernels as a sum of Gaussians.

The Poisson kernel allows for an alternative representation [33], as demonstrated below:

$$P(r) = \frac{1}{r} = \frac{4}{\sqrt{\pi}} \int_0^{\infty} e^{-4r^2t^2} dt, \tag{2.73}$$

In this instance, the integrand is separable. By adapting from methodologies outlined in [30, 34], the equation can be restructured, ensuring the integrand decays superexponentially with respect to the integration variable:

$$P(r) = \frac{4}{\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{\cosh w}{1 + e^{-\sinh w}} e^{-4r^2(\sinh w + \log(1+e^{-\sinh w}))^2}\ \mathrm{d}w. \tag{2.74}$$

In this configuration, the integrand shows super-exponential decay as $w \to \pm\infty$, thereby facilitating the accurate computation of the integral using the trapezoidal rule.

Similarly, the Bound State Helmholtz Kernel manifests exponential decay when posed as an integral:

$$H^{\mu}(r) = \frac{2}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-r^2 e^{2w} - \frac{\mu^2}{4} e^{-2w} + w} \, \mathrm{d}w. \tag{2.75}$$

This integral is evaluated across a range of values for w, specifically set between w_0 and w_1 to maintain accuracy. The choice of step size follows the suggestion in [9]:

$$h = \frac{1}{0.2 - 0.47 \log_{10} \epsilon}. \tag{2.76}$$

Utilizing Gaussian approximation guarantees the attainment of high precision within the interval $r \in [r_0, r_1]$, where r_1 denotes the maximum distance in the computational domain, and r_0 is chosen to purge singularity contributions. For more insights into the accuracy of this methodology, readers may refer to [9] for insights on the Helmholtz representation and to the supplementary material of [30] for details on the Poisson representation.

Chapter 3

Introduction to Quantum Chemistry

This chapter provides an introduction to quantum chemistry, a field that merges the principles of quantum mechanics and chemistry to understand the behavior of atoms and molecules at the quantum level. Achieving this understanding requires the use of mathematical tools and approximations.

We will first shed light on the necessity of approximations and explain why we do not solve the Schrödinger equation directly. We will then introduce the Born–Oppenheimer approximation, a fundamental concept in molecular quantum mechanics. Further, we will delve into two distinct methods—the Hartree–Fock method and Kohn–Sham Density Functional Theory—exploring their foundational principles and how they fit into a multiwavelet framework.

For clarity, we will express equations in atomic units throughout the chapter. We set the reduced Planck's constant $\hbar = 1$, the elementary charge $q_e = -1$, and the electron mass $m_e = 1$. We also differentiate between $\mathbf{r}$ and $\mathbf{x}$ where the former is the position vector and the latter contain information about both position and spin.

After understanding how to solve the Hartree–Fock and Kohn–Sham equations, we will further explore solvation models—particularly the Polarizable Continuum Model—in the context of a multiwavelet framework. Finally, we will delve into the role and significance of relativity within the multiwavelet framework. As we cover heavier elements where the speed of an electron comes close to the speed of light, relativistic effects become pronounced and need to be accounted for.

3.1 The Schrödinger Equation

The foundation of quantum mechanics is primarily based on the time-dependent Schrödinger equation. The equation is expressed as follows:

$$i\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t), \tag{3.1}$$

where i is the imaginary unit, $\mathbf{r}$ is the position vector, t specifies time, $\Psi(\mathbf{r},t)$ represents the system's wavefunction, and $\hat{H}$ denotes the Hamiltonian operator.

When the Hamiltonian is time-independent—expressed as $H(\mathbf{r},t) = H(\mathbf{r})$—the wavefunction can be expressed as a product of spatial, $\psi(\mathbf{r})$, and temporal components, $\varphi(t)$, such that $\Psi(\mathbf{r},t) = \psi(\mathbf{r})\varphi(t)$. Inserting this into the time-dependent Schrödinger equation (3.1), we can rewrite it as:

$$\frac{\varphi'(t)}{\varphi(t)} = i\frac{\hat{H}\psi(\mathbf{r})}{\psi(\mathbf{r})}, \tag{3.2}$$

where $\varphi'(t)$ represents the time derivative of $\varphi(t)$. In this equation, the left-hand side depends exclusively on time, while the right-hand side relies solely on the spatial coordinates. As such, both sides must individually equal a constant, usually denoted as E:

$$i\frac{\varphi'(t)}{\varphi(t)} = E, \quad \frac{\hat{H}\psi(\mathbf{r})}{\psi(\mathbf{r})} = E. \tag{3.3}$$

The first equation readily permits a solution, yielding the general solution $\varphi(t) = C\exp(-Et)$. After this, our attention shifts to solving the time-independent Schrödinger equation:

$$\hat{H}\psi(\mathbf{r}) = E\psi(\mathbf{r}). \tag{3.4}$$

For a more detailed description of the steps above see for instance Griffiths[35].

3.2 The Time Independent Schrödinger Equation

The time independent Schrödinger equation, represented as:

$$\hat{H}\Psi(\mathbf{r}) = E\Psi(\mathbf{r}), \tag{3.5}$$

is the cornerstone of quantum mechanics. In this equation, $\hat{H}$ denotes the Hamiltonian operator, a Hermitian operator that encapsulates the total energy of a quantum system. The wavefunction, $\Psi(\mathbf{r},\mathbf{R})$, provides a comprehensive description of the system's quantum state as a function of the electron positions ($\mathbf{r}$) and nuclear positions ($\mathbf{R}$), and E represents its associated energy eigenvalue.

For a normalized wavefunction Ψ, the energy eigenvalue E can be determined as the expectation value:

$$E = \langle \Psi | \hat{H} | \Psi \rangle. \tag{3.6}$$

In a broader sense, any measurable property, Ω, of a quantum system can be deduced from its associated operator $\hat{\Omega}$ using:

$$\Omega = \langle \Psi | \hat{\Omega} | \Psi \rangle. \tag{3.7}$$

The Hermitian nature of the Hamiltonian ensures that its solutions (eigenfunctions) can be chosen to be orthogonal and normalized, ensuring the physical realism and stability of these solutions. Furthermore, these solutions form a complete set, which is a crucial property for representing any state of the quantum system.

The full Hamiltonian operator, $\hat{H}$, is a sum of kinetic and potential energy operators:

$$\hat{H} = \hat{T}_e + \hat{T}_n + \hat{V}_{ee} + \hat{V}_{en} + \hat{V}_{nn} \tag{3.8}$$

where

$$\hat{T}_e = -\sum_i \frac{1}{2} \nabla_i^2 \tag{3.9}$$

is the kinetic energy operator for the electrons, and

$$\hat{T}_n = -\sum_I \frac{1}{2M_I} \nabla_I^2 \tag{3.10}$$

is the kinetic energy operator for the nuclei, with M_I representing the mass of the I-th nucleus. The potential energy operators are given by

$$\hat{V}_{en} = \sum_{i,I} \frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} \tag{3.11}$$

which describes the interaction potential between electrons and nuclei, with Z_I representing the atomic number of the I-th nucleus. The electron-electron repulsion potential is given by

$$\hat{V}_{ee} = \sum_{i>j} \frac{1}{|\mathbf{r}_j - \mathbf{r}_i|} \tag{3.12}$$

and the nuclear-nuclear repulsion potential is

$$\hat{V}_{nn} = \sum_{I>J} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} \tag{3.13}$$

In these equations, lowercase letters (i, j) are used to index electrons, while uppercase letters (I, J) denote nuclear particles. The Schrödinger equation (3.5), though linear and simple at first glance, hides inherent complexities. With three times the total number of particles (electrons and nuclei) as degrees of freedom, it can be solved analytically for a maximum of two particles using traditional calculus methods.

The Hamiltonian, $\hat{H}$, given by equation (3.8), incorporates terms associated with kinetic energy (equations (3.9) and (3.10)), and potential energy (equations (3.11) - (3.13)). These potential energy terms describe a variety of inter-particle interactions, including electron-electron, electron-nucleus, and nucleus-nucleus interactions. The number of these inter-particle interactions grows quadratically with the number of particles N, correspondingly increasing the problem's complexity for larger systems.

To manage this computational complexity, incorporating simplifications is crucial. A common initial step involves the application of the Born–Oppenheimer approximation. This approximation drastically simplifies the problem by treating electronic and nuclear motions separately, significantly reducing the complexity of the potential energy terms involving the nuclei in the many-particle Schrödinger equation.

3.3 The Born–Oppenheimer Approximation

The Born–Oppenheimer approximation [36], a common strategy in quantum mechanics, simplifies calculations by taking advantage of the significant mass difference between electrons and nuclei. Given that electrons move considerably faster due to their lower mass, we can consider the nuclei to be almost stationary, thereby reducing the problem to the sole movement of electrons. So, we can formulate potential energy surfaces based on fixed nuclear positions and solve the electronic Schrödinger equation using the electronic Hamiltonian.

Following Atkins[37] we start with the Hamiltonian outlined previously in (3.8), but we combine the potential terms into $\hat{V} = \hat{V}_{ee} + \hat{V}_{en} + \hat{V}_{nn}$:

$$\hat{H} = \hat{T}_e + \hat{T}_n + \hat{V}. \tag{3.14}$$

Under the Born–Oppenheimer approximation, we postulate that the wavefunction Ψ takes the form:

$$\Psi(\mathbf{r}, \mathbf{R}) = \psi(\mathbf{r}; \mathbf{R})\chi(R), \tag{3.15}$$

where ψ is the electronic wavefunction that is now parametrically dependent on the nuclear positions, and χ is the nuclear wavefunction. Substituting this into the Schrödinger equation (3.5), we obtain:

$$\begin{align} H\psi\chi &= \chi T_e\psi + \psi T_n\chi + \chi T_n\psi + V\psi\chi \tag{3.16} \\ &\approx \chi T_e\psi + \psi T_n\chi + V\psi\chi. \tag{3.17} \end{align}$$

Since $\hat{T}_n$ also needs to be applied to ψ due to its parametric dependency on the nuclear positions R, we note that as the nuclear mass M_J appears in the denominator of $\hat{T}_n$ (see equation (3.10)), this term tends to be small and can be neglected.

Rearranging the terms in equation (3.16), we obtain:

$$\psi T_N \chi + (T_e \psi + V \psi)\chi = E\psi\chi. \tag{3.18}$$

We can now introduce the electronic Hamiltonian $\hat{H}_{el} = T_e + V$ and the clamped Schrödinger equation [37]:

$$\hat{H}_{el}\psi = E_e(\mathbf{R})\psi. \tag{3.19}$$

By inserting this into equation (3.18) and canceling the electronic wavefunction ψ, we get:

$$T_n \chi(\mathbf{R}) + E_e(\mathbf{R})\chi(\mathbf{R}) = E\chi(\mathbf{R}). \tag{3.20}$$

The eigenvalue E represents the total energy of the system under the Born–Oppenheimer approximation. However, in most cases, the primary equation to work with is the clamped nuclei Schrödinger equation (3.19). This approach significantly simplifies the problem, while still offering a comprehensive depiction of a molecular system's quantum behavior. From this point forward, we will encounter the electronic Hamiltonian in our discussions. To simplify notation, we will refer to the electronic Hamiltonian, $\hat{H}_{el}$, as simply $\hat{H}$.

3.4 The Variational Principle

The Variational Principle is a fundamental theorem in quantum chemistry. It states that the energy of an approximate wavefunction is always greater than or equal to the exact energy of the system. This principle is often used to estimate the ground state of a system, denoted as Ψ_0, which is characterized by having the lowest energy E_0.

The Variational Principle states that for a given Hamiltonian $\hat{H}$ with a true ground state Ψ_0, any arbitrary trial wavefunction $\tilde{\Psi}$ will satisfy the following inequality (see for instance[38]):

$$\frac{\langle\tilde{\Psi}|\hat{H}|\tilde{\Psi}\rangle}{\langle\tilde{\Psi}|\tilde{\Psi}\rangle} \geq \frac{\langle\Psi_0|\hat{H}|\Psi_0\rangle}{\langle\Psi_0|\Psi_0\rangle} \tag{3.21}$$

This implies that the problem of finding the ground state can be treated as a minimization problem, where the trial wavefunction is varied until the corresponding energy is minimized. This approach often provides a good approximation to the true ground state, especially if the form of $\tilde{\Psi}$ is chosen to reflect the physical characteristics of the system.

3.5 Hartree–Fock

The Hartree–Fock method is a fundamental approach in quantum chemistry. Its aim is to approximate the ground state of the wavefunction using a *single* Slater determinant. The Slater determinant ensures that the many-electron wavefunction Ψ is anti-symmetric, in accordance with the Pauli exclusion principle[39]. The mathematical representation of the Slater Determinant is:

$$\Phi = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_1(\mathbf{x}_1) & \phi_1(\mathbf{x}_2) & \cdots & \phi_1(\mathbf{x}_N) \\ \phi_2(\mathbf{x}_1) & \phi_2(\mathbf{x}_2) & \cdots & \phi_2(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\mathbf{x}_1) & \phi_N(\mathbf{x}_2) & \cdots & \phi_N(\mathbf{x}_N) \end{vmatrix} \tag{3.22}$$

The Hartree–Fock energy is given by:

$$E_{HF} = \langle \Phi | \hat{H} | \Phi \rangle, \tag{3.23}$$

To make this more explicit, the energy expression in terms of the spin-orbitals $\{\phi_i, i = 1 \dots N\}$ that define the Slater determinant can be written as:

$$\langle \Phi | \hat{H} | \Phi \rangle = \sum_i h_i + \frac{1}{2} \sum_{ij} (J_{ij} - K_{ij}) + U_N, \tag{3.24}$$

The one-electron term h_i arises from the kinetic and potential energy of a single electron. Specifically, it includes the kinetic energy operator $\hat{T} = -\frac{1}{2}\nabla^2$ and the electron-nuclear potential:

$$\hat{V}_n = -\sum_I \frac{Z_I}{\|\mathbf{r} - \mathbf{R}_I\|}. \tag{3.25}$$

Here, $\mathbf{r}$ denotes the position of an electron, $\mathbf{R}_I$ is the position of the I-th nucleus, Z_I is the charge of the I-th nucleus, and ∇^2 is the Laplacian operator.

$$h_i = \langle \phi_i | \hat{h} | \phi_i \rangle = \langle \phi_i | \hat{T} + \hat{V}_N | \phi_i \rangle \tag{3.26}$$

The two-electron terms J_{ij} and K_{ij}, namely the Coulomb interaction and exchange interaction, arise from the electron-electron interactions. The Coulomb term encapsulates the classical electrostatic repulsion between electrons. It quantifies the cost of placing two electrons at positions $\mathbf{r}$ and $\mathbf{r}'$.

$$J_{ij} = \langle \phi_i(x_1)\phi_j(x_2) | 1/r_{12} | \phi_i(x_1)\phi_j(x_2) \rangle \tag{3.27}$$

In contrast to the Coulomb term, the exchange term does not have a classical counterpart. Its existence is a consequence of the Pauli exclusion principle, which mandates the anti-symmetry of the electron wavefunction. This term is a fundamental aspect of quantum mechanics, reflecting the indistinguishability and anti-symmetry of electrons. The exchange term is given by:

$$K_{ij} = \langle \phi_i(x_1)\phi_j(x_2) | 1/r_{12} | \phi_j(x_1)\phi_i(x_2) \rangle \tag{3.28}$$

3.5.1 The Hartree–Fock Equations

To find the optimal Φ, we use the Lagrangian multiplier technique(see for instance[40]), which ensures the orthogonality of the orbitals. The conditions for energy minimization arise when we set the functional derivative of the Lagrangian with respect to the orbital variations to zero:

$$\delta L = \sum_i \langle \delta\phi_i|\hat{F}|\phi_i\rangle - \sum_{ij} \lambda_{ij} \langle \delta\phi_i|\phi_j\rangle + \sum_i \langle \delta\phi_i|\hat{F}|\phi_i\rangle^* - \sum_{ij} \lambda_{ji} \langle \delta\phi_i|\phi_j\rangle^* = 0, \tag{3.29}$$

A key component of the Hartree–Fock method is the Fock operator, $\hat{F}$, which combines the effects of the one-electron operator and the Coulomb and exchange interactions:

$$\hat{F} = \hat{h} + \sum_j \hat{J}_j - \hat{K}_j. \tag{3.30}$$

For clarity, we adopt an operator notation for the Coulomb and Exchange operators:

$$\hat{J}_i|\phi_j\rangle = \langle\phi_i|1/r_{12}|\phi_i\rangle|\phi_j\rangle \tag{3.31}$$

$$\hat{K}_i|\phi_j\rangle = \langle\phi_i|1/r_{12}|\phi_j\rangle|\phi_i\rangle \tag{3.32}$$

Differentiating the Lagrangian expression and setting the derivative equal to zero results in the coupled Hartree–Fock equations:

$$\hat{F}\Phi = \Phi\mathbf{F} \tag{3.33}$$

Here, each element F_{ij} of the Fock matrix, $\mathbf{F}$, is equivalent to the Lagrange multiplier λ_{ij}. The Fock matrix arises from the stationary condition on the Lagrangian and can be formally defined by projecting the Hartree–Fock equations onto the set of occupied orbitals:

$$\mathbf{F} = \langle\Phi|\hat{F}|\Phi\rangle. \tag{3.34}$$

3.6 Density Functional Theory

Density Functional Theory (DFT) offers an alternative approach to solving the Schrödinger equation. Based on the work of Hohenberg and Kohn [41], DFT demonstrates that the electron density, $\rho(\mathbf{r}_1)$, defined as:

$$\rho(\mathbf{r}_1) = N \int |\phi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)|^2 ds_1 d\mathbf{x}_2 \cdots d\mathbf{x}_N, \tag{3.35}$$

provides the same information as the wavefunction. The total energy of the system can then be expressed as a functional of ρ:

$$E[\rho] = T[\rho] + V_{ne}[\rho] + V_{ee}[\rho], \tag{3.36}$$

where $T[\rho]$ is the kinetic energy, $V_{ne}[\rho]$ the electron-nuclear interaction energy, and $V_{ee}[\rho]$ the electron-electron interaction energy functionals. The ground state density is the density that minimizes this energy:

$$E_0 = \min \rho \ E[\rho]. \tag{3.37}$$

In the Born–Oppenheimer approximation, the electron-nuclear interaction energy can be calculated as the classical electrostatic interaction between charge densities:

$$V_{ne}[\rho] = \int \rho(\mathbf{r}) V_n(\mathbf{r}) d\mathbf{r}, \tag{3.38}$$

with $V_n(\mathbf{r})$ given by equation (3.25).

The main challenge in DFT lies in the fact that the kinetic T and electron-electron V_{ee} interaction energy functionals in (3.36) are unknown. These energies stem from quantum mechanical interactions, but their functional forms in terms of ρ are not straightforward. This complexity arises from the quantum mechanical exchange and correlation energies, in addition to the classical electrostatic interactions. The development of accurate approximations for these functionals, either through theoretical frameworks or semi-empirical strategies informed by experimental data, represents a key aspect of DFT.

In the following sections, we will discuss one strategy for overcoming this challenge, namely, Kohn–Sham DFT.

3.6.1 The Kohn–Sham Equations

The derivation of the Kohn–Sham equations presented in this section follows the approach outlined in [42]. As previously mentioned, DFT offers an alternative method for solving the Schrödinger Equation to determine a quantum system. However, the exact forms of both the kinetic energy functional $T[\rho]$ and the electron-electron interaction energy functional $V_{ee}[\rho]$ remain unknown. This presents a challenge in approximating these functionals. Notably, the virial theorem suggests that the kinetic energy is approximately equal to the system's total energy, emphasizing the importance of its accurate representation.

An alternative approach to approximating the kinetic energy and electron-electron interaction energy functionals was proposed by Kohn and Sham [43]. They began by introducing density orbitals, ϕ_i, representing the density using one-particle functions. For simplicity, we assume double occupancy in the orbitals. For such a closed-shell system, this results in:

$$\rho(\mathbf{r}) = 2 \sum_{i}^{N/2} |\phi_i(\mathbf{r})|^2, \tag{3.39}$$

This approach reintroduces the concept of orbitals from Hartree–Fock theory. The key idea is that the kinetic energy of a set of non-interacting orbitals is well-defined:

$$T_s[\rho] = 2\sum_{i}^{N/2} \langle \phi_i | -\frac{1}{2}\nabla^2 | \phi_i \rangle. \tag{3.40}$$

However, it is important to note that this is not equal to the true kinetic energy functional, leading to the discrepancy: $T[\rho] - T_s[\rho]$. Following this approach, we separate the classical component J:

$$J[\rho] = \frac{1}{2}\int\int \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{\|\mathbf{r}-\mathbf{r}'\|} d\mathbf{r}d\mathbf{r}', \tag{3.41}$$

from the energy expression V_{ee} noting the energy discrepancy: $V_{ee}[\rho] - J[\rho]$. These differences are then collected in an exchange-correlation functional[1]:

$$E_{xc}[\rho] = T[\rho] - T_s[\rho] + V_{ee}[\rho] - J[\rho], \tag{3.42}$$

resulting in the Kohn–Sham energy:

$$E[\rho] = T_s[\rho] + V_{en}[\rho] + J[\rho] + E_{xc}[\rho]. \tag{3.43}$$

Minimizing the energy with respect to the density yields the Euler equation:

$$\mu = \frac{\delta T_s[\rho]}{\delta \rho(\mathbf{r})} + v_{eff}(\mathbf{r}), \tag{3.44}$$

where μ is a Lagrange multiplier ensuring electron number conservation. The effective potential, expressed in terms of functional derivatives, is:

$$v_{eff}(\mathbf{r}) = \frac{\delta V_{en}[\rho]}{\delta \rho(\mathbf{r})} + \frac{\delta J[\rho]}{\delta \rho(\mathbf{r})} + \frac{\delta E_{xc}[\rho]}{\delta \rho(\mathbf{r})}, \tag{3.45}$$

$$= v_{nuc}(\mathbf{r}) + v_{el}(\mathbf{r}) + v_{xc}(\mathbf{r}). \tag{3.46}$$

The Euler equation can be interpreted as a system of non-interacting electrons moving within an effective potential v_{eff}. Its associated Hamiltonian is:

$$\hat{H} = -\sum_{i}^{N/2} \frac{1}{2}\nabla_i^2 + \sum_{i}^{N/2} v_{eff}(\mathbf{r}_i), \tag{3.47}$$

from which the Fock or Kohn–Sham operator is derived:

$$\hat{F} = -\frac{1}{2}\nabla^2 + v_{eff}(\mathbf{r}). \tag{3.48}$$

Solving DFT's Euler equation leads to the Kohn–Sham equations:

$$\hat{F}\Phi = \Phi\mathbf{F} \tag{3.49}$$

[1]The exact form of this functional is unkown. Coutless parametrisations are available. Derived either from empirical or theoretical assumtions.

While the reintroduction of orbitals (equation (3.39)) means forgoing a single three-dimensional equation for the problem, we now encounter $N/2$ coupled non-linear equations for the orbitals. Since the effective potential in the Kohn–Sham operator is density-dependent (and thus orbital-dependent), Kohn–Sham DFT emerges as a self-consistent field (SCF) method. Given its parallels with the Hartree–Fock equations, similar techniques can be employed to address both. Importantly, Kohn–Sham DFT is computationally similar to Hartree–Fock, yet it often yields significantly better results. However, it is worth noting that there is no systematic way to improve the results towards the exact solution in Kohn–Sham DFT, in contrast with wavefunction theory.

3.7 Solving the Hartree–Fock and Kohn–Sham Equation in Integral Form

Both the Hartree–Fock (3.33) and Kohn–Sham (3.49) equations share a common structure:

$$\hat{F}\Phi = \Phi\mathbf{F} \tag{3.50}$$

For the Hartree–Fock equation, the Fock operator, $\hat{F}$, is defined as:

$$\hat{F} = -\frac{1}{2}\nabla^2 + V_{eff}, \tag{3.51}$$

with the effective potential, V_{eff}, defined as:

$$V_{eff} = \hat{J} - \hat{K} + V_n. \tag{3.52}$$

In the Kohn–Sham equation, V_{eff} is defined as provided in equation (3.46).

The goal is to solve these equations iteratively using the Helmholtz operator (section 2.8.2). The chosen strategy for this includes subtracting the diagonal matrix Λ, where $\Lambda_{ij} = \lambda_i\delta_{ij}$ with $\lambda_i < 0$, from both sides of equation (3.50):

$$(\hat{F} - \Lambda)\Phi = \Phi(\mathbf{F} - \Lambda). \tag{3.53}$$

This transformation facilitates the recasting of the equation as follows:

$$(-\frac{1}{2}\nabla^2 + V_{eff} - \Lambda)\Phi = \Phi(\mathbf{F} - \Lambda) \rightarrow \tag{3.54}$$

$$\left(\nabla^2 + 2\Lambda\right)\Phi = 2\left[V\Phi + \Phi\left(\Lambda - \mathbf{F}\right)\right]. \tag{3.55}$$

By choosing $\mu_i = \sqrt{-2\lambda_i}$, the equation can be inverted using the Helmholtz operator, as discussed in Section 2.8.2:

$$\Phi = -2\hat{G}^{\mu}\left[V\Phi + \Phi\left(\Lambda - \mathbf{F}\right)\right]. \tag{3.56}$$

Given that Φ appears on both sides of the equation, the solution necessitates an iterative approach until self-consistency is achieved:

$$\tilde{\Phi}^{n+1} = -2\hat{G}^{\mu^n}\left[V\Phi^n + \Phi^n\left(\Lambda - \mathbf{F}^n\right)\right]. \tag{3.57}$$

Managing self-consistency can pose computational challenges, particularly for systems involving a large number of electrons, where orbital rotations may lead to slow convergence. To overcome such hurdles, acceleration techniques like KAIN (Krylov subspace Accelerated Inexact Newton method) [44] are used. KAIN employs information from preceding iterations to accelerate convergence, proving particularly useful in overcoming sluggish orbital rotations. It's important to note that while this method increases the memory footprint of the iteration, it also enhances convergence. KAIN serves a similar role in convergence acceleration to other techniques such as DIIS [45] and Anderson mixing [46].

The selection of λ_i is not a random choice. According to numerical studies, opting for λ_i as the diagonal elements of $\mathbf{F}$, namely $\lambda_i = F_{ii}$, typically delivers optimal convergence (see [47]). Additionally, it should be emphasized that a λ_i value that significantly deviates from the diagonal elements in $\mathbf{F}$ can hinder convergence, reinforcing the importance of a well chosen λ_i value.

The integral operator $\hat{G}^{\mu}$ does not conserve the orthogonality of all orbitals ϕ_i within Φ. To denote the set of vectors needing orthogonality enforcement, we use the tilde notation $\tilde{\Phi}^{n+1}$. Although the Gram-Schmidt procedure is a method for enforcing orthogonality (see, for instance [48]), it is essential to avoid this projective approach as it prevents the carry-over of the Fock matrix to new orbitals [49]. Instead, we use a Löwdin transformation [50] where orthogonality is assured through the overlap matrix:

$$S = \langle \Phi | \Phi \rangle \tag{3.58}$$

$$\Phi = \tilde{\Phi} S^{-1/2}. \tag{3.59}$$

To accelerate convergence and conserve memory, we augment the Löwdin transformation with another rotation M that adjusts the orbitals to a specific form. For small systems0, this could involve diagnonalizing the Fock matrix, but for larger systems, it is often beneficial to localize the orbitals, for instance, through a Foster–Boys transformation [50, 51].In practice, it is not always necessary to diagonalize or localize the Fock matrix during every SCF iteration. Therefore, the matrix M can often be chosen as the identity for many intermediate steps. The combined transformation matrix then becomes $U = S^{-1/2} M_X$, where $X = \{C, L, I\}$ represents the canonical, localized, or identity forms, respectively. The new orbital vector and Fock matrix are then obtained by:

$$\Phi = \tilde{\Phi} U, \tag{3.60}$$

$$\mathbf{F} = U^{\dagger} \mathbf{F} U. \tag{3.61}$$

This approach is further detailed in [49].

3.8 Solvation

Understanding the interactions between a quantum system—commonly a molecular solute—and its surrounding environment or solvent is the central objective of solvation studies in quantum chemistry for the last half-century[52–55]. However, the task of accurately representing these interactions while maintaining reasonable computational costs presents a significant challenge. There are two primary familiers of models developed to deal with this problem.

The first is the explicit models. Here, the environment is rendered in detail using either a cheap quantum mechanical approach[56] or molecular mechanics[57] commonly known as quantum mechanics/molecular mechanics (QM/MM).These models can be polarizable or non-polarizable[58–60], indicating the mutual polarization between the solute and solvent or implying a fixed environment. Defining an optimal cut-off radius for long-range interactions, like electrostatic and van der Waals forces, without significantly increasing computational costs. Some might argue that, with appropriate methodologies, QM/MM computations can be as cost-effective as continuum models. However, this issue is complex. QM/MM methods usually require hands-on adjustments and modifications, such as obtaining varied structures from molecular dynamics and performing necessary averaging. Moreover, assessing the reliability of the classical force field in these models can be complicated, adding another layer of intricacy.

The second family of methods pertains to those where the interaction with the solution is implicitly dealt with. In this implicit solvation approach, the solute is placed within a *cavity* for a quantum mechanical analysis, distinctly separating it from the rest of the system. Traditionally, a sharp boundary exists between the solute in the cavity and the solvent or environment, with the interaction between the solute and solvent treated as a pure boundary problem. However, this hard cutoff is not physically sound as the electronic densities of the solvent and solute overlap. This issue has subsequently been addressed by a normalization procedure[54]. Later, more elaborate methods were developed[61] and for the Integral Equation Formalism formulation of the Polarizable Continuum Model (PCM), it has been shown that the model includes first-order corrections[62].

With the development of real-space methods for quantum chemistry[3–6, 14, 63], it is now possible to employ a soft cavity boundary. Here, the potential is not calculated in a vacuum, but instead within a generalized dielectric medium with position-dependent permittivity. This approach has been followed in [22, 24], as well as several real space codes[64–69].

3.8.1 The Solvation Hamiltonian

Within the framework of the Born–Oppenheimer approximation (see section 3.3), we define the Hamiltonian as outlined by Tomasi[53, 54]:

$$H(\mathbf{r}_S, \mathbf{r}_E) = H_S(\mathbf{r}_S) + H_E(\mathbf{r}_E) + H_{SE}(\mathbf{r}_S, \mathbf{r}_E) \tag{3.62}$$

The term H_S corresponds to the electronic Hamiltonian (refer to equation (3.19) for details). H_E denotes the Hamiltonian for the solvent. The term H_{SE} represents the interaction energy between the solute and the environment. This includes energy contributions from electrostatics and polarization, creating the cavity, dispersion energy, repulsion energy, the last two being components of van der Waals forces. Here, $\mathbf{r}_S$ and $\mathbf{r}_E$ encapsulate coordinates of both electrons and nuclei within the solute and solvent, respectively.

It is not important to get a good description of the solvent. It's sufficient to have a good description of the interaction. The Hamiltionian is then reduced to the form:

$$H(\mathbf{r}_S, \mathbf{r}_E) = H_S(\mathbf{r}_S) + H_{SE}(\mathbf{r}_S, \mathbf{r}_E) \tag{3.63}$$

However, as discussed by Tomasi[53], the remaining degrees of freedom in the Hamiltonian still prove computationally hefty. Addressing this challenge, we introduce a reaction potential V_R:

$$H(\mathbf{r}_S, \mathbf{r}_E) = H_S(\mathbf{r}_S) + V_R \tag{3.64}$$

This reaction potential aims to accurately represent the interaction between electrons and nuclei within the solute and solvent. In the following sections, we delve into how the reaction potential V_R is derived starting from the Generalized Poisson Equation.

3.8.2 The Generalized Poisson Equation

The Generalized Poisson Equation (GPE)[2] is expressed as:

$$\nabla \cdot [\epsilon(\mathbf{r}) \nabla V(\mathbf{r})] = -4\pi\rho(\mathbf{r}). \tag{3.65}$$

Here, $\epsilon(\mathbf{r})$ represents the isotropic permittivity of the medium, and ρ is the charge density, which contain both electronic and nuclear contributions:

$$\rho(\mathbf{r}) = \rho_{el}(\mathbf{r}) + \rho_{nuc}(\mathbf{r}), \tag{3.66}$$

where ρ_{nuc}, the nuclear density, is defined as:

$$\rho_{nuc}(\mathbf{r}) = \sum_{I}^{N} Z_I \delta(\mathbf{r} - \mathbf{R}_I). \tag{3.67}$$

The formulation currently provided for the GPE doesn't align with the tools currently available to us. Thus, to solve this equation using the Poisson operator, as introduced in section 2.8.1, we will need to rewrite it in a suitable form. To do that we need to isolate the Laplacian therm $\nabla^2 V$

By applying the vector identity $\nabla \cdot (f\mathbf{A}) = f\nabla \cdot \mathbf{A} + \mathbf{A} \cdot \nabla f$, we can rewrite the GPE as:

$$\nabla^2 V(\mathbf{r}) = -4\pi \frac{\rho(\mathbf{r})}{\epsilon(\mathbf{r})} - \frac{\nabla\epsilon(\mathbf{r}) \cdot \nabla V(\mathbf{r})}{\epsilon(\mathbf{r})}. \tag{3.68}$$

[2]A derivation of the GPE can be found in [24]

We then introduce the surface charge distribution due to polarization at the solute-solvent boundary, denoted by γ:

$$\gamma(\mathbf{r}) = \frac{\nabla\epsilon(\mathbf{r}) \cdot \nabla V(\mathbf{r})}{4\pi\epsilon(\mathbf{r})}. \tag{3.69}$$

Defining the effective charge density ρ_{eff} as ρ/ϵ, the Generalized Poisson equation takes the form:

$$\nabla^2 V = -4\pi(\rho_{eff} + \gamma). \tag{3.70}$$

Given that the Laplacian term $\nabla^2 V$ is now isolated, this equation must be solved self-consistently. This requirement emerges due to γ containing the potential V, thereby necessitating a self-consistent solution approach. Our ultimate goal is to calculate the reaction potential V_R. To achieve this, we partition the total potential V into the electrostatic potential in a vacuum, denoted as V_{vac}, and the reaction potential V_R:

$$V = V_{vac} + V_R \tag{3.71}$$

Remembering that $\nabla^2 V_{nuc} = -4\pi\rho$ equation (3.70) takes the form:

$$\nabla^2 V_R = -4\pi\left(\rho_{eff} + \gamma - \rho\right) \tag{3.72}$$

With this we can invert it using the Poisson operator

$$V_R = \hat{P}\left[\rho_{eff} + \gamma - \rho\right] \tag{3.73}$$

Now, since γ is a function of the reaction potential V_R, it must be solved iteratively until self-consistency is achieved. In the following sections, we'll discuss how the solution integrates into solving the Kohn–Sham equation or Hartree–Fock equations with the reaction potential.

3.8.3 Integrating and Solving the Reaction Potential in Hartree–Fock and Kohn–Sham Equations

Gerez et al. [24] provide a detailed scheme for solving the self-consistent reaction field (SCRF), along with the associated algorithms. This section outlines their method and illustrates how it fits seamlessly into the self-interaction scheme presented in section 3.7. The main deviation of this method from the conventional approach resides in the composition of the Fock operator:

$$\hat{F}_{sol} = \hat{F} + V_R, \tag{3.74}$$

where V_R is the reaction potential that must be solved for self-consistently between each iteration:

$$V_R^{n,i} = \hat{P}\left[\rho_{eff}^n + \gamma^i - \rho\right]. \tag{3.75}$$

Here, i denotes the microiterations.

This iterative process is crucial from a physical perspective as it aptly portrays mutual polarization between the solvent and the solute. The orbitals, representing the quantum system, are subject to change, influencing their interaction with the potential. In turn, this potential also interacts with the orbitals, thus creating a dynamic interplay. This cycle continuously refines our understanding of the system, accounting for alterations in the solute's electronic configuration and its interaction with the solvent.

According to Gerez et al. [24], optimal convergence is achieved when the converged reaction potential is used as the starting guess for each new microiteration scheme, i.e., $V^{n,0} = V^{n-1}$. It is also beneficial to employ a less strict convergence criterion for the microiterations than for the main iterations. Furthermore, the micro SCF can be accelerated to enhance convergence, thereby improving the overall efficiency of the calculations.

3.9 Relativity in Multiwavelets

The concept of relativity within the context of multiwavelets in quantum chemistry is crucial due to the potential for achieving higher *accuracy* in calculating molecular properties, especially those involving heavier atoms[70]. To this end, significant work has been undertaken by the `MADNESS` group. They construct arbitrariness-precise representations necessary for ut pt second order Douglass–Kroll–Hess[71], thus presenting a model for constructing quasi-relativistic Hamiltonians. Furthermore, and importantly, they report the first fully numerical method to quantum chemical calculations applicable to molecules[72]. This constitutes a complete implementation of the four-component Dirac–Coulomb equation formulated to a user-specified precision.

Notwithstanding advancements made in adjusting existing basis-set methods to account for the effects of special relativity, these adaptations have limitations similar to nonrelativistic basis-set calculations. For instance, they offer poor scaling with respect to the number of basis functions and exhibit a high number of linear dependencies when working with large basis sets[2]. In addition, there is the requirement to select and contract basis sets for relativistic calculations with added caution. This additional care maintains the appropriate relation between the large and small components, thus preventing variational collapse[73]. Fully numerical codes are available, but their application is usually relegated to atoms[74] or small molecules with pronounced symmetry[75–77].

Moreover, the `MRChem` group sets forth by working on the four-component Hamiltonian. In[22], an implementation of the Dirac equation for one electron is demonstrated. Meanwhile, [25] details the implementation of the Full Breit Hamiltonian, an important progression towards incorporating magnetic interactions including spin-other-orbit and the retardation effect due to the finite speed of light[78–81]. Notably, for the sake of realistic core-electron spectroscopies, it is imperative to include the Breit interaction terms[82, 83]. This has been actualized in a practical implementation carried out in `VAMPyR`. In conclusion, this section aims to introduce the Dirac equation that governs the relativistic space and to explore its integral form. Moreover, strategies for solving this will be examined, influenced by similar methodologies used for the Hartree–Fock and Kohn–Sham equations in section 3.7.

In the following two sections we give an introduction to the Dirac equation in its 4-component form. We then present the Green's kernel that can be used to solve it in a multiwavelet framework.

3.9.1 The Dirac Equation

The Dirac equation is a relativistic equation that is consistent with both the principles of quantum mechanics and the theory of special relativity. The equation is characterized by a 4-component wavefunction, which is necessary to form a complete Hamiltonian. In this section, we will start with the relativistic energy equation and investigate the necessity of a 4-component wavefunction. Instead of following Dirac's derivation[84, 85], we adopt the approach of van Waerden[86] and Saue[87].

The derivation of the Dirac equation begins with the relativistic energy equation

$$E^2 = m^2c^4 + p^2c^2 \tag{3.76}$$

where m is the mass, c is the speed of light, and p is the linear momentum. When we take the square root, we encounter a choice of sign:

$$E = \pm\sqrt{p^2c^2 + m^2c^4} \tag{3.77}$$

This implies that possible energy values exist in two bands of opposite sign, separated by an energy gap of $2mc^2$. In classical mechanics, we disregard the negative energy band as unphysical because energy can only change continuously. However, in quantum mechanics, discrete energy jumps are possible, making the negative energy band relevant.

The relativistic energy equation (3.76) does not resemble its non-relativistic counterpart:

$$E = \frac{p^2}{2m} \tag{3.78}$$

However, they are connected, which can be seen by Taylor expanding the positive branch of the energy $E = \sqrt{p^2c^2 + m^2c^4}$:

$$E = mc^2\sqrt{1 + \frac{p^2}{m^2c^2}} = mc^2 + \frac{p^2}{2m} - \frac{p^4}{8m^3c^2} + ... \tag{3.79}$$

The first term, mc^2, is known as the rest mass, the second term, $p^2/2m$, is the non-relativistic energy (see equation (3.78)), and the remaining terms are relativistic corrections.

To approximate the Hamiltonian, one approach could be to start with the Taylor expansion of the relativistic energy in equation (3.76) and quantize the components. However, this yields an infinite sum as the Hamiltonian. The challenge lies in representing this infinite sum in a more manageable form. Quantization in quantum mechanics involves replacing classical observable with quantum operators. This is done using the following substitutions:

$$\mathbf{x} \to \hat{\mathbf{x}} \qquad \mathbf{p} \to \hat{\mathbf{p}} = -i\frac{d}{d\mathbf{x}} \tag{3.80}$$

and the heuristic substitution

$$E \to i\frac{\partial}{\partial t} \tag{3.81}$$

Applying these substitutions to the non-relativistic energy equation (3.78) leads us to the time-dependent Schrödinger equation (3.1). However, it is important to note that the Schrödinger equation is non-relativistic and does not account for spin, which are key considerations in the context of the Dirac equation. To incorporate spin and relativistic effects, we turn to the Dirac identity:

$$(\sigma \cdot \mathbf{A})(\sigma \cdot \mathbf{B}) = \mathbf{A} \cdot \mathbf{B} + i\sigma \cdot (\mathbf{A} \times \mathbf{B}) \tag{3.82}$$

In the Dirac identity, σ represents the three Pauli spin matrices:

$$\sigma = (\sigma_1, \sigma_2, \sigma_3) \tag{3.83}$$

Each component is defined as:

$$\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.84}$$

The Pauli spin matrices, σ_1, σ_2, and σ_3, where constructed by Pauli[39], to take into account the interaction of spin of a particle with an external electromagnetic field.

Inserting the linear momentum operator $\mathbf{p}$ for $\mathbf{A}$ and $\mathbf{B}$ into equation (3.82) we get

$$(\sigma \cdot \mathbf{p})(\sigma \cdot \mathbf{p}) = p^2 \tag{3.85}$$

This suggests that spin may be "hidden" in the non-relativistic wave equation and only appear when an external magnetic field is introduced. To obtain the Dirac equation, we start by quantizing the relativistic energy equation (3.76) using the substitutions for energy (3.81) and momentum (3.80). This yields:

$$\left(-\frac{1}{c^2}\frac{\partial^2}{\partial t^2} - p^2\right)\phi_1 = (mc)^2\phi_1 \tag{3.86}$$

The left-hand side can be expanded using the Dirac identity (3.82):

$$\left[\frac{i}{c}\frac{\partial}{\partial t} + (\sigma \cdot \mathbf{p})\right]\left[\frac{i}{c}\frac{\partial}{\partial t} - (\sigma \cdot \mathbf{p})\right]\phi_1 = (mc)^2\phi_1 \tag{3.87}$$

This form with $(\sigma \cdot \mathbf{p})$ implies that the wavefunction ϕ_1 is two-component, as the Pauli matrices σ are 2x2 matrices.

If we introduce a second wavefunction ϕ_2

$$\phi_2 = \left[\frac{i}{c}\frac{\partial}{\partial t} - (\sigma \cdot \mathbf{p})\right]\phi_1/(mc) \tag{3.88}$$

we obtain two coupled equations

$$\left[\frac{i}{c}\frac{\partial}{\partial t} - (\sigma \cdot \mathbf{p})\right]\phi_1 = mc\phi_2 \tag{3.89}$$

$$\left[\frac{i}{c}\frac{\partial}{\partial t} + (\sigma \cdot \mathbf{p})\right]\phi_2 = mc\phi_1 \tag{3.90}$$

Then taking the linear combinations of these equations, adding and subtracting them, we get

$$\frac{i}{c}\frac{\partial}{\partial t}\Psi^L - (\sigma \cdot \mathbf{p})\Psi^S = mc\Psi^L \tag{3.91}$$

$$-\frac{i}{c}\frac{\partial}{\partial t}\Psi^S + (\sigma \cdot \mathbf{p})\Psi^L = mc\Psi^S \tag{3.92}$$

Here we have introduced the notation for large Ψ^L and small component Ψ^S of the wavefunction:

$$\Psi^L = [\phi_1 + \phi_2] \tag{3.93}$$

$$\Psi^S = [\phi_1 - \phi_2] \tag{3.94}$$

The large component wavefunction, Ψ^L, corresponds to the wavefunction previously discovered by Pauli[39]. This wavefunction describes spin-1/2 particles, like electrons, in non-relativistic quantum mechanics. The small component wavefunction, Ψ^S, on the other hand, is a unique feature of the Dirac equation and accounts for the relativistic effects or "corrections". The resulting equations can then be written in a matrix form

$$\begin{bmatrix} \frac{i}{c}\frac{\partial}{\partial t} & -(\sigma \cdot \mathbf{p}) \\ (\sigma \cdot \mathbf{p}) & -\frac{i}{c}\frac{\partial}{\partial t} \end{bmatrix}\begin{bmatrix}\Psi^L \\ \Psi^S\end{bmatrix} = mc\begin{bmatrix}\Psi^L \\ \Psi^S\end{bmatrix} \tag{3.95}$$

or in its conventional form

$$(\beta mc^2 + c(\alpha \cdot \mathbf{p}))\Psi = i\frac{\partial}{\partial t}\Psi \tag{3.96}$$

Here, Ψ is the 4-component wavefunction:

$$\Psi = \begin{bmatrix} \Psi^L \\ \Psi^S \end{bmatrix} \tag{3.97}$$

The Dirac matrices in equation (3.96), $\boldsymbol{\alpha}$ and β, play a crucial role in the Dirac equation. They are defined as follows:

$$\beta = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & -I \end{bmatrix}, \quad \boldsymbol{\alpha} = \left(\begin{bmatrix} \mathbf{0} & \sigma_1 \\ \sigma_1 & \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{0} & \sigma_2 \\ \sigma_2 & \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{0} & \sigma_3 \\ \sigma_3 & \mathbf{0} \end{bmatrix} \right) \tag{3.98}$$

where σ_1, σ_2, and σ_3 are the Pauli matrices as defined in equation (3.84). These matrices were introduced by Dirac as a way to take the square root of the Hamiltonian, which is derived from the relativistic energy equation (3.76). This approach resulted in the first derivation of the Dirac equation (3.96).

4-component Relativistic Bound-State Green's Kernel

The strategy used to solve the Kohn–Sham and Hartree–Fock equations in section 3.7 is to represent them in their integral form, after which they are solved iteratively until self-consistency is achieved. We wish to apply a similar approach to the Dirac equation. First, we need to determine the 4-component bound-state Helmholtz kernel, which we do by following the process detailed in [88]. Starting with the equation,

$$(h_d - \varepsilon I_4) G_d^{\mu}(\mathbf{r}) = \delta(\mathbf{r}) I_4, \tag{3.99}$$

we introduce the 4-component operator h_d,

$$h_d = c\alpha \cdot \mathbf{p} + \beta mc^2 I_4, \tag{3.100}$$

where G_d^{μ} represents the 4-component relativistic Green's function.

Interestingly, the kernel G_d^{μ} can be expressed in terms of the bound-state Helmholtz Green's kernel, G from equation (2.71) in section 2.8.2 as:

$$G_d^{\mu}(\mathbf{r}) = \frac{1}{2mc^2} [h_d + \epsilon I_4] G^{\mu}(\mathbf{r}), \tag{3.101}$$

where μ is defined as:

$$\mu = \sqrt{\frac{m^2c^4 - \epsilon^2}{mc^2}}. \tag{3.102}$$

To demonstrate this, we adopt the approach outlined by Blackledge and Babajanov in [88]. We begin by observing that the expression $(h_d - \varepsilon I_4)(h_d + \varepsilon I_4)$ can be rewritten as:

$$(h_d - \varepsilon I_4)(h_d + \varepsilon I_4) = -c^2 \nabla^2 I_4 + m^2c^4 I_4 - \varepsilon^2 I_4 \tag{3.103}$$

$$= -c^2 \left(\nabla^2 I_4 + \frac{\varepsilon^2 - m^2c^4}{c^2} I_4 \right) \tag{3.104}$$

$$= -c^2 \left(\nabla^2 I_4 - \frac{m^2c^4 - \varepsilon^2}{c^2} I_4 \right) \tag{3.105}$$

By setting μ as above, we can replace $(\nabla^2 - \mu^2)I_4$ with $-\frac{1}{c^2}(h_d - \varepsilon I_4)(h_d + \varepsilon I_4)$ in equation (2.70), yielding:

$$\frac{1}{c^2}(h_d - \varepsilon I_4)(h_d + \varepsilon I_4) G^\mu(\mathbf{r} - \mathbf{r}') = \delta(\mathbf{r} - \mathbf{r}')I_4. \tag{3.106}$$

From this, it follows that:

$$(h_d - \varepsilon I_4) G_r^\mu(\mathbf{r} - \mathbf{r}') = \delta(\mathbf{r} - \mathbf{r}')I_4, \tag{3.107}$$

The relativistic bound state Helmholtz kernel G_d^μ's definition, given in equation (3.101), is dependent on the form of h_d as noted by Anderson et al. [72]. In practice, the energy in the Dirac equation is often shifted by the rest energy to align the positive branch of the energy with the non-relativistic energy. For instance, h_d^- is defined as $h_d - mc^2$. In such cases, corresponding adjustments must be made to $\mu \to \mu^-$ and $G_d^\mu \to G_d^{\mu^-}$ to ensure consistency. Now that we have the Green's function for the relativistic bound-state Helmholtz kernel. We can look into solving the Dirac equation in it's integral form.

3.9.2 Solving the Dirac Equation Using Green's Function and Integral Form

We start by introducing the stationary version of the 4-component Dirac equation (3.96) for one electron

$$(h_d + V)\Psi = E\Psi \tag{3.108}$$

or its matrix from:

$$\begin{bmatrix} V + mc^2 & -(\sigma \cdot \mathbf{p}) \\ (\sigma \cdot \mathbf{p}) & V - mc^2 \end{bmatrix} \begin{bmatrix} \Psi^L \\ \Psi^S \end{bmatrix} = E \begin{bmatrix} \Psi^L \\ \Psi^S \end{bmatrix} \tag{3.109}$$

Here, we have introduced an external potential V.

As we did for the Hartree–Fock and Kohn–Sham equations in section 3.7, we need to rewrite the stationary Dirac equation (3.108) into its integral form:

$$\Psi = -\hat{G}_d^\mu[V\Psi] \tag{3.110}$$

Here $\hat{G}_d^\mu$ is the relativistic Bound-state Helmholtz (convolution) operator. It can be applied in three equvivalent ways[72]

$$\hat{G}_d^\mu[\Psi] = \frac{1}{2mc^2}(\Psi * (h_d + E)G^\mu) \tag{3.111}$$

$$= \frac{1}{2mc^2}[h_d + E](\Psi * G^\mu) \tag{3.112}$$

$$= \frac{1}{2mc^2}(([h_d + E]\Psi * G^\mu)) \tag{3.113}$$

In the first variant, the operator (h_d+E) is applied directly to the Bound-State kernel G^μ, followed by the convolution. In the second, the operator is applied to the result of the convolution between g and the wavefunction Ψ. In the third, the operator is applied to the orbital Ψ before the convolution with the kernel g is performed.

The first and third methods have an inherent advantage in that the integral operator can smooth out the numerical noise introduced by the derivatives in h_d. However, the last two methods do not require altering the kernel g, which is already implemented in codes like `MRCPP` and `VAMPyR`. This makes these methods more convenient for practical applications.

Since we have Ψ on both sides of the equation (3.110) it has to be solved iteratively until self-consitency is achieved:

$$\tilde{\Psi}^{n+1} = -\hat{G}_d^{\mu^n}[V\Psi^n] \tag{3.114}$$

Again we are using the tilde notation since the operation $\hat{G}_d^\mu$ does not conserve the norm of the wavefunction Ψ. So we need to normalize Ψ between each iteration:

$$\Psi^{n+1} = \frac{\tilde{\Psi}^{n+1}}{||\tilde{\Psi}^{n+1}||} \tag{3.115}$$

This can be generalized for more than one electron. This has been done by Anderson et al[72]. Where they solve the Dirac Fock equation.

Chapter 4

Very Accurate MultiResolution Python Routines

This chapter provides an overview of the installation procedures and inherent functionalities of `VAMPyR`, a software package implemented using `pybind11` integrating the computational capabilities of `MRCPP` with the approachable scripting ease of Python. We detail several installation approaches to accommodate users' preferences across various platforms, including direct source installation, package management systems, and precompiled binaries.

As we discuss the fundamental data structures that are central to `VAMPyR` (and the underlying software `MRCPP`), readers will gain insight into the software's handling of multidimensional computational challenges.

Advanced users requiring higer control over the software's core functionality have the option to use `VAMPyR`'s advanced modules, which preserve the control of `MRCPP` within a Python framework. The chapter also addresses the usability of `VAMPyR`, providing examples and documentation resources that facilitate a smooth transition for users of varying expertise levels, and underscores the software's role in simplifying the application of multi-resolution analysis.

The sections to follow ensure that readers are acquainted with both the practical usage and the broader context of `VAMPyR`, providing essential knowledge for users to effectively deploy the tool in diverse computational scenarios. From exploring key structures to applying the software to specific operations, this chapter serves as both an introductory guide and a resource for deeper engagement with the utility and adaptability of `VAMPyR` in the field of computational science.

4.1 Installation and Documentation

`VAMPyR` is available under the LGPLv3 license. The source code can be accessed on GitHub within the `MRChemSoft`[89] repository. Users who wish to install the source code directly can utilize `pip` for virtual environment installations or opt for the more traditional `cmake` approach. For users that prefer `conda`. It is also possible to download precompiled binaries compatible with several Linux and macOS architectures from `anaconda.org`[90]. Then installation is a simple command (see Lising 4.1)

Source Code 4.1: Conda installation

```
conda install -c conda-forge vampyr
```

Details on the installation process can be found in the `README` file in the `VAMPyR` GitHub repository[89].

The `VAMPyR` documentation[91] includes a collection of Jupyter notebooks available at Binder[92]. This allows users to interactively try out the code without the need to install the software locally, making it easier for new and potential users to explore the software.

4.2 Key Data Structures

`VAMPyR` is built on top of `MRCPP` and uses `pybind11` to bridge the C++ core of `MRCPP` into a Python environment. It shares several key data structures which are fundamental to its operations. This section highlights the most important structures: the `MultiResolutionAnalysis`<D>, `MWTree`<D>, and `MWNode`<D>, along with the `MWTree`<D> derivatives `FunctionTree`<D> and `OperatorTree`<D>.

Dimensionality in `MRCPP` is handled using C++ templates with the template parameter `<D>`, representing spatial dimensions. This allows for a flexible and efficient implementation[1]. Some operations, such as those involving the Poisson and Helmholtz operators (see sections 2.8.1-2.8.2), are dimension-specific and only work in 3D.

Python, unlike C++, does not support native templates. To manage complexity in `VAMPyR`, which stems from C++ templates in `MRCPP`, the framework abstracts this into three distinct Python submodules. Users select the appropriate submodule based on their problem's dimensionality:

Source Code 4.2: Handling Dimensionality in `VAMPyR`

```python
from vampyr import vampyr1d
from vampyr import vampyr2d
from vampyr import vampyr3d
```

[1]For a deeper understanding of C++ Templates, refer to [93] or [94]

Additionally, `VAMPyR` offers `advanced` submodules for users who require high control and are comfortable with the complexity of `MRCPP`'s functionalities. These functions presented without simplifications in the `advanced` submodules:

Source Code 4.3: Advanced `VAMPyR` Submodules

```
from vampyr import vampyr1d.advanced
from vampyr import vampyr2d.advanced
from vampyr import vampyr3d.advanced
```

For an example of this usage see Listing 4.6.

Throughout this discussion, we refer to the shared key data structures between `MRCPP` and `VAMPyR` with the template parameter `<D>`.

4.2.1 Multiresolution Analysis (`MultiResolutionAnalysis<D>`)

The `MultiResolutionAnalysis<D>` object in `MRCPP` and `VAMPyR` serves as the data structure that defines the multiwavelet basis and the physical domain it covers. It allows functions and operators to be represented and manipulated within the same framework.

4.2.2 MultiWavelet Node (`MWNode<D>`) and MultiWavelet Tree (`MWTree<D>`)

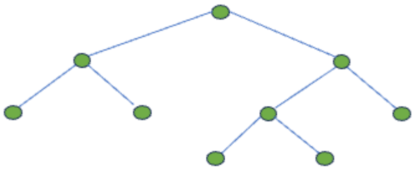
The `MWNode<D>` is a multi-dimensional container with a scaling index n and a translation index $\mathbf{l} = (l_1, l_2, ..., l_d)$. It stores $(k+1)^d$ scaling coefficients $s^{n,f}_{j,l}$ and $(2^d - 1)(k+1)^d$ wavelet coefficients $w^{n,f}_{j,l}$. In general, each `MWNode<D>` give rise to 2^d child nodes, based on a thresholding regime (section 2.5).

The `MWNode<D>` is organized in a hierarchical manner, with the root node symbolizing the entire physical space. Each node apart from the leaf nodes owns 2^d child nodes, each of which holds $1/2^d$ the space of its parent, in line with the discussion in section 1.3.2. This hierarchical architecture of `MWNode<D>` is depicted in Figures 4.1a and 4.1b, providing insight into the organization of an `MWTree<D>` and an `MWNode<D>`.

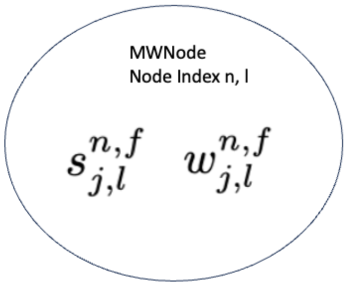
4.2.3 `FunctionTree<D>` and `OperatorTree<D>`

`MWTree<D>` is a base class and is not used directly, its most important derivatives are the `FunctionTree<D>` and `OperatorTree<D>`, which are fundamental in practical computations.

The `FunctionTree<D>` inherits from both the `MWTree<D>` and `RepresentableFunction<D>` classes. Where `RepresentableFunction<D>` is the primary abstraction for analytic functions that can be projected onto the multiwavelet basis. Essentially, `FunctionTree<D>` denotes functions that are numerically projected onto a `MultiResolutionAnalysis<D>`.



(a) The hierarchical structure of an `MWTree<D>`. The top node is referred to as the Root node. The bottom nodes along each branch are referred to as leaf nodes.



(b) An `MWNode<D>` has an Index $(n, \mathbf{l})$ and stores $(k+1)^d$ scaling coefficients $s^{n,f}_{j,l}$ and $(2^d-1)(k+1)^2$ wavelet coefficients $w^{n,f}_{j,l}$.

Figure 4.1: The `MWTree<D>` and `MWNode<D>` structures in `MRCPP`, where (a) represents the hierarchical structure of an `MWTree<D>`, while (b) depicts the structure of an `MWNode<D>`.

The `OperatorTree<D>` is another crucial derivative of `MWTree<2>`. It stores the scaling and wavelet coefficients for operators in the `MultiResolutionAnalysis<D>`. It inherits from the two dimensional `MWTree<2>`, since all operators are applied in a single dimension at a time. Derivative operators only operate in a single dimension at a time and currently all convolution operators are represented as a sum of one-dimensional Gaussian kernels (see section 2.8).

4.3 Intuitive Multiresolution Analysis with `VAMPyR`

This section demonstrates the use of `VAMPyR` for multi-resolution analysis with an emphasis on its Pythonic syntax. `VAMPyR` simplifies the transition from the foundational, C++-based mathematics of `MRCPP` to the more widely-used Python programming environment. It provides users with tools to carry out scientific computations effectively, leveraging Python's well-known syntax and structure.

In the examples that follow, the focus will be on how procedures are implemented in `VAMPyR` compared to `MRCPP`. The aim is to highlight how `VAMPyR` can handle complex computational patterns using Python's clear and concise scripting style.

These examples will also demonstrate the practicality of `VAMPyR`, offering insights into how it streamlines the process of performing multi-resolution analyses and reduces the amount of code required to achieve these tasks.

4.3.1 Interactive Design: VAMPyR vs. MRCPP

Understanding the differences in how users interact with MRCPP and VAMPyR sheds light on the streamlined workflow VAMPyR offers. In MRCPP, operations on FunctionTree<D> objects are explicit, involving direct object manipulation and reference-based function calls. For example, addition in MRCPP requires declaring a FunctionTree<D> and then applying the ‘add‘ function as follows:

Source Code 4.4: MRCPP Addition Operation

```
// Declare the new tree
auto a_tree = FunctionTree<D>(mra);

// Perform the addition
add<D>(prec, a_tree, 1.0, b_tree, 1.0, c_tree);
```

Conversely, VAMPyR abstracts the FunctionTree<D> instantiation, providing a simplified and more Pythonic approach:

Source Code 4.5: VAMPyR Addition Operation

```
a_tree = b_tree + c_tree;
```

While reducing code verbosity and aligning with Python's familiar scripting style, VAMPyR retains the robust computational model underlying both frameworks. Additionally, for use cases that demand a higher degree of control—reminiscent of MRCPP workflows—VAMPyR provides 'advanced' submodules which expose more explicit operations akin to the core C++ library:

Source Code 4.6: Explicit operation in VAMPyR

```
from vampyr import vampyrXd as vp # Replace X with 1, 2 or 3.

a_tree = vp.FunctionTree(mra)
vp.advanced.add(prec, a_tree, 1.0, b_tree, 1.0, c_tree)
```

This dual approach endorses VAMPyR's flexibility—achieving an ease of use for general tasks while providing an option for more complex and controlled operations when needed.

4.3.2 Function Projection in VAMPyR

In any program using VAMPyR (or MRCPP), the initial step is to define a MultiResolutionAnalysis<D> object. This setup in VAMPyR is straightforward, requiring only the essential parameters such as the domain box and the polynomial order k. Selecting k is important as it correlates with the desired precision of calculations—higher precision necessitates a larger k, while for lower precision, a smaller k suffices.

Source Code 4.7: VAMPyR MultiResolutionAnalysis Setup

```
mra = MultiresolutionAnalysis(box=[-1, 1], order=k)
```

With the MultiResolutionAnalysis<D> established, function projection can proceed in a manner consistent with the methodologies outlined in section 2.1.2, specifically the creation of scaling and wavelet projection operators, P^n and Q^n. Below is an example:

Source Code 4.8: Scaling and Wavelet projection operators in VAMPyR

```
# Define an analytic function
f = lambda x: <expression>

# Create scaling and wavelet projection operators
P_n = ScalingProjector(mra, n)
Q_n = WaveletProjector(mra, n)

# Project the function onto scaling and wavelet basis
f_n = P_n(f) # Scaling projection of f to scale n
d_n = Q_n(f) # Wavelet projection of f to scale n
```

These operators can create a FunctionTree<D> that is uniformly populated to a specified scale n[2], with each MWNode<D> either containing only scaling or only wavelet coefficients. This feature is mostly created for teaching purposes. In general FunctionTree<D>s that include both scaling and wavelet coefficients based on a defined precision threshold are constructed:

Source Code 4.9: Function Projection with Precision Control in VAMPyR

```
# Define an analytic function
f = lambda x: <expression>

# Create a projector with precision control
P_eps = ScalingProjector(mra, prec)
# Project the function onto MRA with precision eps
f_eps = P_eps(f)
```

Here, prec serves as the precision threshold, which plays a key role in node generation within the FunctionTree<D>. When a node's approximation of the function satisfies the specified precision, further subdivision of that branch can cease, avoiding unnecessary computational effort. This process, known as thresholding (see section 2.5). Through this mechanism, VAMPyR enhances computational efficiency by focusing detail where it's needed most and allowing for a leaner, more optimized tree structure.

[2]It's important to note that the MultiResolutionAnalysis<D> can have a root scale different from zero.

A notable constraint when performing function projection in VAMPyR, in contrast to MRCPP, is the Global Interpreter Lock (GIL) of Python [95]. The GIL is a mutex that permits only a single thread to govern the Python interpreter at any given time, thereby preventing multi-threaded execution even on systems with multiple CPU cores. Consequently, this restricts VAMPyR to single-threaded operations during function projection, an impediment not encountered with MRCPP. Nonetheless, this limitation can be mitigated in specific scenarios, such as when projecting Gaussian functions. The GaussFunc, a feature inherited and directly ported from MRCPP to VAMPyR, is a case in point. Implemented in C++, GaussFunc effectively bypasses the GIL, enabling parallel execution in particular instances. This is demonstrated in Listing 4.17.

4.3.3 Arithmetic Operations

The ergonomic design of VAMPyR extends to arithmetic operations, enabling users to perform these tasks with syntax that closely resembles standard arithmetic expressions in Python. The interface provided by VAMPyR abstracts the management of FunctionTree<D> objects, simplifying the execution of operations between FunctionTree<D>s or between FunctionTree<D>s and scalars.

For example, adding two FunctionTree<D> objects or modifying one by another is accomplished with the familiar '+' and '+=' operators, similar to how Python handles basic numerical types:

Source Code 4.10: Addition and Increment in VAMPyR

```
a = b + c  # Add two FunctionTrees
b += c       # Increment one FunctionTree by another
```

Subtraction is equally intuitive, allowing for the direct subtraction of one FunctionTree<D> from another using the '-' and '-=' operators:

Source Code 4.11: Subtraction and Decrement in VAMPyR

```
a = b - c  # Subtract one FunctionTree from another
b -= c       # Decrement one FunctionTree by another
```

Multiplication, too, can be performed cleanly between two FunctionTree<D> objects or between a FunctionTree<D> and a scalar:

Source Code 4.12: Multiplication in VAMPyR

```
a = b * c  # Multiply two FunctionTrees
b *= 2.0   # Scale a FunctionTree by a factor
```

While VAMPyRdirectly supports addition, subtraction, and multiplication, operations such as raising to a power or division are primarily designed to interact with scalars. Here are examples that illustrate using power and division with a FunctionTree<D>:

Source Code 4.13: Power and Division Operations with Scalars in VAMPyR

```
c = b ** 3  # Raise FunctionTree coefficients to power of 3
c = b / 2.0 # Divide FunctionTree coefficients by 2

# In-place power and division operations
b **= 3
b /= 2.0
```

These operators not only streamline the overall coding experience but also mitigate potential errors that might arise from handling `FunctionTree<D>` references and operations manually. By offering an intuitive interface, `VAMPyR` provides researchers the ability to focus on the core aspects of their scientific work, encoding mathematical expressions directly into their computational workflows with ease and precision.

An additional benefit of these dunder methods is the seamless integration with NumPy, a widely-used library in Python for array and matrix manipulation. The dunder methods allow `FunctionTree<D>` objects in `VAMPyR` to be used directly in NumPy arrays, completely utilizing the efficient computation that NumPy's design provides. These combined capabilities of `VAMPyR` and NumPy allow for vectorized operations on multiple function trees at once, improving code readability while maintaining computational efficiency.

For example, if A and B are two NumPy arrays populated with function trees, we can perform simple arithmetic like '$C = A + B$'. Here, C is a new array resulting from the addition of corresponding function trees in A and B. This application of function-level operations across entire arrays of function trees at once is a powerful testament to the enhanced capabilities of `VAMPyR` when combined with NumPy. There are examples of this in [22] and in the Jupyter notebooks appended to [49].

4.3.4 Derivative Operators

Derivative operations are pivotal in the analysis of functions, particularly when dealing with physical phenomena where gradients and divergence play crucial roles. `VAMPyR` offers a suite of derivative operators to carry out these operations efficiently and accurately within the multiwavelet framework.

In `VAMPyR`, the handling of derivatives is adapted to the particularities of the multiwavelet basis, which is inherently discontinuous. As explained in section 2.7, classical derivative operators do not exist in the usual sense within this context. Thus, we employ different methods suited to various situations:

1. The `ABGVOperator`[28] utilizes the concept of a weak derivative to compute derivatives through integration by parts, best suited for non-continuous functions where a conventional derivative is undefined.

2. The BSOperator[31] relies on a B-spline derivative operator for continuous functions and is particularly useful for higher order derivatives.

The choice between these two operators should be informed by the nature of the function (continuous or not) and the requirements of the derivative computation (single or higher order derivatives). For instance, the ABGVOperator would be preferable for a function with discontinuities, while the BSOperator would be optimal for a smooth function where multiple derivatives are needed.

Given these considerations, two principal types of derivative operators are employed in VAMPyR: the BSOperator and the ABGVOperator.

Source Code 4.14: Using ABGVOperator for Derivatives in VAMPyR

```python
a, b = (0.5, 0.5) # Correspond to central differences
D = vp.ABGVOperator(mra, a, b)

# Compute the gradient components
f_x = D(f, 0) # Derivative along x-axis
f_y = D(f, 1) # Derivative along y-axis
f_z = D(f, 2) # Derivative along z-axis
```

Alternatively, first- and second-order derivatives can be calculated using the BSOperator:

Source Code 4.15: Using BSOperator for First and Second Derivatives in VAMPyR

```python
# First-order derivative operator
D = vp.BSOperator(mra, 1)

# Compute the first-order gradient components
f_x = D(f, 0)
f_y = D(f, 1)
f_z = D(f, 2)

# Second-order derivative operator
D2 = vp.BSOperator(mra, 2)

# Compute the second-order derivatives along each dimension
f_xx = D2(f, 0)
f_yy = D2(f, 1)
f_zz = D2(f, 2)
```

In vector calculus, the gradient of a scalar field f is a vector field ∇f that points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is the greatest rate of change:

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix}. \tag{4.1}$$

Conversely, the divergence of a vector field $\mathbf{F} = (F_x, F_y, F_z)$ measures the extent to which the vector field is expanding or converging at a given point:

$$\text{div } \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}. \tag{4.2}$$

Utilizing these concepts, `VAMPyR` efficiently computes the gradient and divergence of functions as demonstrated below:

Listing 4.1: Calculating Gradient and Divergence in `VAMPyR`

```
# Given f is a FunctionTree object representing a scalar field

# Compute gradient vector
grad_f_vec = gradient((oper=D, inp=f)

# Compute divergence, yielding Laplacian of f
lap_f = divergence(oper=D, inp=grad_f_vec)
```

This implementation in `VAMPyR` offers an intuitive and readable approach to conducting differentiation operations on `FunctionTree<D>` while abstracting the complexity of underlying multiwavelet-based calculations.

4.3.5 Convolution Operators

Convolution operators play an essential role in various physical applications, such as solving differential equations related to electrostatic potential and quantum mechanical wavefunctions. These operators, such as the Poisson and Helmholtz operators, are designed around separable Gaussian kernels that significantly reduce the computational complexity from $O((k+1)^{2d})$ to $O(dM(k+1)^{d+1})$, as outlined in Section 2.8.

The Poisson operator, which addresses the Poisson equation described in Section 2.8.1, and the Helmholtz operator, which is applicable to the Bound State Helmholtz equation detailed in Section 2.8.2, are readily accessible in `VAMPyR` as follows:

Source Code 4.16: Using the Poisson and Helmholtz operators in `VAMPyR`

```
poisson_operator = PoissonOperator(mra, prec)
helmholtz_operator = HelmholtzOperator(mra, mu, prec)
```

```
# Apply the Poisson operator to a function
g_poisson = poisson_operator(f)

# Apply the Helmholtz operator to a function
g_helmholtz = helmholtz_operator(f)
```

Furthermore, custom convolution kernels can be constructed by combining Gaussian functions. Each Gaussian function in the expansion, defined by parameters β and α, can be appended to form a kernel object. The subsequent convolution operator, built using this kernel, is then capable of convolving any given function within the multiwavelet basis:

Source Code 4.17: Creating and using a custom convolution kernel in `VAMPyR`

```
# Initialize a convolution kernel with Gaussian expansions
kernel = GaussExp()
kernel.append(GaussFunc(beta, alpha))

# Construct the convolution operator
T = ConvolutionKernel(mra, kernel, prec)

# Apply the convolution operator to a FunctionTree object
g = T(F)
```

4.4 Implementation

`VAMPyR` is built on top of `MRCPP`, using the `pybind11` library to facilitate the integration of its C++ core into a Python environment. The union of these two technologies enables `VAMPyR` to meld the high-performance computation of `MRCPP` with the ease-of-use and accessibility of Python.

These key technologies underpin the implementation details of `VAMPyR`. The `pybind11` library is instrumental in exposing the C++ data structures and algorithms of `MRCPP` to Python, offering users the computational efficiency of `MRCPP` while enabling them to interact with it through a Pythonic interface.

Within this fusion, the automatic conversion of data types and the direct invocation of `MRCPP`'s algorithms in Python play a crucial role. This preserves the performance benefits of C++ while offering the flexibility and ease-of-use inherent in Python workflows. Such capabilities are further elaborated in the following subsections which delve into the binding process, multi-dimensional data handling, and the Pythonic interface of `VAMPyR`.

4.4.1 Modifying the `FunctionTree<D>` for `VAMPyR`

The `FunctionTree<D>` is a fundamental object for users of `VAMPyR` and `MRCPP`. To ensure its proper functionality within `VAMPyR`, it is essential to not only import the `FunctionTree<D>` class from `MRCPP` but also to establish its inheritance from the `MWTree<D>` and `RepresentableFunction<D>` classes. This step is crucial for the compatibility of methods that are transferred from `MRCPP` to `VAMPyR`, which often require inputs of type `MWTree<D>` or `RepresentableFunction<D>`. If the inheritance of the `FunctionTree<D>` is not correctly specified, these methods may fail to recognize the tree as a valid input.

The `FunctionTree<D>` class is the cornerstone of user interaction within both `MRCPP` and `VAMPyR`, serving as the principal data structure through which various computational operations are executed. In `MRCPP`, users have access to a suite of useful methods, exemplified as follows:

Source Code 4.18: Useful methods of a FunctionTree in MRCPP

```
// Direct integration of the function
auto integral = f_tree.integrate()
// Computes the L2 norm squared
auto norm_squared = f_tree.getSquareNorm()
// In-place normalization of the FunctionTree<D> instance
f_tree.normalize()
```

As we transition to `VAMPyR`, the `FunctionTree<D>` remains the pivotal object for user operations. Operations similar to those in Listing 4.18 are performed in `VAMPyR` as shown below:

Source Code 4.19: Updated methods on a FunctionTree in `VAMPyR`

```
# Direct integration of the function
integral = f_tree.integrate()
# Computes the L2 norm
norm = f_tree.norm()
# Normalization that yields a new FunctionTree<D> instance
f_normalized = f_tree.normalize()
```

The updates made in `VAMPyR` enhance user experience: the `getSquareNorm` method from `MRCPP` is changed to `norm`, aligning with the `numpy` naming convention. The refactored `normalize` method now returns a new `FunctionTree<D>` instance, reflecting user feedback.

Arithmetic Operations in `VAMPyR`

A notable divergence between `VAMPyR` and `MRCPP` lies in the implementation of arithmetic operations. The convenience in `VAMPyR` comes from overloading standard arithmetic operators to work with `FunctionTree<D>` objects, a design choice that is exemplified by Listings 4.10 through 4.13. This approach not only makes the code more intuitive but also secures against common errors when dealing with complex object manipulations.

The implementation relies on dunder (double underscore) methods, special functions that Python recognizes for operator overloading. In VAMPyR, these methods are defined for various arithmetic operations, such as:

Source Code 4.20: Operator overloading for addition in VAMPyR

```
.def("__add__", &impl__add__<D>, py::is_operator())
```

For instance, the __add__ method is implemented using a template function that generates a new FunctionTree<D> by adding two given FunctionTree<D> objects together:

Source Code 4.21: Implementation of addition operation for FunctionTree

```
template <int D>
auto impl__add__(FunctionTree<D> *inp_a, FunctionTree<D> *inp_b)
    -> std::unique_ptr<FunctionTree<D>> {
    auto out = std::make_unique<FunctionTree<D>>(inp_a->getMRA());
    FunctionTreeVector<D> vec;
    vec.push_back({1.0, inp_a});
    vec.push_back({1.0, inp_b});
    build_grid(*out, vec);
    add(-1.0, *out, vec);
    return out;
};
```

This function begins by creating a unique pointer to a new FunctionTree<D> based on the MultiResolutionAnalysis<D> of the input trees. Then constructs a FunctionTreeVector<D>, a container to store the input function trees with their associated coefficients, before finally invoking the build_grid and add functions to combine the input trees on a common grid and sum their values. Note that all dunder methods for arithmetic's in VAMPyR for arithmetic operations adhere to a similar pattern.

The primary distinction between the methods used in VAMPyR and MRCPP lies in their approach to tree refinement. MRCPP adapts the refinement based on precision. In particular, this is important for multiplication. Since, as discussed in section 2.4, the tree has to be expanded prior to the operation.

In contrast, VAMPyR's dunder methods forego dynamic refinement in favor of a static, predetermined refinement. In general, a union between the trees are taken before the arithmetic operation. An visual illustration of this can be seen in Figure 4.2 for the case of addition. For multiplication, a union is first taken and the tree is refined one level before the multiplication takes place. See Figure 4.3 for a visual illustration. Where it theoretically should be infinite. However, numerical tests have shown that one level of refinement is usually sufficient to keep numerical precision.

A problem with the multiplication operator is how the tree grows with each operation; it is easy to see how this can expand exponentially. VAMPyR's crop method allows users to contain numerical growth. It trims the `FunctionTree<D>` based on a set precision, as depicted in the example below, maintaining accuracy while preventing unwieldy tree expansions.

Source Code 4.22: Combining multiple multiplications with the crop function in `VAMPyR`

```
h = f * f * f * f * f  # Multiple multiplications leading to tree refinement
f.crop(prec)            # Cropping f to maintain precision
h.crop(prec)            # Cropping h to precision after multiplications
```

Thus, while `VAMPyR`'s methods are less adaptable than those of `MRCPP`, they typically meet precision standards. The `crop` method enhances this by recursively trimming nodes starting from the leaf level, preventing unnecessary computational overhead and aligning with the users' precision needs. But as always an `MRCPP` type of multiplication in available in the `advanced` submodule is avaiable if an `MRCPP` type of control is required.

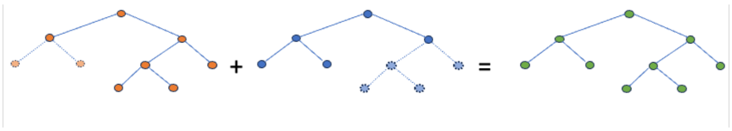


Figure 4.2: Tree structure showcasing the dunder method for addition in `VAMPyR`. The input trees and the resulting output tree are displayed. Nodes shaded in the input trees indicate newly generated nodes necessary for performing the operation. Prior to commencing addition, a union operation aligns the structure of input trees.

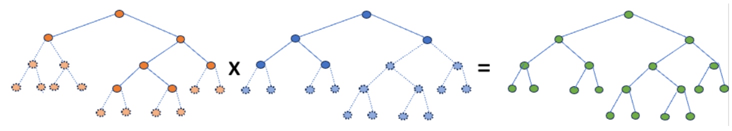


Figure 4.3: Tree structure illustrating the dunder method for multiplication in `VAMPyR`. The input trees and the output tree are depicted. The nodes shaded in the input trees denote nodes specifically generated for the operation. Post aligning the tree structures via a union operation, each leaf node gives rise to new leaf nodes.

4.4.2 Modifying `MRCPP` Functions for `VAMPyR`

Most user-facing functions from `MRCPP` are directly ported and made available through the `advanced` submodule (see Listing 4.3). This module proves useful when users require a level of control equivalent to that of `MRCPP` in their code. An example of such cases is the dunder methods for multiplication, where we risk losing precision and experiencing exponential growth of the `FunctionTree<D>` object in special scenarios.

In the previous section, we observed how the usability of `VAMPyR` was enhanced by augmenting `FunctionTree<D>` during the porting. For instance, by introducing dunder methods for arithmetic operations. In `MRCPP`, most functions operate on the `FunctionTree<D>` by passing the `FunctionTree<D>` to be modified by reference. In `VAMPyR`, function projections, convolution operators, and derivative operators are created and then applied to an analytic function for projection or a `FunctionTree<D>`, and they return a new `FunctionTree<D>`.

In this section, we will explore how operators for projections, derivatives, and convolutions — similar to those introduced in section 4.3 — are implemented within `VAMPyR`. To recapitulate the process, an operator is first instantiated and subsequently applied either to an analytic function, in the case of function projections, or to a `FunctionTree<D>`, when dealing with derivatives and convolutions.

In `VAMPyR`, the design for function projection and operator application prioritizes user convenience. Operators can be instantiated with the necessary parameters and can then be applied to functions using a syntax that is consistent with typical Python functions. The following code demonstrates this process:

Source Code 4.23: Creation and application of an operator in Python.

```python
# Initialize the operator with an mra
O = Operator(mra)

# Apply the operator to the function f
g = O(f)
```

In Listing 4.23, we see how the operator `O` is first defined and then applied directly onto a function f, which would be an analytic function in the case of a function projection or a `FunctionTree<D>` in the case `O` is a Derivative or Convolution Operator. The output `FunctionTree<D>` is then returned directly.

To understand how this is implemented, let us consider the porting code for the `DerivativeOperator<D>` as depicted in Listing 4.24. The most important part of this example, at least with respect to usability, is the implementation of the dunder method `__call__`. In this method, first, a unique `FunctionTree<D>` pointer is created. Then the `apply` function from `MRCPP` with the derivative operator is applied onto the `FunctionTree<D>` pointer, which is then returned. This is all abstracted away, and the creation and application of the derivative operator are streamlined.

Source Code 4.24: Implementation of the `DerivativeOperator<D>` callable method.

```
py::class_<DerivativeOperator<D>>(m, "DerivativeOperator")
    .def(py::init<const MultiResolutionAnalysis<D> &, int, int>(),
         "mra"_a, "root"_a, "reach"_a)
    .def("getOrder", &DerivativeOperator<D>::getOrder)
    .def("__call__",
         [](DerivativeOperator<D> &oper, FunctionTree<D> *inp, int axis) {
             auto out = std::make_unique<FunctionTree<D>>(inp->getMRA());
             apply(*out, oper, *inp, axis);
             return out;
         },
         "inp"_a,
         "axis"_a = 0);
```

Through this method, `VAMPyR` streamlines the process of managing mathematical operations, enabling users to concentrate on scientific inquiries and computational tasks with reduced concern for the complexity of the underlying code.

Chapter 5

Papers

5.1 Paper 1: `VAMPyR` – A High-Level Python Library for Mathematical Operations in a Multiwavelets Representation

Abstract

Wavelets and Multiwavelets have been progressively employed in the domain of Quantum Chemistry to address the limitations encountered with traditional basis sets, such as atomic orbitals and plane waves. They offer numerous numerical benefits: precision, locality, efficient algorithms, and scalable calculations. To harness these advantages, we introduce `VAMPyR`, a Python library developed to facilitate the implementation of Quantum Chemistry equations through a Multiwavelets approach. This library bridges the gap between theoretical constructs and practical computation, providing access to advanced Multiwavelet calculations (algebra, integral, and differential operator application) via Python. We discuss the main features of `VAMPyR`, detail its design, and demonstrate its capabilities and integration with other software platforms through various examples.

Personal Contributions:

- Initiated the project `VAMPyR` and assumed the role of the main developer.
- Authored the predominant portion of the theoretical groundwork presented in the paper.
- Created extensive examples that illustrate the application of the library in complex mathematical problems and quantum chemistry simulations.

- Implemented examples showcasing `VAMPyR`'s compatibility with other Python packages, thus proving its potential for collaborative and interdisciplinary projects.

VAMPyR – A High-Level Python Library for Mathematical Operations in a Multiwavelets Representation

Magnar Bjorgve*,[1] Christian Tantardini,[1, 2] Stig Rune Jensen,[1] Gabriel A. Gerez S.,[1] Peter Wind,[1] Roberto Di Remigio Eikås,[3, 1] Evgueni Dinvay,[1] and Luca Frediani*[1]

[1]*Hylleraas center, Department of Chemistry, UiT The Arctic University of Norway, PO Box 6050 Langnes, N-9037 Tromsø, Norway.*

[2]*Department of Materials Science and NanoEngineering, Rice University, Houston, Texas 77005, United States of America*

[3]*Algorithmiq Ltd, Kanavakatu 3C, FI-00160 Helsinki, Finland*

(*Electronic mail: magnbjor@gmail.com, luca.frediani@uit.no)

(Dated: November 30, 2023)

Wavelets and Multiwavelets have lately been adopted in Quantum Chemistry to overcome challenges presented by the two main families of basis sets: atomic orbitals and plane waves. In addition to their numerical advantages (high precision, locality, fast algorithms, linear scaling, to mention a few) they provide a framework which narrows the gap between the theoretical formalism of the fundamental equations and the practical implementation in a working code. This realization led us to the development of Python code so called with acronym VAMPyR that means Very Accurate Multiresolution Python Routines. VAMPyR encodes the binding to a C++ library for Multiwavelet calculations (algebra, integral and differential operator application) and exposes the required functionality to write simple Python code to solve among others, the Hartree–Fock equations, the generalized Poisson Equation and the Dirac equation up to any predefined precision. In this contribution we will outline the main features of Multiresolution Analysis using multiwavelets and we will describe the design of the code. A few illustrative examples will show the code capabilities and its interoperability with other software platforms.

I. INTRODUCTION

Wavelets and Multiwavelets have emerged in the last few decades as a versatile tool for computational science. Their strength derives from the combination of frequency separation with locality (in contrast to Fourier Transform) and a robust mathematical framework to gauge the precision of a calculation. In particular Multiwavelets have been recently employed within the field of Quantum Chemistry to overcome some of the known drawbacks of traditional Atomic Orbital (AO)-based calculations.[1–9] The code which pioneered this approach is `M-A-D-N-E-S-S`,[10] followed by our own code `MRChem`.[11] To date, they are the only two codes available worldwide for quantum chemistry calculations using multiwavelets.

As the development of `MRChem` unfolded, we found advantageous to separate the mathematical code dealing with the MultiWavelet (MW) formalism in a separate library, called `MRCPP`. Once `MRCPP` was available, we realized it would be useful to be able to access its functionality in a simple and low threshold way. That led us to the delvelopment of `VAMPyR` (Very Accurate Multiwavelets Python Routines), which leverages on the functionality of `MRCPP`, combined with Python's status as a high-level programming language that simplifies the processes of code development, verification, and exploration.

In contrast to traditional quantum chemistry codes typically implemented in lower-level languages such as Fortran, C, or C++, which can pose significant barriers for non- expert programmers, `VAMPyR` offers an accessible and user-friendly platform in Python.

For this reason Python has become a *de facto* standard in the scientific community, widely taught to science students and used extensively in research. Its high-level syntax and rich ecosystem, including libraries such as `NumPy`[12], `SciPy`[13] and `Matplotlib`[14], make it a powerful tool for scientific computing. Moreover, Python's integration with Jupyter Notebooks[15] has facilitated interactive and reproducible research.

The `VAMPyR` project aims at providing the scientific community with a robust tool that builds upon the capabilities of the Multi-Resolution Computation Program Package (`MRCPP`). `MRCPP` is a high-performance C++ library, which provides the tools of Multi–Resolution Analysis (MRA) with Multiwavelets. It implements low-scaling algorithms and a robust mechanism for error control during numerical computations. The goal of `VAMPyR` is to allow a broader audience to make use of these tools while maintaining the original power and efficiency of the `MRCPP` library.

One key feature of `MRCPP` that `VAMPyR` inherits is its parallelization using OpenMP, a multi-platform shared-memory parallel programming model.[16,17] Despite Python's limitations with shared-memory parallelism due to the Global Interpreter Lock (GIL), `VAMPyR` retains high computational performance and efficiency by leveraging OpenMP.[18]

II. MULTIWAVELETS

In this section, we elucidate the mathematical structures that constitute the basis of multiwavelet theories. We focus on the Multiresolution Analysis (MRA), its subsequent wavelet space, and the inherent hierarchical structure, delineated by a sequence of nested subspaces V^n.

A. Hierarchical Subspaces in Multiresolution Analysis

A Multiresolution Analysis (MRA) is defined as a nested sequence of subspaces, V^n, spanned by $k+1$ polynomials, conforming to the structure presented in Eq. (1). This hierarchy asymptotically approaches to square integrable space $L^2(\mathbb{R})$.

$$\cdots \subset V^{-1} \subset V^0 \subset V^1 \subset \cdots \subset V^n \subset \cdots \tag{1}$$

According to the framework put forth by Alpert[19], each subspace V^n partitions the real line into non-overlapping segments, each of length 2^{-n}.

B. Scaling and Wavelet Spaces

The subspace V^n is spanned by a basis $\{\varphi^n_{i,l}\}^k_{i=0}$. This basis can be generated from a set of $k+1$ scaling functions $\{\varphi_i\}^k_{i=0}$ that are defined on the unit interval:

$$\varphi^n_{i,l}(x) = 2^{n/2}\varphi_i(2^n x - l) \tag{2}$$

These scaling functions are typically taken either as Legendre polynomials or interpolating Lagrange polynomials[20]. It follows that each interval in V^n splits into two sub-intervals in V^{n+1}, resulting in a doubling of basis functions with every increment in the level n. This establishes the nested structure $V^n \subset V^{n+1}$ seen in (1). In order to account for features not captured by V^n, a complementary "wavelet space" W^n is defined as:

$$W^n = V^{n+1} \ominus V^n \tag{3}$$

This wavelet space also consists of disjoint intervals on the real line, with a basis constructed from a set of wavelet functions $\{\psi_i\}_{i=0}^k$:

$$\psi_{i,l}^n(x) = 2^{n/2}\psi_i(2^n x - l) \tag{4}$$

The wavelet functions ψ_i, originally conceived by Alpert[19], are orthogonal to the polynomials in the corresponding scaling space, this property, known as "vanishing moments", is essential for the numerical efficacy of approximations and operations in this bases.

Equation (3) leads to the recursive relation Eq. (5):

$$V^N = V^n \oplus W^n \oplus W^{n+1} \oplus \cdots \oplus W^{N-1} \tag{5}$$

C. Two-Scale Filter Relations (MW Transform)

The mathematical framework extends to include two-scale filter relations, between functions at different scales, through the filter matrices:

$$\begin{pmatrix} \psi_l^n \\ \varphi_l^n \end{pmatrix} = \begin{pmatrix} G^{(1)} & G^{(0)} \\ H^{(1)} & H^{(0)} \end{pmatrix} \begin{pmatrix} \varphi_{2l+1}^{n+1} \\ \varphi_{2l}^{n+1} \end{pmatrix} \tag{6}$$

The matrices $H^{(0)}, H^{(1)}, G^{(0)}$, and $G^{(1)}$ are each of dimension $(k+1)\times(k+1)$, and their elements are derived from the wavelet and scaling functions. See Ref. 20 for a comprehensive derivation of these matrices.

The inherent local properties of these two-scale filter relations are of significance for numerical applications. Specifically, they facilitate efficient linear scaling algorithms. The transformation articulated in Eq. (6) constitutes the forward wavelet transform, also known as wavelet decomposition. The inverse of this operation is termed the backward wavelet transform or wavelet reconstruction.

D. Function projection onto the multiwavelet space

Function projections are an integral part of the MRA framework and associated basis functions. These projections facilitate the representation of arbitrary functions within the confines of the theory established above, serving as a conduit between abstract mathematical concepts and practical, computable forms.

Consider first the scaling space V^n. This space is associated with a projector P^n, employed to approximate a function in one dimension (1D):

$$f(x) \approx P^n f(x) = \sum_{l=0}^{2^n-1} \sum_{j=0}^{k} s_{j,l}^{n,f} \varphi_{j,l}^n(x), \tag{7}$$

Here, $s_{j,l}^{n,f}$ are the *scaling coefficients*, given by:

$$s_{j,l}^{n,f} = \int f(x) \varphi_{j,l}^n(x) dx. \tag{8}$$

Similarly, a wavelet projector Q^n is associated with the wavelet space W^n:

$$Q^n f(x) = \sum_{l=0}^{2^n-1} \sum_{j=0}^{k} w_{j,l}^{n,f} \psi_{j,l}^n(x), \tag{9}$$

The wavelet coefficients $w_{j,l}^{n,f}$ are defined analogously through a projection integral:

$$w_{j,l}^{n,f} = \int f(x) \psi_{j,l}^n(x) dx. \tag{10}$$

E. Adaptive Projection

Within the framework of multi-resolution representation, as formalized by Equation 5, we use the notations $f^n = P^n f$ and $df^n = Q^n f$ to express the representation of a function in the following manner:

$$f^N = f^0 + \sum_{n=0}^{N-1} df^n. \tag{11}$$

This equation gains special significance when considering that the wavelet coefficients, w_l^n, approach zero for smooth functions. The function representation can thus be rigorously assessed by examining the wavelet coefficients at a specific scale n and translation l:

$$|w_l^n| < \varepsilon. \tag{12}$$

This mathematical property provides the foundation for an adaptive projection methodology. Rather than constraining the function's projection to a fixed scale n, the scale may be incrementally increased to $n+1$. Utilizing the two-scale filter relations specified in Equation (6), we can compute the wavelet coefficients and assess their magnitude. If these coefficients meet predefined precision criteria at a specific translation l, that branch of the function representation can be truncated. This truncation effectively focuses computational and data resources only where high precision is necessary. Should the desired precision level not be reached, the refinement process continues in a recursive manner.

The strategy takes full advantage of the wavelet basis' inherent adaptability for efficient function representation. Computational and data resources are thereby allocated where they are most beneficial, facilitating high-precision representations.

F. Operator Application in Multiwavelet Framework – Non-Standard Form

The projection of an operator T onto an MRA can be represented using a non-standard form, expressed as follows:

$$T^N = T^0 + \sum_{n=0}^{N} (A^n + B^n + C^n) . \tag{13}$$

In this decomposition, each term at a given scale n is uniquely defined:

$$T^n = P^n T P^n , \qquad A^n = Q^n T Q^n , \tag{14}$$

$$B^n = Q^n T P^n , \qquad C^n = P^n T Q^n . \tag{15}$$

Here A^n, B^n, and C^n components exhibit sparsity properties, similar to what is observed for functions, and are therefore leading to fast, and often linearly scaling algorithms, for a any arbitrary predefined precision. The purely scaling part T of the operator is only required at the coarsest scale where only a handful of grid points are present. Another important feature of the non-standard (NS) form is the absence of coupling between different scales, which allows to preserve the adaptive precision of the representation, and independent or asynchronous operator application across different scales, which boost computational efficiency.

III. `VAMPYR`

In the architecture of `VAMPyR`, `MRCPP` serves as the foundational layer, integrated into Python through `Pybind11`, as illustrated in Figure 1. This setup enables seamless interoperability between Python and C++, allowing `Pybind11` to automatically convert many standard C++ types to their Python equivalents and vice versa. This leads to a more Pythonic and natural interface to the `MRCPP` codebase.

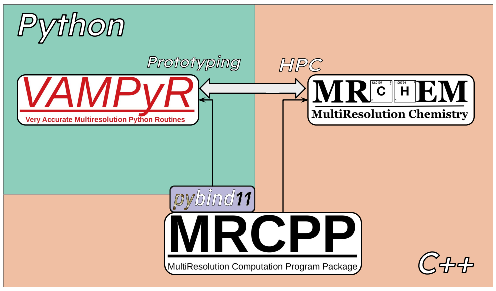

Python
Prototyping
HPC
VAMPyR
Very Accurate Multiresolution Python Routines
MRCHEM
MultiResolution Chemistry
pybind11
MRCPP
MultiResolution Computation Program Package
C++

Figure 1. Connections between MRChem, MRCPP and VAMPyR. MRCPP implements a high-performance MRA framework with Multiwavelets (MW). It has function and operator representations as well as fast operator application. MRChem uses MRCPP as an external library to solve electronic structure systems. VAMPyR imports features from MRCPP using Pybind11 and brings intuitive design and easy prototyping through Python. It is used as a python library and is therefore compatible with other quantum chemistry software. VAMPyR doesn't include functionality from MRChem.

For the development of VAMPyR, we opted for Pybind11 as our binding framework, chiefly for its robustness, maintainability, and alignment with our development objectives. Pybind11 is a lightweight, header-only library that utilizes modern C++ standards to automatically infer type information, thus streamlining the binding process and enhancing overall code quality.

To facilitate accessibility and streamline the user experience, multiple installation options are available for VAMPyR and MRCPP. The source code for both is openly accessible and distributed under the LGPLv3 license on GitHub, specifically within the MRChemSoft repository[21]. For users who prefer a simplified installation procedure, binary packages are also provided through the Conda package manager, which is compatible with various Linux and macOS architectures[22,23]. Each new release triggers an automated build process that uploads the binary packages to the Conda Forge channel on anaconda.org, thereby easing the installation process and incorporating all requisite dependencies. The packages are designed to be compatible with current Python versions, ranging from 3.8 to 3.11. For a practical illustration of the installation process using Conda, see Listing 1. Thanks to its availability on the Conda package manager, VAMPyR is well-suited for seamless integration into existing scientific computing workflows. This makes it a valuable tool for both researchers and professionals involved in high-performance scientific computations. Moreover, its Pythonic interface allows VAMPyR to capitalize on the extensive user base of the

Listing 1. Installing VAMPyR using Conda from the terminal.

```
conda install vampyr -c conda-forge
```

Python community.

A. Implementation

In the development of VAMPyR, the choice of a binding framework is critically important. Several alternatives are available, such as Cython, SWIG, and Boost.Python. We selected Pybind11 to ensure robust and maintainable bindings between C++ and Python. Pybind11 aligns well with our development objectives and offers multiple advantages. Specifically, it is a lightweight, header-only, and standalone library. It employs modern C++ standards to automatically infer type information, thereby streamlining the binding process significantly. This enhances code maintainability, minimizes the potential for bugs, and contributes to the library's overall efficiency and ease of use.

The integration of MRCPP into Python through Pybind11 serves as the foundational layer for VAMPyR. Pybind11 enables seamless communication between Python and C++, capable of automatically converting many standard C++ types to their Python counterparts, and vice versa. This feature results in a more natural and Pythonic interface to the MRCPP code.

In MRCPP, C++ template classes and functions are utilized to provide abstraction over the dimensionality of the simulation box. These templates enable the generic implementation of data structures and algorithms for problems in D dimensions.[24] The specialization for 1-, 2-, and 3-dimensional problems is performed at compile-time, thereby eliminating any impact on runtime performance.[25]

In Python, native template constructs are absent, presenting a challenge for dimension-specific specialization. In VAMPyR, we emulate MRCPP's generic approach by implementing dimension-dependent bindings using Pybind11. Specifically, our binding code incorporates template classes and functions similar to those in MRCPP. This design permits the relatively straightforward addition of functionality for new dimensions, such as 4-dimensional problems, requiring only the insertion of a new function invocation in the binding source code. MRCPP's 1-, 2-, and 3-dimensional template classes and functions are directly bound to their corresponding dimensions in VAMPyR, as demonstrated in Listing 2.

Listing 2. Importing dimensional-dependent bindings from VAMPyR.

```python
from vampyr import vampyr1d
from vampyr import vampyr2d
from vampyr import vampyr3d
```

Through this approach, VAMPyR manages to uphold the dimensionality-flexible philosophy inherent in MRCPP, despite Python's limitations in handling generic datatypes.

While the MultiResolutionAnalysis and FunctionTree classes in VAMPyR are inherited from MRCPP, there are noteworthy distinctions between the vanilla C++ and the corresponding Python classes. These differences are introduced to *enhance* the Python version of the MRCPP classes with syntax sugar, making them uniquely suited for quantum chemistry computations in a Python environment. Among these enhancements are the overload of various *dunder* (double underscore, also known as *magic*) methods that have been added to the classes to extend their functionalities and make them more Pythonic.

These *dunder* (or magic) methods enable the use of Python's built-in operators on FunctionTree objects. For instance, the addition (__add__) and multiplication (__mul__) operators allow for the direct use of + and * operators with FunctionTree objects. This improves readability and ease-of-use, as it allows developers to write more intuitive code.

As an example, addition of two FunctionTree objects can be expressed in natural Python syntax (listing III A). Where this is possible thanks to the binding code in listing III A.

Listing 3. Syntax sugar for FunctionTree addition in action.

```python
f_tree1 = vp.FunctionTree(MRA)
f_tree2 = vp.FunctionTree(MRA)
f_tree_sum = f_tree1 + f_tree2
```

Listing 4. C++ binding for operator +.

```
// allocate new FunctionTree for the output
auto out = std::make_unique<FunctionTree<D>>(inp_a->getMRA());
// arrange input operands in vector of summands
FunctionTreeVector<D> vec;
vec.push_back({1.0, inp_a});
vec.push_back({1.0, inp_b});
// build grid for output value based on summands
build_grid(*out, vec);
// perform summation with no additional refinements
add(-1.0, *out, vec);
// return sum of FunctionTree objects
return out;
```

These additions provide `VAMPyR` with a strong advantage over the C++ implementation of `MRCPP`. The use of *dunder* methods makes `VAMPyR` more compatible with Python's syntactic sugar and thereby makes the package more accessible and appealing to the broad Python community.

The above Python code can be equivalently written in desugared form as:

Listing 5. De-sugared syntax for `FunctionTree` addition using the `advanced` bindings submodule.

```
from vampyr import vampyr3d as vp

f_tree1 = vp.FunctionTree(MRA)
f_tree2 = vp.FunctionTree(MRA)

f_tree_sum = vp.FunctionTree(mra)

vp.advanced.add(-1.0, f_tree_sum, 1.0, f_tree1, 1.0, f_tree2)
```

Although the above listing is more involved than simply applying an arithmetic operator, it does provide more flexibility and control which might be necessary in some specific applications.

IV. MATHEMATICS WITH VAMPYR

In this section we display some of the theoretical consepts discussed in II with practical examples using `VAMPyR`.

A. The MultiResolutionAnalysis Object in VAMPyR

The foundation of `VAMPyR`'s internal operations is built upon the `MultiResolutionAnalysis` object. This object encapsulates essential information about the physical space being modeled, along with configurations for the scaling and wavelet basis functions as defined in the theory of MRA (see Section II).

a. Standard Usage For users seeking a straightforward implementation, `VAMPyR` provides a "Standard Usage" option. In this approach, users are only required to specify the box size and polynomial order. The remaining parameters, such as the choice of interpolating polynomials for the scaling basis, are automatically configured by the library. Listings 6 illustrate how to create an MRA object for unit cells either centered at the origin or anchored at the origin's left corner.

Listing 6. Standard usage of the MRA object.

```python
# For a unit cell centered at the origin
MRA = vp.MultiResolutionAnalysis(box=[-L,L], order=k)
# For a unit cell with left corner at the origin
MRA = vp.MultiResolutionAnalysis(box=[L], order=k)
```

b. Advanced Usage For users requiring more control over the computational setup, `VAMPyR` offers an "Advanced Usage" approach. This method allows for the customization of various parameters, including the world box dimensions and the specific type of scaling basis, such as Legendre Polynomials. This level of customization grants the user full control over the basis configuration, as demonstrated in Listing 7. Thus, `VAMPyR` can cater to a wide range of users, from those who prefer the ease of "Standard Usage" to those who require the additional control offered by

Listing 7. Advanced usage of the MRA object.

```
# Construct scaling basis
scaling_basis = InterpolatingBasis(order) # or LegendreBasis(order)
# Define world box
world_box = BoundingBox(scale=0, corner=[x0, y0, z0],
            nboxes=[Nx, Ny, Nz], scaling=[a, b, c])
            # in 3D, for 1D and 2D replace list of size 3 with list of size 1 or 2
# Construct MRA
MRA = vp.MultiResolutionAnalysis(box=world_box, basis=scaling_basis)
```

"Advanced Usage". This flexibility allows users to choose the level of interaction with the MRA object based on their specific needs and expertise.

B. Function Projectors

In `VAMPyR`, functions are represented using a tree-based data structure, specifically a graph, termed a `FunctionTree`. This structure naturally arises from the Multi-Resolution Analysis (MRA) framework outlined in Section II D.

a. Tree Structure and Nodes The `FunctionTree` consists of interconnected nodes organized hierarchically. The root node represents the entire physical domain, and each non-terminal node or “branch” begets 2^d child nodes while connecting to a single parent node. Terminal nodes, or “leaves,” possess a parent but have no children. Here, d represents the dimensionality of the system, which `VAMPyR` supports as 1, 2, or 3.

b. Node Properties Each node corresponds to a d-dimensional box within the physical domain and encapsulates information about a specific set of scaling and wavelet functions defined therein. Specifically, a node stores $(k+1)^d$ scaling coefficients and $(2^d-1)(k+1)^d$ wavelet coefficients, which are governed by the same scaling, n, and translation vector $\mathbf{l} = (l_1, l_2, \ldots, l_d)$. These parameters dictate the node’s spatial size and position.

C. Scaling and Wavelet projectors in `VAMPyR`

Building on the theoretical concepts described in Section II D and the `FunctionTree` architecture discussed in Section IV B, this section provides practical examples to further elucidate these ideas.

To facilitate the implementation of function projections into scaling and wavelet spaces, `VAMPyR` provides the `vp.ScalingProjector` and `vp.WaveletProjector` classes. In Listing 8, we define an abstract analytic function f denoted as *analytic function*. This serves to illustrate the essence of how to project a function using the `vp.ScalingProjector` and `vp.WaveletProjector` classes. Instances `P_n` and `Q_n` are then created to perform the projection into the scaling and wavelet spaces. The resulting functions, denoted `f_n` and `df_n`, are fully described by their mathematical definitions in Equations (7) and (9), respectively.

Listing 8. Function Projection in `VAMPyR` by Scaling and Wavelet projectors.

```
f = lambda x: <analytic function>
P_n = vp.ScalingProjector(MRA, scale=n)
Q_n = vp.WaveletProjector(MRA, scale=n)
f_n = P_n(f)
df_n = Q_n(f)
```

Figures 2 and 3 provide visual insights into the projection of an exponential (Slater) function at different scales. Each figure consists of two subfigures: the left-hand side displays the scaling function space (Figures 2(a) and 3(a)), while the right-hand side illustrates the wavelet function space (Figures 2(b) and 3(b)).

In Figure 2, the projection onto the root scale captures only the general characteristics of the Slater function. The vertical lines in both subfigures delineate the physical space spanned by $k+1$ scaling and wavelet functions, offering a broad approximation of the function.

Figure 3, on the other hand, showcases the projection at the 4th scale. The scaling function space on the left-hand side captures more structural details, as indicated by the nuanced curve. Despite the improvements, it still falls short of replicating the sharpness of the original Slater function. On the right-hand side, the reduced magnitude of the wavelet coefficients suggests a more accurate approximation. Importantly, this also highlights areas around the function's cusp

Listing 9. Adaptive Function Projection in `VAMPyR`.

```
P_eps = vp.ScalingProjector(MRA, prec=epsilon)
f_eps = P_eps(f)
```

where information is still lacking, affirming the role of wavelet space as an indicator of approximation quality.

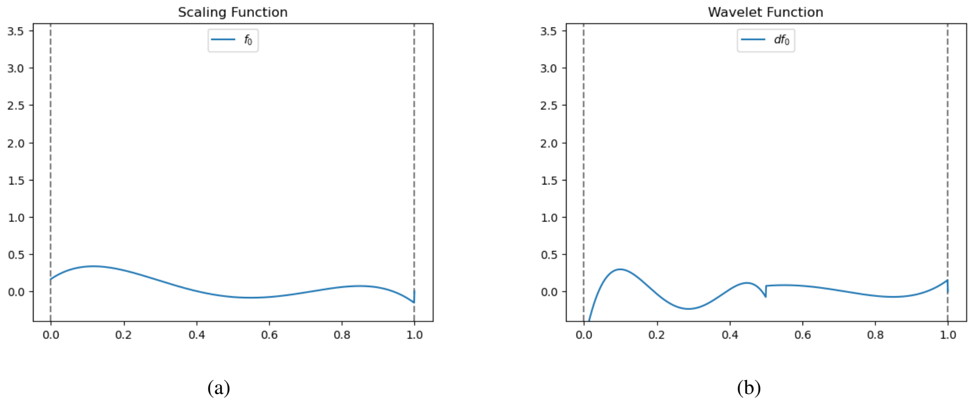


Figure 2. Projection of Slater function onto different bases at root scale. Left: Projection onto scaling function space V_4^0 with vertical lines marking the physical space spanned by $k+1$ scaling functions. Right: Projection onto wavelet function space W_4^0, where vertical lines indicate the physical space spanned by $k+1$ wavelet functions.

D. Adaptive Projection in `VAMPyR`

Building on the mathematical framework presented in Section II E, we delve into the practical aspects of adaptive projection within `VAMPyR`. Unlike the fixed-scale projection illustrated in Listing 8, adaptive projection leverages the *prec* parameter to autonomously fine-tune the approximation's precision, as shown in Listing 9. The visual comparison in Figure 4 between the adaptive and fixed-scale projections (previously shown in Figures 2 and 3) reveals the efficacy of the adaptive method. Notably, it accomplishes a nuanced approximation using fewer basis functions. This is manifested by the non-uniform distribution of vertical lines: sparse in smooth regions and dense in areas with sharp variations.

By adaptively concentrating computational and data resources where they are most efficacious, this technique offers an optimized pathway to both efficient and precise function approximations.

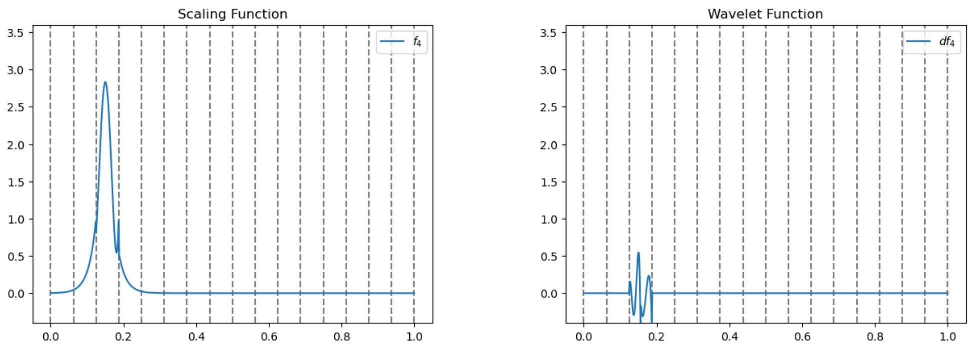
E. Arithmetic Operations in FunctionTrees

In `VAMPyR`, arithmetic operations on `FunctionTree` objects are intuitive and flexible. Standard Python operators like $+$, $-$, $*$, $/$, and $**$ are employed for these elementary operations.

For a detailed example, refer to Listing 10, which showcases arithmetic operations with `FunctionTree` objects.

In-place operations can also be performed on existing `FunctionTree` objects, as demonstrated in Listing 11.

In these examples, f and g are instances of `FunctionTree`, and a is a scalar. These operations provide an efficient and user-friendly way to manipulate `FunctionTree` objects in `VAMPyR`.



(a) (b) .

Figure 3. Projection of Slater function onto different bases at the 4th scale. Left: Projection onto scaling function space V^4, with vertical lines delineating the physical space spanned by $k+1$ scaling functions. Right: Projection onto wavelet function space W^4, where vertical lines mark the physical space spanned by $k+1$ wavelet functions.

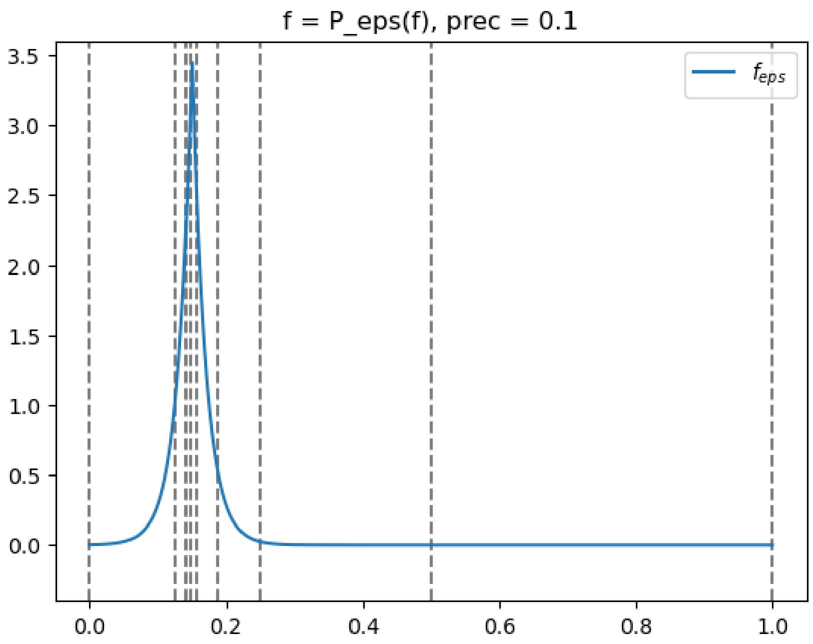


Figure 4. Adaptive projection of the Slater function with *prec* = 1.0×10^{-1}. The non-uniform vertical lines indicate the physical space spanned by the adaptive basis functions, underscoring the method's efficiency and precision.

F. Vectorizing Function Trees with `NumPy`

The seamless integration between the overloaded arithmetic methods in `VAMPyR` and `NumPy`'s flexible data structures provides a range of computational advantages. Among these advantages, the support for **multi-dimensional arrays**, **broadcasting**, and **linear algebra operations** stand out. Although `NumPy` is primarily optimized for integers and floating-point numbers, its data structures can be extended to accommodate custom objects such as `FunctionTrees`. This capability enables the incorporation of `FunctionTrees` within `NumPy` arrays, thus allowing for the full suite of `NumPy`'s array-oriented computing features.

For illustration, consider the example in Listing 12, where we encapsulate a matrix of function trees within a `NumPy` array and execute various matrix operations:

As demonstrated in Listing 12, the integration between `VAMPyR`'s `FunctionTree` objects

Listing 10. Basic Arithmetic Operations in VAMPyR.

```
# Arithmetics with scalars
a_mult_f = a * f  # Scalar on the left
f_mult_a = f * a  # Scalar on the right
f_div_a = f / a   # Division by scalar
f_pow_a = f ** a  # Power operator

# Arithmetics between FunctionTrees
f_mult_g = f * g  # Multiplication operator
f_add_g = f + g   # Addition operator
f_sub_g = f - g   # Subtraction operator
```

Listing 11. In-Place Operations in VAMPyR.

```
# In-place arithmetics
f *= a  # Multiplication with scalar
f *= g  # Multiplication with function
f += g  # Addition with function
f -= g  # Subtraction with function
```

and NumPy arrays simplifies the implementation of complex mathematical operations. Through NumPy's multi-dimensional arrays, it is possible to organize and manipulate arrays of function trees efficiently. Broadcasting allows for the addition of a single FunctionTree (or scalar) to an entire array of FunctionTrees. Furthermore, NumPy's built-in linear algebra functions, such as the matrix multiplication operator '@', can be used for matrix operations. This approach not only improves the readability of the code but also offers computational benefits, as we will further discuss in Section V.

Listing 12. Performing matrix operations using function trees in a NumPy array.

```python
# Creation of a matrix containing function trees
matrix = np.array([[a_tree, b_tree], [c_tree, d_tree]])

# Performing matrix multiplication
matrix_mul = matrix @ matrix

# Performing addition and subtraction
matrix_sum = matrix + matrix
matrix_neg = matrix - matrix

# Multiplication by a scalar
matrix_mul_scalar = matrix * 2

# Demonstrating broadcasting by adding a single function tree to all elements
matrix_bcast_add = matrix + single_tree
```

G. Derivative Operators in VAMPyR

Handling derivatives in VAMPyR presents a unique challenge due to the discontinuous nature of the multiwavelet basis. Traditional derivative operators are inapplicable, making the task of differentiating functions non-trivial. However, VAMPyR offers two specialized operators to address this issue, each tailored for specific requirements concerning the continuity of the function and the desired order of the derivative. These methods are based on the works by Alpert et. al[20]and Anderson et. al.[26] approaches.

Both the ABGVOperator and the BSOperator apply the operators following the non-standard form as described in Subsection II F, allowing for efficient and accurate calculations.

1. ABGVOperator: Designed for non-continuous functions, this operator provides a weak formulation for first-order derivatives. It is primarily suitable for handling first-order derivatives and offers a balance between accuracy and computational efficiency. Usage is demon-

strated in Listing 13.

2. `BSOperator`: This operator is more versatile and designed for continuous functions. It accommodates higher-order derivatives by transforming the function onto a B-spline basis before differentiation. See Listing 14 for a practical example.

Both operators provide distinct advantages depending on the application, making `VAMPyR` a flexible tool for a variety of computational scenarios.

Listing 13. First-order derivatives using `ABGVOperator`.

```
D = vp.ABGVOperator(MRA)


# x, y, z derivatives using ABGVOperator
f_x = D(f, 0)
f_y = D(f, 1)
f_z = D(f, 2)
```

H. Convolution Operators in `VAMPyR`

In `VAMPyR`, convolution operators serve as more than just a reliable alternative to poorly-defined derivative operators. They are essential tools for solving integral forms of key equations in Quantum Chemistry, particularly in three-dimensional spaces. The two most crucial operators—the Poisson and the bound-state Helmholtz operators—are directly implemented in the library.

$$K(\mathbf{x},\mathbf{y}) = \frac{e^{-\mu\|\mathbf{x}-\mathbf{y}\|}}{\|\mathbf{x}-\mathbf{y}\|}, \tag{16}$$

where $\mu = 0$ yields the Poisson kernel, and $\mu > 0$ corresponds to the bound-state Helmholtz kernel. Importantly, both operators are implemented as a sum of Gaussian functions, consistent with the general approach for constructing convolution operators in `VAMPyR`.

The MW basis is particularly well-suited for handling these equations due to its sparse function representations and non-standard form. Furthermore, these kernels can be approximated as a sum of Gaussians:

Listing 14. First- and second-order derivatives using BSOperator.

```
D1 = vp.BSOperator(MRA, 1) # First-order derivative operator
D2 = vp.BSOperator(MRA, 2) # Second-order derivative operator

# First-order derivatives
f_x = D1(f, 0)
f_y = D1(f, 1)
f_z = D1(f, 2)

# Second-order derivatives
f_xx = D2(f, 0)
f_yy = D2(f, 1)
f_zz = D2(f, 2)
```

$$K(\mathbf{x},\mathbf{y}) \approx \sum_{i=1}^{M} \alpha_i \exp(-\beta \|\mathbf{x}-\mathbf{y}\|^2). \tag{17}$$

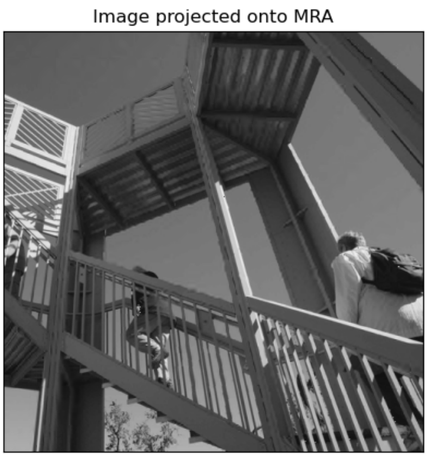
This Gaussian approximation, combined with the tensor product nature of the MW basis, allows for precise and fast operator applications.

Listing 15. Creating and applying Poisson and Helmholtz operators in VAMPyR.

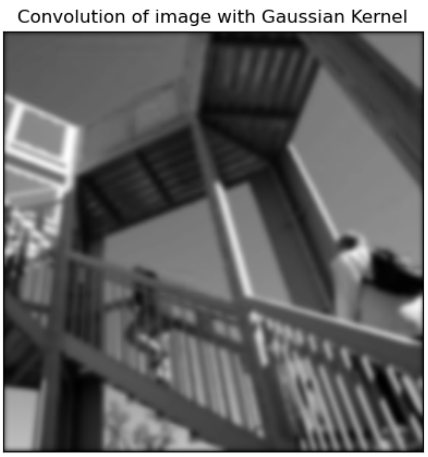
```
P_oper = vp.PoissonOperator(mra, prec)
H_oper = vp.HelmholtzOperator(mra, mu, prec)

g_tree = P_oper(f_tree)
g_tree = H_oper(f_tree)
```

A more general method for constructing convolution operators based on Gaussian expansion is also available in VAMPyR. This approach is illustrated with an example in Listing 16. Additionally, to highlight the versatility afforded by the Python framework, Figure 5 demonstrates a



(a) Original image projected onto the Multi-Resolution Analysis (MRA).



(b) Resulting image after performing convolution with a Gaussian Kernel, leading to smoothing or blurring of the original image.

Figure 5. Illustration of convolution process in `VAMPyR`.

non-traditional application in image processing, although `VAMPyR` is not generally optimized for such tasks.

Listing 16. Construction of a Custom Convolution Operator in `VAMPyR`.

```python
# Create a 1D GaussExp object, serving as a container for Gaussians
kernel = vp1.GaussExp()
# Append a Gaussian function to the kernel
kernel.append(vp1.GaussFunc(beta, alpha))
# Construct the convolution operator
T = vp.ConvolutionOperator(mra, kernel, prec)
# Apply the operator to a function tree
g_tree = T(f_tree)
```

V. FOUR LITTLE PIECES OF QUANTUM CHEMISTRY

One of the main technical challenges of quantum chemistry is the large gap between the equations describing the physical nature of the system and their practical realization in a working code. This gap arises because the equations in their original form are in general too complicated to be solved. To achieve equations which have a manageable computational cost, it has so far been necessary to make use of representations which convey most of the physics in a simplified way. We will here present *four little pieces* of quantum chemistry where we show how the MW representation of functions and operators in `MRCPP` and the simple Python interface provided by `VAMPyR` can close this semantic gap.

A. The Self Consistent Field equations of Hartree–Fock and Density Functional Theory

The starting point of most Quantum Chemistry calculations involve one Slater determinant, because it automatically accounts for the Pauli exclusion principle and allows to express quantities in a sum of orbital contributions. In particular, the energy expression for a Slater determinant Ψ in atomic units is given by:

$$E[\Psi] = \sum_i \langle \psi_i | \hat{h} | \psi_i \rangle + \frac{1}{2} \sum_{i,j} \langle \psi_i | \hat{J}_j - \hat{K}_j | \psi_j \rangle \tag{18}$$

Here, $\hat{h}$ is the one-electron Hamiltonian:

$$\hat{h}\phi_i = \left(\hat{T} + \hat{V}_n\right) \phi_i \tag{19}$$

$\hat{T} = -\frac{1}{2}\nabla^2$ is the kinetic energy operator, capturing the energy associated with the motion of the particles. The nuclear potential energy operator, $\hat{V}_n = -\sum_I \left(\frac{Z_I}{|r-R_I|}\right)$, represents the energy from the potential between the nuclei in the system and the electrons. $\hat{J}_j$ and $\hat{K}_j$ denote the Coulomb and exchange interaction operators, respectively, and are defined as:

$$\hat{J}_j \psi_i = \hat{P}\left[|\psi_j|^2\right] \psi_i \tag{20}$$

$$\hat{K}_j \psi_i = \hat{P}\left[\psi_j \psi_i\right] \psi_j \tag{21}$$

By minimizing the energy $E[\Psi]$ with respect to variations in the orbitals, under the constraint that the spatial orbitals remain orthogonal:

$$\langle E \rangle_{HF} = \min E[\Psi] \qquad \langle \psi_i | \psi_j \rangle = \delta_{i,j} \tag{22}$$

, the Hartree–Fock equations are obtained[27]:

$$\hat{F}\psi_i = \left(\hat{h}+\hat{J}-\hat{K}\right)\psi_i = \sum_j F_{ij}\psi_j \tag{23}$$

where $\hat{F} = \hat{T} + v_{nuc} + \hat{J} - \hat{K} + \hat{V}_{xc}$ is the Fock operator and $F_{ij} = \langle \psi_i | \hat{F} | \psi_j \rangle$are its matrix elements in the chosen oribitals.

Traditional quantum chemistry methods make use of an expansion of the orbitals in a fixed set of atomic orbitals χ: the problem is then cast in matrix form and the eigenvalues (energies) and eigenvectors (orbitals) are obtained by standard linear algebra techniques. The obvious advantage is a representation closely related to the physics of the system (atomic orbitals), but this comes at a price: the implementation deals with the representation of such functions, their integrals, their overlaps, and seemingly simple operations such as applying an operator, multiplying two functions, a change of gauge become technically complicated obfuscating the physical significance.

By using MWs orbitals are represented directly in an adaptive real-space grid and the basis is therefore not predefined.

1. Practical Implementation of Hartree–Fock Equations

This discussion will primarily focus on closed-shell systems, where every orbital is doubly occupied with one electron having spin up (alpha) and the other with spin down (beta). The Hartree–Fock equations (23) for these systems can be reformulated as:

$$\left[\hat{T}+\hat{V}\right]\Phi = \Phi F. \tag{24}$$

In the above equation, Φ denotes a vector of doubly occupied orbitals, $F = \langle \Phi | \hat{T} + \hat{V} | \Phi \rangle$ represents the Fock matrix, and $\hat{V} = \hat{V}_{nuc} + 2\hat{J} - \hat{K}$ is the potential operator. The terms here are the nuclear potential:

$$\hat{V}_{nuc} = \sum_I \frac{Z_I}{|\mathbf{r}-\mathbf{R}_I|}, \tag{25}$$

Coulomb and exchange operators. The latter two defined above in equations (20), and (21).

The closed-shell assumption eases our calculations by negating the need to consider unpaired electrons and enabling us to treat spinorbitals as real functions instead of complex ones. Many stable molecules and atoms have closed shell configurations, which is why this is a common scenario in quantum chemistry.

In the context of a multiwavelet approach, we opt to address partial differential equations (PDEs) in their integral form. The closed-shell Hartree–Fock equations (24) are then restructured into their integral representation, facilitating self-consistent resolution via an iterative scheme. This scheme, demonstrated below, continues until self-consistency is reached:

$$\tilde{\Phi}^{n+1} = -2\hat{G}^{\vec{\mu}^n}\left[\hat{V}\Phi^n + \Phi^n\left(\Lambda^n - F^n\right)\right] \tag{26}$$

In the above equation, $\hat{G}^{\vec{\mu}}$ is the bound-state Helmholtz operator, functioning as a convolution operator:

$$\hat{G}^{\mu}[\phi] = \int G^{\mu}(\mathbf{r}')\phi(\mathbf{r}-\mathbf{r}')d\mathbf{r}' \tag{27}$$

with the Helmholtz kernel $G^{\mu} = \frac{e^{-\mu r}}{r}$.

Within this context, $\mu_i^n = \sqrt{-2\lambda_i^n}$ is defined as a positive real number, and $\Lambda_{ij}^n = \lambda_i^n\delta_{ij}$ is a diagonal matrix with elements corresponding to the parameters used in constructing the Helmholtz vector. In practical applications, we set $\lambda_i^n = F_{ii}^n$ to nullify the diagonal elements in $(\Lambda^n - F^n)$.

The programmatic implementation of these equations incorporates a Lowdin orthogonalization of the orbital vectors, ensuring the desired orthogonality condition. This process is illustrated in Listing **??**.

We base our implementation on these equations. The SCF equation (26) is implemented in Python, as demonstrated in Listing 17:

Listing 17. Python implementation of the SCF equation.

```
VPhi = V_nuc(Phi_n) + 2J_n(Phi_n) - K_n(Phi_n)
Phi_np1 = -2G(VPhi + (Lambda_n - F_n) @ Phi_n)
```

This approach results in a direct analogue to the original mathematical formulation. Here, we operate on a `NumPy` vector of function trees representing our orbital vector ■. The functions `V_nuc`, `J_n`, and `K_n` correspond to the nuclear potential, Coulomb operator, and exchange operator, respectively. These operators (defined in equations (**??**), (**??**), and (**??**)) are implemented in a vectorized manner using `NumPy`. For example, the CouluombOperator class is presented in Listing 18: This pythonic implementation leverages `NumPy`'s vectorization capabilities to handle function tree objects efficiently, thereby enabling straightforward, readable implementations.

Listing 18. Python implementation of the Couloumb operator.

```
class CouloumbOperator():
    def __init__(self, mra, Psi, prec):
        self.mra = mra
        self.Psi = Psi
        self.prec = prec
        self.poisson = vp.PoissonOperator(mra=mra, prec=self.prec)
        self.potential = None
        self.setup()
    def setup(self):
        rho = self.Psi[0]**2
        for i in range(1, len(self.Psi)):
            rho += self.Psi[i]**2
        rho.crop(self.prec)
        self.potential = (4.0*np.pi)*self.poisson(rho).crop(self.prec)
    def __call__(self, Phi):
        return np.array([(self.potential*phi).crop(self.prec) for phi in Phi])
```

B. Solvation through the Polarizable Continuum model

PCM has been used to represent solvent effects for over fifty years and has proven to be an effective way to compute electrostatic effects in solvation[28]. This is because the degrees of freedom of the solvent are reduced as no solvent molecule is explicitly included in the computation. Instead the solvent is represented by its relative permittivity ε_r, which can be parameterized from its bulk properties.

The solvation energy E_R is computed from the solute-solvent interaction reaction potential, often just called a reaction potential V_R, and the total solute charge density ρ_{sol}

$$E_R = \frac{1}{2}\int d\mathbf{x} V_R(\mathbf{x})\rho_{sol}(\mathbf{x}). \tag{28}$$

The reaction potential V_R arises from the surface charge distribution $\gamma_s(\mathbf{x})$ induced at a boundary between the bulk solvent and the solute. This boundary is called a cavity, and it encapsulates the whole solute. The position dependent permittivity, $\varepsilon(\mathbf{r})$, takes the value of the permittivity of free space ε_0 inside this boundary while on the outside it takes the value of the relative permittivity of the solvent ε_r.

We compute the reaction potential by repeated application of the Poisson operator $P(\mathbf{x})$

$$V_R(\mathbf{x}) = P(\mathbf{x}) \star \left[\rho_{eff}(\mathbf{x}) + \gamma_s(\mathbf{x})\right] \tag{29}$$

where the effective density ρ_{eff} is defined as

$$\rho_{eff}(\mathbf{x}) = 4\pi \left(\frac{\rho_{sol}(\mathbf{x})}{\varepsilon(\mathbf{x})} - \rho(\mathbf{r}) \right) \tag{30}$$

For further details on the equations and variables, the reader is directed to Gerez S. *et al.*[29] where a thorough study of theory and implementation has been done.

1. Practical Implementation of Solvation

The example in listing 19 shows how PCM has been implemented using VAMPyR. Here we have assumed we already have a projected permittivity function and the total solute density function together with a suitable mra and an function that computes γ_s called computeGamma. Look up on the VAMPyR documentation page for a hands-on example of this implementation.

C. The Dirac equation for one electron

The last of the three pieces we present is the Dirac equation for one electron:

$$\begin{pmatrix} (V + mc^2 - \varepsilon) & c(\sigma \cdot \mathbf{p}) \\ c(\sigma \cdot \mathbf{p}) & (V - \varepsilon - mc^2) \end{pmatrix} \begin{pmatrix} \psi^{\mathrm{L}} \\ \psi^{\mathrm{S}} \end{pmatrix} = 0 \tag{31}$$

where V is the external potential, ε is the energy eigenvalue, c is the speed of light, $\mathbf{p}$ is the momentum operator and σ is the vector collecting the three Pauli matrices. As shown by Blackledge and Babajanov[?] and later exploited by Anderson *et al.* [?], the Dirac equation can also be reformulated as an integral equation as:

$$\begin{pmatrix} \psi^{\mathrm{L}} \\ \psi^{\mathrm{S}} \end{pmatrix} = \frac{-1}{2mc^2} \begin{pmatrix} (\varepsilon + mc^2) & c(\sigma \cdot \mathbf{p}) \\ c(\sigma \cdot \mathbf{p}) & (\varepsilon - mc^2) \end{pmatrix} G^{\mu} * \left[V \begin{pmatrix} \psi^{\mathrm{L}} \\ \psi^{\mathrm{S}} \end{pmatrix} \right] \tag{32}$$

where G^{μ} is the bound-state Helmholtz kernel as in the non relativistic case, and $\mu = \sqrt{\frac{m^2c^4 - \varepsilon}{mc^2}}$. The above integral equation can be solved iteratively as for the non relativistic case.

Listing 19. Python implementation of PCM in VAMPyR.

```
# Initialize the poisson operator
P = vp.PoissonOperator(mra=mra, prec=1.0e-5)
# Solve the standard Poisson equation
V_vac = P(4*pi*density)
# Compute the initial gamma_s
gamma_s = computeGamma(V_vac)
#  Construct effective density rho_eff
rho_eff = (4 * np.pi) * ((density * (permittivity)**(-1)) - density)
# Compute the initial reaction potential
V_R = P(rho_eff + gamma_s)
# Solve the GPE by iteration
for i in range(100):
    V = V_vac + V_R
    gamma_s = computeGamma(V, permittivity)
    V_R_np1 = P(rho_eff + gamma_s)
    dV_R_n = V_R_np1 - V_R
    update = dV_R_n.norm()
    V_R = V_R_np1.deepCopy()
    if (abs(update) <= 1.0e-5):
        break
# compute the final energy and finish.
E_R = (1/2)*vp.dot(V_R, rho)
```

1. *Practical implementation of the Dirac Equation*

The algorithm to solve the Dirac equation is in its general traits identical to the non-relativistic case. The main difference is that the infrastructure required to deal with 4-component spinors needs to be implemented.

Our design is built essentially on two Python classes: one to deal with complex functions (a single component of a spinor is a complex function) and another one to deal with the four components. Thanks to the design of `VAMPyR` those classes are easy to implement and use, for the most part overloading *dunder* operators.

The listing**??** shows a portion of the class implementation for the 4-component spinor. The objects are made callable and work as `NumPy` arrays. Each component is itself a complex function object (defined in a separate class). Some *dunder* methods are shown and the methods to perform the operation $c\alpha \cdot p\psi$ are also reported.

With the spinor class implementation at hand, the iterative scheme to converge the Hydrogen atom is easily implemented:

First the energy is computed in order to obtain the parameter μ, then comes the convolution of $V\psi$ with the Helmholtz kernel G^{μ}, and finally the Dirac Hamiltonian plus the energy $h_D + \varepsilon$ is applied to the convolution. The iteration is repeated until the norm of the spinor update is below the requested threshold. At the end of the iteration the energy must be computed once more (not reported in the listing).

D. Time dependent Schrödinger equation

Here we demonstrate how the multiwavelets machinery can be exploited for time evolution simulations. The main equation of interest is the following

$$i\partial_t \Psi = \left(\hat{T} + \hat{V}\right)\Psi, \tag{33}$$

where the potential $\hat{V}$ may depend on time in general. This equation is complemented by a given initial wave function $\Psi(x,0) = \Psi_0(x)$. Consideration of time-dependent potentials does not make the problem more difficult to solve, and so in this paper it won't be utilized for simplicity of presentation. Moreover, we will focus on one dimensional problem, so $V = V(x)$ and $\Psi = \Psi(x,t)$ with $x,t \in \mathbb{R}$.

A usual way of numerical treatment of (33) is to choose a small time step $t > 0$ and construct the time evolution operator on interval $[0,t]$. Then applying it iteratively one can arrive to the solution at some given finite time moment, thanks to the semigroup property of the propagator. This propagator can be written down in the very simple terms

$$\Psi(t) = \exp\left(-it\left(\hat{T} + \hat{V}\right)\right)\Psi_0, \tag{34}$$

where we used the fact that the potential is time-independent. Our aim is to construct this exponential operator at least for small time step $t > 0$. As we normally do not have an access to the full collection of eigenfunctions and eigenvalues for the Hamiltonian, we have to split the propagator on kinetic $\exp\left(-it\hat{T}\right)$ and potential $\exp\left(-it\hat{V}\right)$ parts. Note that the latter is a multiplication operator in the physical space, whereas the first one is a multiplication operator in the momentum space. Such splitting leads to an additional source of error, since the kinetic and potential energy operators do not commute. Ignoring this, one obtains a time evolution numerical scheme of the first order

$$\exp\left(-it\left(\hat{T}+\hat{V}\right)\right) = \exp\left(-it\hat{T}\right)\exp\left(-it\hat{V}\right) + \mathscr{O}\left(t^2\right)$$

that is the simplest possible splitting, too rough though for practical applications. We will make use of the following fourth order scheme[30]

$$e^{At+Bt} = \exp\left(\frac{t}{6}B\right)\exp\left(\frac{t}{2}A\right)\exp\left(\frac{2t}{3}\widetilde{B}\right)\exp\left(\frac{t}{2}A\right)\exp\left(\frac{t}{6}B\right) + \mathscr{O}\left(t^5\right), \tag{35}$$

where

$$\widetilde{B} = B + \frac{t^2}{48}[B,[A,B]]. \tag{36}$$

In the case of $A = -i\hat{T}$ and $B = -i\hat{V}$ this $\widetilde{B}$ turns out to be a multiplication operator containing potential gradient $\partial_x V(x)$. Remarkably, this high order scheme requires only two applications of the free-particle semigroup operator $\exp\left(-it\hat{T}/2\right)$ per time step. It is also very easy to implement:

```
class ChinChenA(object):
    def __init__(self, expA, expB, exp_tildeB):
        self.expA = expA
        self.expB = expB
        self.exp_tildeB = exp_tildeB
    def __call__(self, u):
        u = self.expB(u)         # 1/6*dt
        u = self.expA(u)         # 1/2*dt
        u = self.exp_tildeB(u)   # 2/3*dt
        u = self.expA(u)         # 1/2*dt
        u = self.expB(u)         # 1/6*dt
        return u
```

Now we can conclude that for an effective simulation of small quantum systems it is enough to have a good numerical presentation of $\exp\left(it\partial_x^2\right)$, where $t > 0$ is normally taken less than a time step. We remind that the exponent operator forms a semigroup representing solutions of the free particle equation

$$i\partial_t \Psi + \partial_x^2 \Psi = 0.$$

In other words, for any $\Psi_0 \in L^2(\mathbb{R})$ function $\Psi = \exp\left(it\partial_x^2\right)\Psi_0$ solves this equation. This is a convolution operator with the kernel

$$K(x,y) = \frac{\exp(-i\pi/4)}{\sqrt{4\pi t}} \exp\left(\frac{i(x-y)^2}{4t}\right),$$

and so one anticipates that the machinery described above would work here as well. However, in practice it turns out to be difficult to discretise it in a similar manner without damping the high frequencies of Green's function[31]. We developed a completely different approach to this discretisation problem[32].

1. Practical implementation of the Schrödinger semigroup

The detailed theory behind algorithm in use follows in a separate upcoming publication[32]. Here we present only the working formulas encoded in MRCPP. Let $\left[\sigma^n_{l'-l}\right]_{j'j}(t)$ stay for matrix elements of the time evolution operator $P^n \exp\left(it\partial_x^2\right) P^n$ at scale n with respect to the Legendre scaling basis $\varphi^n_{j,l}(x)$. Then

$$[\sigma^n_l]_{pj}(t) = \sum_{k=0}^{\infty} C^{2k}_{jp} J_{2k+j+p}(l, 4^n t), \tag{37}$$

where

$$J_m(l,a) = \frac{e^{i\frac{\pi}{4}(m-1)}}{2\pi(m+2)!} \int_{\mathbb{R}} \exp\left(\rho l \exp\left(i\frac{\pi}{4}\right) - a\rho^2\right) \rho^m d\rho \tag{38}$$

satisfying the following relation

$$J_{m+1} = \frac{il}{2a(m+3)} J_m + \frac{im}{2a(m+2)(m+3)} J_{m-1}, \quad m = 0, 1, 2, \ldots, \tag{39}$$

with the agreement $J_{-1} = 0$ and

$$J_0 = \frac{e^{-i\frac{\pi}{4}}}{4\sqrt{\pi a}} \exp\left(\frac{il^2}{4a}\right). \tag{40}$$

These power integrals depend on the time step parameter $t > 0$, whereas the coefficients C^{2k}_{jp} are problem-independent and can be calculated once as

$$C^k_{jp} = \sum_{m=0}^{j} \sum_{q=0}^{p} \frac{(-1)^{m+1}(k+2+j+p)!}{(k+2+j+p+m+q)!} \left(A^j_m B^p_q + (-1)^{k+j+p+m+q} B^j_m A^p_q\right).$$

The coefficients appearing here under the double sum may be found as follows

$$A_0^1 = \sqrt{3}, \quad A_1^1 = 2\sqrt{3}, \quad B_0^1 = \sqrt{3}, \quad B_1^1 = -2\sqrt{3}.$$

For $j \geqslant 1$ we have the following relation

$$\begin{aligned}
A_0^{j+1} &= \sqrt{\frac{2j+3}{2j-1}} A_0^{j-1} \\
A_1^{j+1} &= \sqrt{\frac{2j+3}{2j-1}} A_1^{j-1} - 2\sqrt{(2j+1)(2j+3)} A_0^j \\
& \dots \quad \dots \quad \dots \quad \dots \quad \dots \\
A_{j-1}^{j+1} &= \sqrt{\frac{2j+3}{2j-1}} A_{j-1}^{j-1} - 2\sqrt{(2j+1)(2j+3)} A_{j-2}^j \\
A_j^{j+1} &= -2\sqrt{(2j+1)(2j+3)} A_{j-1}^j \\
A_{j+1}^{j+1} &= -2\sqrt{(2j+1)(2j+3)} A_j^j
\end{aligned}$$

and B_m^j obey the same recurrence for $j \geqslant 1$.

The MRCPP implementation of the time evolution operator $\exp\left(it\partial_x^2\right)$ is under optimization currently, though it is already available in VAMPyR:

```python
def create_unitary_kinetic_operator(mra, precision, time, finest_scale):
    real = vp1.TimeEvolutionOperator(mra, precision, time, finest_scale, False)
    imag = vp1.TimeEvolutionOperator(mra, precision, time, finest_scale, True)
    return UnitaryExponentGroup(real, imag)
```

VAMPyR is using real-space techniques. So if we associate with each complex-valued function a vector consisting of two real functions, then the semigroups under consideration will have the following common form

$$\exp\left(-i\widehat{H}t\right) = \begin{pmatrix} \cos\widehat{H}t & \sin\widehat{H}t \\ -\sin\widehat{H}t & \cos\widehat{H}t \end{pmatrix}, \quad \text{operating on vector-functions } \Psi(t) = \begin{pmatrix} u(t) \\ v(t) \end{pmatrix}.$$

Here self-adjoint $\widehat{H}$ can stand for the kinetic energy $-\partial_x^2$, time independent potential $V(x)$ or Hamiltonian $-\partial_x^2 + V(x)$, which one incorporates in the code:

```
class UnitaryExponentGroup(object):
    def __init__(self, real, imag):
        self.real = real
        self.imag = imag
    def __call__(self, psi):
        u = psi[0]
        v = psi[1]
        res0 = self.real(u) - self.imag(v)
        res1 = self.imag(u) + self.real(v)
        return np.array([ res0, res1 ])
```

It finishes the description of the unitary exponential operator associated with the kinetic energy. In order to encode the unitary group associated with the potential energy, we define the general multiplication operator:

```
class MultiplicationOperator(object):
    def __init__(self, function):
        self.function = function
    def __call__(self, function):
        return self.function * function
```

Now the tree structure of the potential semigroup $\exp(-itV)$ is introduced:

```
def create_unitary_potential_operator(P, V, t):
    def real(x):
        return np.cos(V(x) * t)
    real = P(real)
    def imag(x):
        return - np.sin(V(x) * t)
    imag = P(imag)
    real = MultiplicationOperator(real)
    imag = MultiplicationOperator(imag)
    return UnitaryExponentGroup(real, imag)
```

Together with a splitting scheme, for example (35), this completes the algorithm description for the time evolution simulations.

The above algorithm in work is demonstrated on a very simple example of the Gaussian wave packet

$$\Psi_0(x) = \left(\frac{1}{2\pi\sigma^2} \right)^{1/4} \exp\left(-\frac{(x-x_0)^2}{4\sigma^2} \right)$$

evolving with time in the harmonic potential $V(x) = V_0(x-x_1)^2$. The equation under consideration is (33) with $\hat{T} = -\partial_x^2$. It is well known that the density $|\Psi(t)|^2$ oscillates in the harmonic potential with the period $t_{\text{period}} = \pi/\sqrt{V_0}$. More precisely, $\Psi\left(t_{\text{period}}\right) = -\Psi_0$. This can immediately be seen taking into account that the eigenvalues for the Hamiltonian are $\sqrt{V_0}(2n+1)$. We take $x_0 = 0.3$, $\sigma = 0.04$ and $x_1 = 0.5$, $V_0 = 25000$. All the parameters are chosen in a way that the solution stays localized mainly on the space interval $[0,1]$. Note that $\widetilde{B}$ being used in (35) simplifies to

$$\widetilde{B} = -i\widetilde{V} = -iV + \frac{it^2}{24}(\partial_x V)^2 = -i\widetilde{V}_0(x-x_0)^2, \quad \text{where } \widetilde{V}_0 = V_0 - \frac{(tV_0)^2}{6},$$

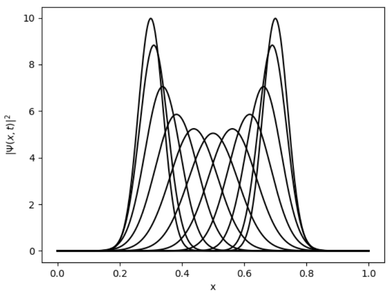
according to (36). Thus as we have already mentioned above, $\widetilde{B}$ and so the corresponding exponent in (35) are multiplication operators.

In Listing 22 we define all the necessary operators and initialize Scheme (35). Actual simulations are conducted in Listing 23. Figure 6 demonstrates the results of these calculations: the oscillation movement of the density $|\Psi(t)|^2$ and the difference between the numerical and exact solution at the time moment $t = t_{\text{period}}$.

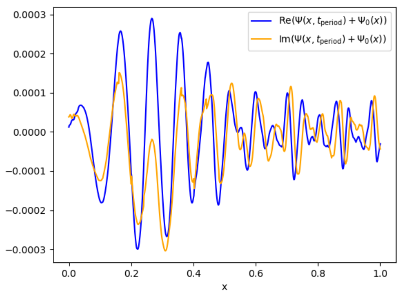
VI. INTEROPERABILITY WITH OTHER PACKAGES

One of the main benefits of introducing a Python interface is the seamless interoperability it offers with a vast number of other useful packages. This interoperability not only simplifies the usage of the package but also enhances its capabilities by allowing the integration of features provided by other packages. In this section, we will explore some potential applications of this interoperability, showcasing how VAMPyR can interact with other packages to efficiently perform tasks that may otherwise be computationally demanding or require complex implementation.

The power of interoperability has already been illustrated through our previous examples,



(a) Density evolution in the harmonic potential.



(b) The difference between the numerical and exact solution.

Figure 6. Simulation of (33) in `VAMPyR`.

where the `NumPy` and `Matplotlib` packages were used in tandem with `VAMPyR` for various computational and visualization tasks. However, we can push the envelope even further.

As an example, the Self-Consistent Field (SCF) solver implemented in `VAMPyR`, though fully general, encounters practical limitations with more complex molecular systems. One primary challenge is the selection of an appropriate initial guess for the orbitals; a poor choice can severely impact the convergence rate and even result in a failure to locate the global energy minimum. The first step beyond the Hartree-Fock (HF) approximation might be to incorporate exchange and correlation density functionals, which, while beneficial, can be tedious to implement.

Here is where interoperability steps in. We can leverage the capabilities of different Python packages to address these issues. We will walk through examples demonstrating how `VAMPyR` can:

- Use VeloxChem to generate an initial guess for a SCF, speeding up the process and increasing the chance of successful convergence.
- Cooperate with VeloxChem to compute a potential energy surface grid, effectively utilizing computational resources.
- Perform stability analysis on Density Functional Theory (DFT) functionals from Libxc, simplifying the incorporation of exchange and correlation effects.

These examples, while not exhaustive, demonstrate the potential for integrating `VAMPyR` with other Python-based scientific computing packages, creating an efficient and powerful computational toolset.

A. Generating an Initial Guess with VeloxChem

The efficiency of SCF calculations can be significantly improved with a suitable initial guess. A common approach to obtaining this is to utilize electronic structure packages like `VeloxChem`. As both `VeloxChem` and `VAMPyR` provide Python interfaces, they can seamlessly interoperate, exemplifying the benefits of using Python for scientific software. A suitable initial guess not only enhances the chances of reaching convergence but also accelerates the convergence rate, as we have seen in the earlier examples with suboptimal starting guesses (see Section V).

Here, we illustrate how to generate an initial guess using `VeloxChem` that can be imported into `VAMPyR`. Reconstructing the Molecular Orbitals (MOs) (or electron density) from the AO basis set using the MO (or density) matrix is a rather complex task, especially if the basis contains high angular momentum functions. We thus want to make use of VeloxChem's own internal evaluator for these objects, and wrap a simple function around it which can be projected onto the MW basis using a ScalingProjector. In principle, any $\mathbb{R}^3 \to \mathbb{R}$ function can be projected in this way, so we just have to define a function that takes as argument a point in real-space, runs it through VeloxChems internal AO evaluater code, and returns the function value of a given MO.

First, we read the molecular structure and basis set:

```
# Read molecule and basis
molecule = vlx.Molecule.read_str(mol_str)
basis = vlx.MolecularBasis.read(molecule, "PCSEG-1")
```

Next, we create an instance of the ScfRestrictedDriver and run the SCF calculation:

```
# Make SCF and run driver
scf_drv = vlx.ScfRestrictedDriver()
scf_results = scf_drv.compute(molecule, basis)
```

To extract the molecular orbitals (MOs) from the SCF calculation, we need to use VeloxChem's VisualizationDriver:

```python
# Make a visualization driver that can be used to evaluate orbitals.
vis_drv = vlx.VisualizationDriver()


# Get the MOs
mol_orbs = scf_drv.mol_orbs
```

We can define a function, orb(r), that returns the value of the MO at a given point in space. This function will be used for projecting onto an MRA in `VAMPyR`:

```python
# Define a function to get the MOs, this can be projected onto an MRA by vampyr
def mo_i(r):
    R = np.array([r])
    return vis_drv.get_mo(R, molecule, basis, mol_orbs, i, "alpha")[0]
```

Finally, we use a projector from `VAMPyR` to project the MOs from VeloxChem onto an MRA, generating a list of `FunctionTrees` for the initial guess:

```python
# This can now be imported into vampyr as:


P_eps = vp.ScalingProjector(mra, prec)
Phi_0 = [P_eps(mo_i) for i in range(nr_orbs)]
```

In the last step, we loop over the desired number of orbitals, project each one onto the MRA, and store the result in `Phi_0`. This list of `FunctionTrees` represents an approximation to the molecular orbitals of the system, and serves as an excellent initial guess for the SCF procedure in `VAMPyR`. This approach demonstrates how we can leverage the strengths of `VeloxChem` and `VAMPyR` together, creating efficient workflows in quantum chemistry calculations.

B. Calculate a potential energy surface with VeloxChem and `VAMPyR`

A Potential Energy Surface (PES) can be computed as a series of single point energy calculations while varying one (or more) bond distance(s) in a molecule. We will here use `VAMPyR`'s adaptive function projector as a driver for computing the PES of the Hydrogen molecule using

VeloxChem's single-point Hartree–Fock evaluator. We define a Python function, `pes(r)`, that computes the energy for two Hydrogen atoms given the bond distance as input:

```python
def pes(r):
    mol_str = f"""
    H       0.0  0.0000   -0.2
    H       0.0  0.0000   {r[0]}
    """
    molecule = vlx.Molecule.read_str(mol_str)
    basis = vlx.MolecularBasis.read(molecule, "PCSEG-1")

    scf_drv = vlx.ScfUnrestrictedDriver()
    scf_results = scf_drv.compute(molecule, basis)
    return scf_drv.scf_energy
```

This function generates a diatomic Hydrogen molecule at a given bond distance (+0.2 bohrs to avoid a singularity, since r will be evaluated between 0 and 10), performs an SCF calculation at the given geometry, and returns the computed energy. This function, being a $\mathbb{R} \to \mathbb{R}$ mapping, can be projected on a 1D MRA using `VAMPyR`.

```python
mra = vp.MultiResolutionAnalysis(box=[0, 10], order=1)
P_eps = vp.ScalingProjector(mra, prec=1.0e-3)
pes_tree = P_eps(pes)
```

Here the adaptive projector will automatically sample the `pes(r)` function in appropriate points in order to produce a smooth surface. This approach effectively focuses computational resources where they are needed most, as determined by the adaptive projection algorithm. The potential energy surface obtained in this manner can be visualized, as shown in Figure 7.

It's important to note that the grid lines in Figure X represent the adaptive MRA grid, and the potential energy surface depicted is interpolated from this grid rather than computed directly at each point. This approach emphasizes the efficacy of the adaptive grid algorithm in efficiently focusing computational resources. It also underscores the power of combining Python packages like `VeloxChem` and `VAMPyR` to efficiently and accurately model complex quantum systems.

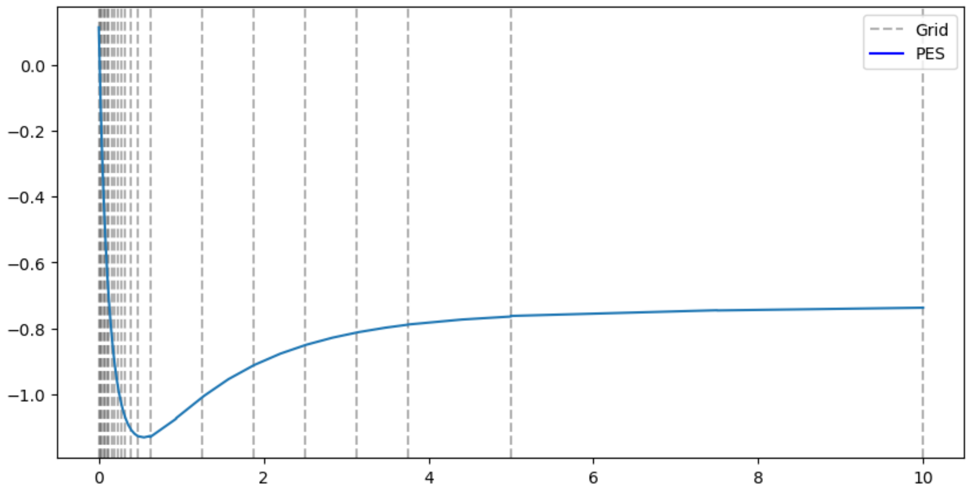


Figure 7. Potential energy surface calculated using `VeloxChem` and `VAMPyR`. The grid lines represent the adaptive MRA grid, from which the PES is interpolated.

C. Numerical Stability Analysis using `VAMPyR` and PylibXC

Inspired by the paper by Lehtola and Marques[33], we investigate the numerical stability of density functional approximations (DFAs) lda-c-vwn and lda-c-gk72, with `VAMPyR` and PylibXC. The choice of these DFAs is due to their contrasting convergence behavior, making them interesting subjects for our analysis. The lda-c-gk72, developed by Gordon and Kim in 1972, was known for its problematic convergence while the lda-c-vwn (Vosko, Wilk, and Nusair functional) was noted for its numerical stability. Our objective is to verify these claims, illustrating the potential of `VAMPyR` in detailed and precise computational quantum chemistry.

Firstly, we initialize a Neon molecule and calculate the self-consistent field (SCF) using `VeloxChem`:

```
def init_molecule_and_scf(mol_str, basis_set="PCSEG-1"):
    molecule = vlx.Molecule.read_str(mol_str)
    basis = vlx.MolecularBasis.read(molecule, basis_set)
    scf_drv = vlx.ScfRestrictedDriver()
    scf_results = scf_drv.compute(molecule, basis)
    return molecule, basis, scf_drv.density
```

Next, we calculate the density and project it onto a Multi-Resolution Analysis (MRA) grid:

```
def calc_density(r, molecule, basis, mol_orbs, vis_drv):
    R = np.array([r])
    rho_a = vis_drv.get_density(R, molecule, basis, mol_orbs, "alpha")[0] +
    rho_b = vis_drv.get_density(R, molecule, basis, mol_orbs, "beta")[0]
    return rho_a + rho_b


rho = P_eps(lambda r: calc_density(r, molecule, basis, mol_orbs, vis_drv))
```

Subsequently, we use PylibXC to define the chosen DFA functionals:

```
func_c = xc.LibXCFunctional("lda_c_gk72" or "lda_c_vwn", "unpolarized")
def f_e(n):
    c = func_c.compute({"rho": n})["zk"]
    return n * (c)
F_e = vp.FunctionMap(f_e, precision)
```

Following this, an exchange potential is generated. We iteratively refine the grid based on precision, one scale at a time, until no new nodes are created:

```
v_x = vp.FunctionTree(mra)
nNodes = v_x.nNodes()
while nNodes > 0:
    vp.advanced.clear_grid(v_x)
    vp.advanced.map(-1.0, v_x, rho, f_e)
    print(f"nodes : {v_x.nNodes()} norm : {v_x.norm()}", flush=True)
    nNodes = vp.advanced.refine_grid(v_x, precision)
```

Observing the output for each DFA, we note: If we look at the printout of the number of nodes we see it converging for the lda_c_vwn, no exponential growth in the number of nodes. For

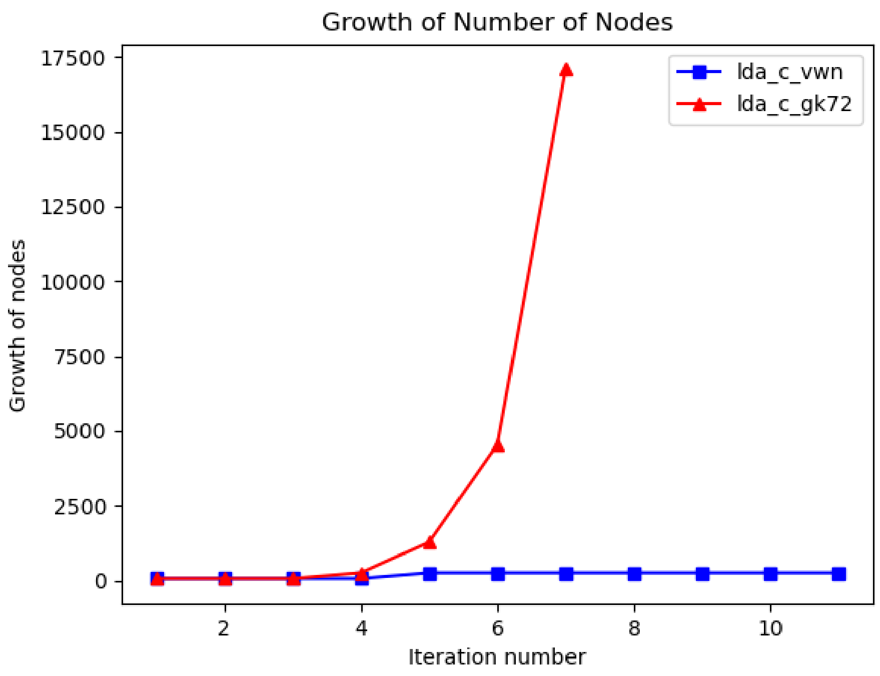


Figure 8. Comparison of node growth between the lda-c-vwn and lda-c-gk72 DFAs. The plot illustrates the sharp contrast in the convergence behaviour, with lda-c-vwn showing a stable, linear growth, and lda-c-gk72 displaying an exponential growth.

`lda_c_gk72` We see explosive growth in the number of nodes. So in this case we stopped the calculation since it didn't converge in a reasonable time frame. Looking at the norm we see that we're no closer to convergene. In this example we went for a precision of $1.0e-8$

REFERENCES

[1]R. J. Harrison, G. I. Fann, T. Yanai, Z. Gan, and G. Beylkin, J. Chem. Phys. **121**, 11587 (2004).

[2]T. Yanai, G. I. Fann, Z. Gan, R. J. Harrison, and G. Beylkin, J. Chem. Phys. **121**, 6680 (2004).

[3]T. Yanai, G. I. Fann, Z. Gan, R. J. Harrison, and G. Beylkin, J. Chem. Phys. **121**, 2866 (2004).

[4]T. Yanai, R. J. Harrison, and N. C. Handy, Mol. Phys. **103**, 413 (2005).

[5]A. Brakestad, P. Wind, S. R. Jensen, L. Frediani, and K. H. Hopmann, J. Chem. Phys. **154**,

214302 (2021).

[6]S. R. Jensen, J. Jusélius, A. Durdek, T. Flå, P. Wind, and L. Frediani, Int. J. Model. Simul. Sci. Comput. **05**, 1441003 (2014).

[7]S. R. Jensen, T. Flå, D. Jonsson, R. S. Monstad, K. Ruud, and L. Frediani, Phys. Chem. Chem. Phys. **18**, 21145 (2016).

[8]S. R. Jensen, S. Saha, J. A. Flores-Livas, W. Huhn, V. Blum, S. Goedecker, and L. Frediani, J. Phys. Chem. Lett. **8**, 1449 (2017).

[9]Q. Pitteloud, P. Wind, S. R. Jensen, L. Frediani, and F. Jensen, J. Chem. Theory Comput. (2023), 10.1021/acs.jctc.3c00693.

[10]R. J. Harrison, G. Beylkin, F. A. Bischoff, J. A. Calvin, G. I. Fann, J. Fosso-Tande, D. Galindo, J. R. Hammond, R. Hartman-Baker, J. C. Hill, J. Jia, J. S. Kottmann, M.-J. Yvonne Ou, J. Pei, L. E. Ratcliff, M. G. Reuter, A. C. Richie-Halford, N. A. Romero, H. Sekino, W. A. Shelton, B. E. Sundahl, W. S. Thornton, E. F. Valeev, A. Vázquez-Mayagoitia, N. Vence, T. Yanai, and Y. Yokoi, SIAM J. Sci. Comput. **38**, S123 (2016).

[11]P. Wind, M. Bjørgve, A. Brakestad, G. A. Gerez S, S. R. Jensen, R. D. R. Eikås, and L. Frediani, Journal of Chemical Theory and Computation **19**, 137 (2022).

[12]C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, Nature **585**, 357 (2020).

[13]P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, Nature Methods **17**, 261 (2020).

[14]J. D. Hunter, Computing in Science & Engineering **9**, 90 (2007).

[15]T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt (IOS Press, 2016) pp. 87 – 90.

[16]R. van der Pas, E. Stotzer, and C. Terboven, *Using OpenMP—The Next Step: Affinity, Acceler-*

ators, Tasking, and SIMD, Scientific and engineering computation series (MIT Press, 2017).

[17]T. G. Mattson, Yun, and A. E. Koniges, *The OpenMP Common Core: Making OpenMP Simple Again* (The MIT Press, 2019).

[18]Note that Python user-defined functions cannot be represented leveraging shared-memory parallelism due to the infamous global interpreter lock (GIL).

[19]B. K. Alpert, SIAM journal on Mathematical Analysis **24**, 246 (1993).

[20]B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi, Journal of Computational Physics **182**, 149 (2002).

[21]"Mrchemsoft on github," (2023), accessed: September.

[22]"Mrcpp on conda-forge," (2023), accessed: September.

[23]"Vampyr on conda-forge," (2023), accessed: September.

[24]It should be noted that some performance-oriented functionalities, as well as specific operators like Helmholtz and Poisson, are currently exclusive to 3-dimensional problems. While extensions to lower or higher dimensions are feasible, such efforts have not yet been undertaken.

[25]D. Vandevoorde, N. M. Josuttis, and D. Gregor, *C++ Templates: The Complete Guide*, 2nd ed. (Addison-Wesley, 2018).

[26]J. Anderson, R. J. Harrison, H. Sekino, B. Sundahl, G. Beylkin, G. I. Fann, S. R. Jensen, and I. Sagert, Journal of Computational Physics: X **4**, 100033 (2019).

[27]F. Jensen, *Introduction to computational chemistry* (John wiley & sons, 2017).

[28]J. Tomasi, B. Mennucci, and R. Cammi, Chemical Reviews **105**, 2999 (2005), pMID: 16092826, https://doi.org/10.1021/cr9904009.

[29]G. A. Gerez S., R. Di Remigio Eikås, S. R. Jensen, M. Bjørgve, and L. Frediani, Journal of Chemical Theory and Computation **19**, 1986 (2023), pMID: 36933225, https://doi.org/10.1021/acs.jctc.2c01098.

[30]S. A. Chin and C. R. Chen, The Journal of Chemical Physics **114**, 7338 (2001), https://pubs.aip.org/aip/jcp/article-pdf/114/17/7338/10831663/7338_1_online.pdf.

[31]N. Vence, R. Harrison, and P. Krstić, Phys. Rev. A **85**, 033403 (2012).

[32]E. Dinvay and L. Frediani, In preparation for submission.

[33]S. Lehtola and M. A. Marques, The Journal of Chemical Physics **157** (2022).

Listing 20. Excerpt of the 4-component spinor class.

```python
class spinor:
    """Four components orbital."""
    mra = None
    light_speed = -1.0
    comp_dict = {'La': 0, 'Lb': 1, 'Sa': 2, 'Sb': 3}
    def __init__(self):
        self.comp_array = np.array([cf.complex_fcn(),
                                    cf.complex_fcn(),
                                    cf.complex_fcn(),
                                    cf.complex_fcn()])
    def __getitem__(self, key):
        return self.comp_array[self.comp_dict[key]]
    def __setitem__(self, key, val):
        self.comp_array[self.comp_dict[key]] = val
    def __len__(self):
        return 4
    def __add__(self, other):
        output = orbital4c()
        output.comp_array = self.comp_array + other.comp_array
        return output
    def __call__(self, position):
        return [x(position) for x in self.comp_array]
    def __rmul__(self, factor):
        output = orbital4c()
        output.comp_array =  factor * self.comp_array
        return output
    def alpha(self, direction, prec):
        out_orb = orbital4c()
        alpha_order = np.array([[3, 2, 1, 0],
                                [3, 2, 1, 0],
                                [2, 3, 0, 1]])
```

Listing 21. Iterative solution of the Dirac equation.

```python
while orbital_error > prec:

    hd_psi = orb.apply_dirac_hamiltonian(spinor_H, prec, der = default_der)

    v_psi = orb.apply_potential(-1.0, V_tree, spinor_H, prec)

    add_psi = hd_psi + v_psi

    energy = (spinor_H.dot(add_psi)).real

    mu = orb.calc_dirac_mu(energy, light_speed)

    tmp = orb.apply_helmholtz(v_psi, mu, prec)

    tmp.crop(prec/10)

    new_orbital = orb.apply_dirac_hamiltonian(tmp, prec, energy, der = default_der)

    new_orbital.crop(prec/10)

    new_orbital.normalize()

    delta_psi = new_orbital - spinor_H

    orbital_error = (delta_psi.dot(delta_psi)).real

    print('Error',orbital_error, imag, flush = True)

    spinor_H = new_orbital
```

Listing 22. Preparation to implementation of time evolution in VAMPyR.

```
# Define parameters, final time moment and time step

x0 = 0.3

sigma = 0.04

x1 = 0.5

V0 = 25000

N = 20

t_period = np.pi / np.sqrt(V0)

time_step = t_period / N

# Set the precision and make the MRA

precision = 1.0e-5

finest_scale = 9

mra = vp1.MultiResolutionAnalysis(vp1.BoundingBox(0), LegendreBasis(5))

# Make the scaling projector

P = vp1.ScalingProjector(mra, prec = precision)

# Define the harmonic potential with its scheme modification

def V(x):

    return V0 * (x[0] - x1)**2

def tilde_V(x):

    A = V0 - ( time_step * V0 )**2 / 6.0

    return A * (x[0] - x1)**2

# Define the iteration procedure

iteratorA = ChinChenA(

    expA = create_unitary_kinetic_operator(mra, precision, time_step / 2, finest_scale),

    expB = create_unitary_potential_operator(P, V, time_step / 6),

    exp_tildeB = create_unitary_potential_operator(P, tilde_V, 2 * time_step / 3)

)
```

Listing 23. Time evolution simulation in `VAMPyR`.

```
# Define the initial wave function
psi0 = vp1.GaussFunc(
    beta = 1.0 / (4 * sigma**2), alpha = (2 * np.pi * sigma**2)**(-1/4), position = [x0]
)
psi0 = np.array([ P(psi0), vp1.FunctionTree(mra).setZero() ])
# Solve the initial value problem
psiA = psi0
for n in range(N):
    psiA = iteratorA(psiA)
# Find error at t = period
per_errorA = psiA + psi0
print( f"L2-norm of real part error:      {per_errorA[0].norm()}" )
print( f"L2-norm of imaginary part error: {per_errorA[1].norm()}" )
```

5.2 Paper II: Kinetic Energy-free Hartree–Fock equations: an integral formulation

Abstract

A novel Self-Consistent Field (SCF) solver tailored for Hartree–Fock calculations harnesses the power of Multiwavelets and Multiresolution Analysis, presenting a robust preconditioned steepest descent framework characterized by swift convergence. Notably, our implementation omits the kinetic energy operator, a critical adaptation when employing Multiwavelets due to the complexities in accurately representing differential operators, such as the Laplacian. The paper thoroughly expounds the theoretical underpinnings and delineates the algorithm, which is also shared as an executable Python notebook. We provide two rudimentary examples that showcase the efficacy of our approach, including its capacity for highly precise calculations, expedited and dependable convergence, and the notable omission of the kinetic energy operator.

Personal Contributions:

- Authored significant sections of the foundational theoretical explanations and results discussion in the paper's final version.
- Developed and implemented the Python notebooks that are appended to the paper, which detail the computational procedures.
- Generated all the figures included in the paper from the provided notebooks, underlining the implementation's reproducibility and effectiveness.

Journal of Mathematical Chemistry (2023) 61:343–361
https://doi.org/10.1007/s10910-022-01374-3

ORIGINAL PAPER

Kinetic energy-free Hartree–Fock equations: an integral formulation

Stig Rune Jensen[1] · **Antoine Durdek**[2] · **Magnar Bjørgve**[3] · **Peter Wind**[3] · **Tor Flå**[4] · **Luca Frediani**[3]

Received: 17 March 2022 / Accepted: 3 June 2022 / Published online: 18 July 2022

Abstract
We have implemented a self-consistent field solver for Hartree–Fock calculations, by making use of Multiwavelets and Multiresolution Analysis. We show how such a solver is inherently a preconditioned steepest descent method and therefore a good starting point for rapid convergence. A distinctive feature of our implementation is the absence of any reference to the kinetic energy operator. This is desirable when Multiwavelets are employed, because differential operators such as the Laplacian in the kinetic energy are challenging to represent correctly. The theoretical framework is described in detail and the implemented algorithm is both presented in the paper and made available as a Python notebook. Two simple examples are presented, highlighting the main features of our implementation: arbitrary predefined precision, rapid and robust convergence, absence of the kinetic energy operator.

Keywords Multiwavelets · Hartree–Fock · Self-consistent field · Real space methods

This article is dedicated to the memory of Prof. János Pipek. Prof. Pipek had in the latest years of his career turned his attention to Multiresolution Analysis and Wavelet methods. We honor his contribution to the field with a paper on the same subject.

✉ Luca Frediani
luca.frediani@uit.no

Stig Rune Jensen
stig.r.jensen@uit.no

1 Hylleraas Centre, IT-Department, UiT, The Arctic University of Norway, Tromsø, Norway

2 Department of Mathematics, UiT, The Arctic University of Norway, Tromsø, Norway

3 Hylleraas Centre, Department of Chemistry, UiT, The Arctic University of Norway, Tromsø, Norway

4 Hylleraas Centre, Department of Mathematics, UiT, The Arctic University of Norway, Tromsø, Norway

1 Introduction

Atom-centered Gaussians have traditionally been the most common and widespread choice of basis set for molecules [1]. Several strong arguments are in favor of such a choice: the compactness of the representation which is defined by a handful of coefficients, the ability to represent atomic orbitals well (Slater functions are in theory superior due to the cusp at the nuclear position and the correct asymptotic), the simplification in the computation of molecular integrals which are often obtained analytically (this is the weak point of Slater orbitals which require expensive numerical evaluations). Their main disadvantage is the non-orthogonality of the basis which can become a severe problem especially for large bases leading to a computational bottleneck when orthonormalization is required or worse numerical instabilities due to near linear-dependency in the basis [2].

On the opposite side of the spectrum, plane waves (PWs) are ideally suited for periodic systems and are orthonormal by construction. However a very large number of them needs to be employed in order to achieve good precision, especially if one is interested in high resolution in the nuclear-core regions [3]. A popular choice to circumvent the problem is to use pseudopotentials [4] in the core region, thereby reducing the number of electrons to be treated and at the same time removing the need for very high-frequency components. Lately, the use of projector augmented wave (PAW) [5] and linearized augmented plane wave (LAPW) [6] techniques, has made this issue less critical for PW calculations. Another challenge for PWs is constituted by non-periodic systems, which can only be dealt with by using a supercell approach [7].

Quantum chemical modeling is constantly expanding its horizons: cutting edge research is focused on achieving good accuracy (either in energetics or molecular properties) on large non-periodic systems such as large biomolecules or molecular nanosystems. This progression is constantly exposing the weaknesses of the traditional approach thus rendering the use of unconventional methods, which are free from the above mentioned limitations ever more attractive. One such choice is constituted by numerical, real-space grid-based methods which are gaining popularity in quantum chemistry as a promising strategy to deal with the Self Consistent Field (SCF) problem of Hartree Fock(HF) and density functional theory (DFT).

Among real-space approaches, three strategies have been commonly employed: Finite Differences [8], Finite Elements [9], Wavelets [10, 11] and Multiwavelets (MWs) [12]. MWs are particularly well suited for all-electron calculations [12, 13]. The basis functions are localized (as Gaussian-type orbitals) yet orthonormal (as plane waves). One crucial property of MWs is the disjoint support (zero overlap) between basis functions in adjacent nodes [14], paving the way for adaptive refinement of the mesh, tailored to each given function. This is essential for an all-electron description where varying resolution is a prerequisite for efficiency. The price to pay, to provide a representation with a given number of vanishing moments, is a basis consisting of several wavelet functions per node. The most common choice of basis functions in the MW framework is a generic orthonormal polynomial basis of order k, providing a second possibility to increase the resolution of the representation alongside the adaptive grid refinement [15]. Currently, the main drawbacks of this approach are a large memory footprint (a numerical representation of a molecular orbital is much larger in terms of number of coefficients), and a significant computational overhead [16, 17]. On the other hand, a localized orthonormal basis is an ideal match for modern massively-parallel architectures [18] and we are confident that it is only a matter of time before real-space grid methods in general and MWs in particular will become competitive with or even superior to traditional ones.

To achieve high precision and keep the memory footprint at a manageable level, an adaptive strategy which refines grids only if needed is necessary [19]. This choice has a profound impact on the minimization strategies that can be adopted in order to solve SCF problems such as the Roothaan–Hall equations of the Hartree–Fock (HF) method. In other words, strategies which rely upon having a fixed basis, such as the most common atomic orbital based methods [20] are excluded. On the other hand, only the occupied molecular orbitals are needed both in HF and DFT to describe the wavefunction/electronic density. Methods providing a *direct* minimization of the orbitals without requiring a fixed basis representation must be considered. Additionally, using MWs on an adaptive grid generates representations with discontinuities at the nodal surfaces, which poses a challenge when differential operators are considered. As will be shown in the paper, if the Hartree–Fock equations are reformulated as coupled integral equations, it becomes possible to minimize the occupied orbitals, without ever recurring to differential operators.

2 Functions and operators in the Multiwavelet framework

When defining the MW framework we think in terms of scaling spaces V^n and wavelet spaces W^n. The scaling space V^0 in 3D real-space is spanned by a set of orthogonal polynomials on the unit cube, and the spaces V^n for $n > 0$ are obtained recursively by splitting the intervals of V^{n-1} in 2^3 sub-cubes, then translate and dilate the original polynomial basis onto those intervals. This results in the ladder of scaling spaces

$$V^0 \subset V^1 \subset \ldots \subset V^n \subset \ldots \tag{1}$$

which are approaching completeness in L^2. The wavelet spaces W^n are defined as the orthogonal complement of the scaling space V^n in V^{n+1}

$$W^n \oplus V^n = V^{n+1}, \tag{2}$$

which results in the following relation

$$V^N = V^0 \oplus W^0 \oplus W^1 \oplus \ldots \oplus W^{N-1}. \tag{3}$$

2.1 Functions

Functions can be approximated by a projection P^n onto the scaling space V^n, which we denote as

$$f \approx P^n f \overset{\text{def}}{=} f^n = \sum_{\mathbf{l}} f^n_{\mathbf{l}}, \tag{4}$$

where the latter sum runs over all the 2^{3n} cubes that make up a uniform grid at length scale 2^{-n}. Obviously, larger n means higher resolution and thus a better approximation. Importantly, these cubes completely fill the space of the unit cube, without overlapping, which means that all of them are necessary in order to get a complete description of f.

Similarly, a function projection onto the wavelet space W^n is denoted as

$$Q^n f \overset{\text{def}}{=} df^n = \sum_{\mathbf{l}} df^n_{\mathbf{l}}. \tag{5}$$

Here it should be noted that such a wavelet projection is not an approximation to the function, but should be regarded as a difference between two consecutive approximations. By making use of the relation in Eq. (3), we can arrive at two equivalent representations for the approximated function:

$$f \approx f^N = \sum_{\mathbf{l}} f^N_{\mathbf{l}} = f^0_{\mathbf{0}} + \sum_{n=0}^{N-1} \sum_{\mathbf{l}} df^n_{\mathbf{l}}, \tag{6}$$

where the former can be thought of as a *high-resolution* representation at a uniform length scale N, while the latter is a *multi-resolution* representation that spans several different length scales $n = \{0, \dots, N-1\}$. The two representations are completely equivalent both in terms of precision and complexity (number of expansion coefficients), but the latter has one significant advantage: since it is built up as a series of *corrections* to the coarse approximation at scale zero, one can choose to keep only the terms that add a significant contribution [12, 21]

$$||df^n_{\mathbf{l}}|| > \frac{\epsilon}{2^{n/2}} ||f||, \tag{7}$$

where ϵ is some global precision threshold.

2.2 Convolution operators

As will be shown in the following sections, for SCF algorithms within a preconditioned steepest descent framework, the necessary operators are the Poisson operator for the Coulomb and exact exchange contributions and the bound-state Helmholtz operator for the SCF iterations. Their Green's kernel can be written as

$$G^{\mu}(r - r') = \frac{e^{-\mu\|r-r'\|}}{4\pi\|r - r'\|}, \tag{8}$$

where $\mu > 0$ yields the bound-state Helmholtz kernel, whereas $\mu = 0$ is the Poisson kernel. Their application is achieved by convolution of a function with the corresponding Green's kernel

$$g(r) = [Tf](r) = \int G^{\mu}(r - r')\big[f(r')\big]\, \mathrm{d}r', \tag{9}$$

once an approximate separated form in terms of Gaussian functions has been computed [21–23]:

$$G^{\mu}(r - r') \approx \sum_{i=1}^{M} a_i e^{-\alpha_i (r-r')^2}. \tag{10}$$

The non-standard [22, 24] form of the operator T is built as a telescopic expansion of the finest scale projection $T^N = P^N T P^N$

$$T^N = T^0 + \sum_{n=0}^{N-1} (A^n + B^n + C^n), \tag{11}$$

where $A^n = Q^n T Q^n$, $B^n = Q^n T P^n$, $C^n = P^n T Q^n$. Thanks to the vanishing moments of the MW basis, the matrix representations of A^n, B^n and C^n (which contain at least one wavelet projector) are diagonally dominant for both the Poisson and bound-state Helmholtz kernels. Therefore, all terms beyond a predetermined bandwidth can be omitted in the operator application, by a screening similar to the one for functions in Eq. (7). In particular, we have previously shown that the application of the Poisson operator for the calculation of the electrostatic potential scales linearly with the size of the system [25].

2.3 Derivative operators

The discontinuities in the MW basis leads to a number of problems when considering derivative operators. In contrast to the well-behaved smoothing properties of the integral operators as discussed above, the application of the derivative operator will amplify the numerical noise arising from the discontinuity between adjacent intervals in the representation. In particular, since the standard construction [14] of the operator allows only for a first derivative to be defined, higher derivatives have to be computed by repeated application of the first derivative, which will effectively propagate, and further amplify, the numerical noise at every step. This prohibits the use of MW in certain situations, like iterative time-propagation methods involving a derivative in the propagation operator.

A new class of derivative operators was proposed recently by Anderson et al. [26], addressing some of the issues with the original construction. The idea behind the new construction was to realize that the MW representations are usually discontinuous *approximations* of functions that are *supposed to be* smooth and continuous. In these situations, a workaround can be achieved by moving to an auxiliary *continuous* basis, compute the derivative, and then move back to the original (discontinuous) MW basis. Anderson et al. proposed either b-splines or bandlimited exponentials for this auxiliary basis.

It should be noted that the new operators *assume* that the input function is n times differentiable, even if its MW representation is clearly not. It is thus only appropriate if the function does not in fact contain any analytic discontinuities or cusps. It is well known that the non-relativistic electronic wavefunction in any point-nucleus model *does* contain cusps at the nuclear positions, but there are workarounds for this problem, by either removing the cusps from the wavefunction analytically [27], or by introducing effective core potentials.

In the following we will avoid this issue altogether, by formulating the HF equations without any reference to the kinetic energy (or derivative) operator.

3 The Hartree–Fock equations

The HF equations are indisputably the cornerstone of quantum chemistry. We will here briefly revise the formalism as presented by Jensen [28]. It constitute a concise yet formally correct derivation, which also has the advantage of being completely independent of the choice of basis.

We start by considering the energy expression of a Slater determinant:

$$\langle E\rangle_{HF} = \langle \Psi|\hat{H}|\Psi\rangle, \tag{12}$$

where Ψ is a single determinant and $\hat{H}$ is the electronic Hamiltonian operator.

In terms of the spinorbitals $\left\{\varphi_i, i = 1\dots N\right\}$ defining the Slater determinant, the energy expression can be written as follows:

$$\langle E\rangle_{HF} = \sum_i h_i + \frac{1}{2}\sum_{ij}\left(J_{ij} - K_{ij}\right) + U_N, \tag{13}$$

where h_i represents the one-electron part of the energy, J_{ij} is the Coulomb interaction, K_{ij} is the exchange interaction and U_N constitutes the inter-nuclear repulsion. They are obtained respectively as:

$$h_i = \langle\varphi_i|\hat{h}|\varphi_i\rangle = \langle\varphi_i|\hat{T} + \hat{V}_N|\varphi_i\rangle, \tag{14}$$

$$J_{ij} = \langle\varphi_i(x_1)\varphi_j(x_2)|1/r_{12}|\varphi_i(x_1)\varphi_j(x_2)\rangle, \tag{15}$$

$$K_{ij} = \left\langle\varphi_i(x_1)\varphi_j(x_2)|1/r_{12}|\varphi_j(x_1)\varphi_i(x_2)\right\rangle, \tag{16}$$

$$U_N = \sum_{I>J} \frac{Z_I Z_J}{R_{IJ}}. \tag{17}$$

In the above equations, lowercase indices run over the electrons, uppercase ones run over the nuclei, Z is a nuclear charge, R is the inter-nuclear distance, r is the inter-electronic distance, $\hat{T} = -\nabla^2/2$ is the kinetic energy operator, $\hat{V}_N = -\sum_I Z_I/|\boldsymbol{R}_I - r|$ is the nuclear potential, and atomic units ($\hbar = 1$, $q_e = -1$, $m_e = 1$) are used throughout. It is useful to define the effective one-electron Coulomb and Exchange operators as:

$$\hat{J}_i|\varphi_j\rangle \overset{\text{def}}{=} \langle\varphi_i|1/r_{12}|\varphi_i\rangle|\varphi_j\rangle, \tag{18}$$

$$\hat{K}_i|\varphi_j\rangle \overset{\text{def}}{=} \left\langle\varphi_i|1/r_{12}|\varphi_j\right\rangle|\varphi_i\rangle. \tag{19}$$

In accordance to the variational principle, the minimizer is obtained by writing the Lagrangian equations with the constraint of orthonormal occupied orbitals, and differentiating with respect to orbital variations:

$$\delta L = \sum_i \left\langle\delta\varphi_i|\hat{F}|\varphi_i\right\rangle - \sum_{ij} \lambda_{ij}\langle\delta\varphi_i|\varphi_j\rangle + \sum_i \left\langle\delta\varphi_i|\hat{F}|\varphi_i\right\rangle^* - \sum_{ij} \lambda_{ji}\langle\delta\varphi_i|\varphi_j\rangle^*, \tag{20}$$

where the Fock operator $\hat{F}$ is defined as:

$$\hat{F} = \hat{h} + \hat{J} - \hat{K} = \hat{h} + \sum_j \hat{J}_j - \hat{K}_j. \tag{21}$$

The functional derivative of the Lagrangian with respect to an arbitrary orbital variation $\delta\varphi_i$ and of its complex conjugate $\delta\varphi_i^*$ can then be written as:

$$\frac{\delta L}{\delta\varphi_i} = \hat{F}|\varphi_i\rangle - \sum_j \lambda_{ij}|\varphi_j\rangle, \tag{22}$$

$$\frac{\delta L}{\delta\varphi_i^*} = \hat{F}|\varphi_i\rangle^* - \sum_j \lambda_{ji}|\varphi_j\rangle^*. \tag{23}$$

The Lagrange multipliers constitute a Hermitian matrix [28], which leads to the coupled HF equations:

$$\hat{F}\varphi_j = \sum_i \varphi_i F_{ij}. \tag{24}$$

The Fock matrix F can be formally obtained by projecting the above equations along the directions defined by the set of occupied orbitals:

$$F = \left\langle\boldsymbol{\Phi}|\hat{F}|\boldsymbol{\Phi}\right\rangle. \tag{25}$$

In the equation above and in the rest of the paper, we made use of a shorthand notation, indicating with $\boldsymbol{\Phi} = (\varphi_1, \dots, \varphi_N)$ the row-vector of all occupied orbitals.

The Fock operator depends on the orbitals implicitly through $\hat{J}$ and $\hat{K}$. The equations must therefore be solved iteratively. The straightforward iteration would in practice correspond to a steepest descent minimization:

$$\tilde{\boldsymbol{\Phi}}^{n+1} - \boldsymbol{\Phi}^n = -\alpha \frac{\delta L}{\delta \boldsymbol{\Phi}} = -\alpha\left(\hat{F}^n \boldsymbol{\Phi}^n - \boldsymbol{\Phi}^n F^n\right), \tag{26}$$

where α defines the length of the step and the negative sign emphasizes that the step is in the opposite direction of the gradient. We notice also that the orbital set $\left\{\tilde{\varphi}_i^{n+1}, i = 1 \dots N\right\}$ is not orthonormal, because the chosen parametrization is linear and not exponential [1, 29]. Throughout the paper we will use $\tilde{\varphi}$ to refer to non-(ortho)normalized orbitals.

The direct minimization described above is at best lengthy and often leads to either oscillations or even worse to divergent behavior [1]. The usual strategy to solve the HF equations consists in projecting the equations onto a given basis, and solving the so called Roothaan–Hall equations thereby obtained with an acceleration method known as Direct Inversion of the Iterative Subspace (DIIS) [30]. The DIIS is however centered on minimizing the occupied-virtual blocks of the Fock matrix in the finite basis representation. As discussed in the previous section, this is prevented when a MW approach is employed, because the basis set is adaptively refined for each function and should therefore be regarded as infinite and the use of differential operators within a MW basis is problematic.

An alternative is constituted by the integral representation of the HF equations as shown in the next section.

4 Helmholtz kernel and integral formulation

The use of an integral equation formalism to solve the Schrödinger equation was first proposed by Kalos [31]. Let us here consider the derivation for a one-electron system, for which we have the Schrödinger equation:

$$\left[-\frac{\nabla^2}{2} + V(r)\right]\varphi(r) = \epsilon\varphi(r). \tag{27}$$

Such an equation can be rewritten in an integral form by making use of a Green's function formalism. The starting point is the Helmholtz equation

$$(-\nabla^2 + \mu^2)g(r) = f(r), \tag{28}$$

which admits a solution in terms of the following Green's function

$$(-\nabla^2 + \mu^2)G^\mu(r) = \delta(r), \quad G^\mu(r) = \frac{e^{-\mu|r|}}{4\pi|r|}. \tag{29}$$

By making use of this Green's function kernel, and choosing $\mu^2 = -2\epsilon$, the Schrödinger equation can be written in an integral form:

$$\varphi(r) = -2\hat{G}^{\mu}\left[\hat{V}\varphi\right] = -2\int G^{\mu}(r-r')\left[\hat{V}(r')\varphi(r')\right]\mathrm{d}r'. \tag{30}$$

The above integral formulation does not require the explicit use of the kinetic energy operator, which has been formally inverted as follows:

$$(\hat{T}-\epsilon)^{-1} = 2\hat{G}^{\mu}. \tag{31}$$

The integral formulation provides also a natural starting point for efficient iterative algorithms. At each iteration n, the successive orbital φ^{n+1} is obtained as:

$$\tilde{\varphi}^{n+1} = -2\hat{G}^{\mu^n}\left[\hat{V}\varphi^n\right]. \tag{32}$$

For a practical realization of the algorithm, it is also necessary to be able to compute the energy expectation value $\epsilon = \langle\varphi|\hat{H}|\varphi\rangle$ without recurring to the explicit evaluation of the kinetic energy. Consider the expectation value at iteration $n+1$, if φ^{n+1} is obtained through Eq. (32), it is easy to show that ϵ^{n+1} can be obtained as a direct update

$$\epsilon^{n+1} = \epsilon^n + \frac{\left\langle\tilde{\varphi}^{n+1}|\hat{V}|\tilde{\varphi}^{n+1}-\varphi^n\right\rangle}{\langle\tilde{\varphi}^{n+1}|\tilde{\varphi}^{n+1}\rangle}. \tag{33}$$

This shows that the expectation value of the total energy can be obtained by updating ϵ^n with the matrix element of the potential operator involving the new orbital $\tilde{\varphi}^{n+1}$ and the previous one φ^n. We underline that such a prescription is valid for an arbitrary form of the potential and only requires that the Helmholtz operator used in Eq. (32) is computed with $\mu = \sqrt{-2\epsilon^n}$. It is however not required that ϵ^n be the expectation value at iteration n, which allows to start with a predefined initial guess ϵ^0 at the first iteration.

5 Integral formulation for a many-electron system

The procedure described in the previous section can be extended to a many-electron system, to compute all elements of the Fock matrix defined in Eq. (25). To simplify the notation, all occupied orbitals $\{\varphi_i\}$ are collected in the row-vector $\boldsymbol{\Phi}$. Starting from the coupled HF Eq. (26), and applying the Helmholtz operator $\hat{G}^{\boldsymbol{\mu}^n}$:

$$\tilde{\boldsymbol{\Phi}}^{n+1} - \boldsymbol{\Phi}^n = -\alpha\hat{G}^{\boldsymbol{\mu}^n}\left[\left(\hat{T}+\hat{V}^n\right)\boldsymbol{\Phi}^n - \boldsymbol{\Phi}^n F^n\right]. \tag{34}$$

In the above equations the operators $\hat{T}$ and $\hat{V}$ are applied on each component of $\boldsymbol{\Phi}$. Similarly $\hat{G}^{\boldsymbol{\mu}}$ is applied on each component of the resulting vector in the square brackets, making sure the proper μ_i is employed for each component i. By recalling the relationship between the Helmholtz kernel and the shifted kinetic energy, one obtains:

$$\tilde{\Phi}^{n+1} = -2\hat{G}^{\mu^n}\left[\hat{V}^n\Phi^n + \Phi^n(\Lambda^n - F^n)\right], \tag{35}$$

where $\alpha = 1$ and $\Lambda^n_{ij} = \lambda^n_i\delta_{ij}$ is a diagonal matrix, with $\lambda^n_i = -\frac{(\mu^n_i)^2}{2}$.

Although the integral formulation above and the differential one in Eq. (26) are formally equivalent, there is an important distinction to be made. Iterating on the differential formulation corresponds to a steepest descent algorithm, whereas the integral formulation is instead a preconditioned steepest descent algorithm, with the preconditioner $B = (T - \Lambda_{ii})^{-1}$. The integral formulation is therefore a better starting point for optimizations.

Compared to the one-electron case, a few complications arise for the HF coupled equations:

1. The Fock operator, in contrast to the one-electron Hamiltonian, depends on the electronic orbitals. Computing the Fock matrix will therefore require the update of the potential to be taken into account.
2. The electrons are described by a set of orbitals which must be kept orthonormal, in order to arrive at a true Aufbau solution of the HF equations, otherwise a straightforward iteration of Eq. (35) would bring all orbitals to the lowest eigenfunction.

There is also an arbitrariness in the choice of the parameters μ^n_i defining the Helmholtz kernels. The natural choice is to make sure that the diagonal element in the last term cancels ($\Lambda_{ii} = F_{ii}$). Numerical tests, performed by using a fixed μ, have shown that this choice is indeed nearly optimal in terms of the number of iterations needed to converge the orbitals.

The simplest approach to keep orthonormality would be to apply Eq. (35) for each orbital followed by a Gram–Schmidt orthogonalization in order of increasing energy. This would however lead to very slow convergence, especially for valence orbitals, as the convergence of each orbital is restrained by the convergence of lower-lying orbitals, which in turn will depend on *all* orbitals through the potential operator $\hat{V}$.

Harrison et al. [12] described how to use deflation to extract multiple eigenpairs from the Fock operator by recasting the equation for each orbital as a ground state problem. Another approach suggested in the same paper is to simply diagonalize the Fock matrix at each iteration.

5.1 Calculation of Fock matrix

The starting point is a set of orthonormal orbitals Φ^n and an initial guess for the corresponding Fock matrix $F^n \approx \langle\Phi^n|\hat{F}|\Phi^n\rangle$. We emphasize that such a guess need not to be the exact Fock matrix for the given orbital set. The new, and now exact, Fock matrix $\tilde{F}^{n+1} = \langle\tilde{\Phi}^{n+1}|\hat{F}|\tilde{\Phi}^{n+1}\rangle$ in the non-orthonormal basis obtained by applying Eq. (35) can be computed without any reference to the kinetic energy operator. This is in analogy to the one-electron case discussed in Sect. 4.

As for the one-electron case, the application of $(\hat{T} - \lambda_i^n)$ to the new orbital will return the argument from the Helmholtz operator, provided that $\mu_i^n = \sqrt{-2\lambda_i^n}$ was used in this operator:

$$\left(\hat{T} - \lambda_i^n\right)\tilde{\varphi}_i^{n+1} = -\left[\hat{V}^n\varphi_i^n + \sum_j \varphi_j^n\left(\Lambda_{ji}^n - F_{ji}^n\right)\right]. \tag{36}$$

We can now use the above observation to eliminate the kinetic operator from the calculation of the Fock matrix:

$$\begin{aligned}\tilde{F}_{ij}^{n+1} &= \left\langle\tilde{\varphi}_i^{n+1}|\hat{T} + \hat{V}^{n+1}|\tilde{\varphi}_j^{n+1}\right\rangle \\ &= \left\langle\tilde{\varphi}_i^{n+1}|\hat{T} - \lambda_j^n|\tilde{\varphi}_j^{n+1}\right\rangle + \left\langle\tilde{\varphi}_i^{n+1}|\hat{V}^{n+1} + \lambda_j^n|\tilde{\varphi}_j^{n+1}\right\rangle \\ &= -\langle\tilde{\varphi}_i^{n+1}|\hat{V}^n|\varphi_j^n\rangle + \langle\tilde{\varphi}_i^{n+1}|\sum_k\left(\Lambda_{kj}^n - F_{kj}^n\right)\varphi_k^n\rangle + \left\langle\tilde{\varphi}_i^{n+1}|\hat{V}^{n+1} + \lambda_j^n|\tilde{\varphi}_j^{n+1}\right\rangle \\ &= \left\langle\tilde{\varphi}_i^{n+1}|\Delta\hat{V}^n|\tilde{\varphi}_j^{n+1}\right\rangle + \left\langle\tilde{\varphi}_i^{n+1}|\hat{V}^n|\Delta\tilde{\varphi}_j^n\right\rangle + \left(S^nF^n\right)_{ij} + \left(\Delta\tilde{S}_1^nF^n\right)_{ij} + \left(\Delta\tilde{S}_2^n\Lambda^n\right)_{ij}\end{aligned} \tag{37}$$

where in the last step we have defined four updates:

$$\Delta\tilde{\varphi}_i^n = \tilde{\varphi}_i^{n+1} - \varphi_i^n, \tag{38}$$

$$\Delta\hat{V}^n = \hat{V}^{n+1} - \hat{V}^n, \tag{39}$$

$$\Delta\tilde{S}_1^n = \langle\Delta\tilde{\boldsymbol{\Phi}}^n|\boldsymbol{\Phi}^n\rangle, \tag{40}$$

$$\Delta\tilde{S}_2^n = \langle\tilde{\boldsymbol{\Phi}}^{n+1}|\Delta\tilde{\boldsymbol{\Phi}}^n\rangle. \tag{41}$$

Assuming that the original orbital set is orthonormal ($S_{ij}^n = \delta_{ij}$), then the new Fock matrix can be obtained as an update to the previous guess:

$$\tilde{F}^{n+1} = F^n + \Delta\tilde{S}_1^nF^n + \Delta\tilde{S}_2^n\Lambda^n + \Delta\tilde{F}_{pot}^n, \tag{42}$$

where

$$\Delta\tilde{F}_{pot}^n = \langle\tilde{\boldsymbol{\Phi}}^{n+1}|\hat{V}^n|\Delta\tilde{\boldsymbol{\Phi}}^n\rangle + \langle\tilde{\boldsymbol{\Phi}}^{n+1}|\Delta\hat{V}^n|\tilde{\boldsymbol{\Phi}}^{n+1}\rangle. \tag{43}$$

For the definition of the new Fock matrix to be consistent, the two-electron contributions to the potential operator at the $n+1$ step needs to be constructed using an *orthonormalized* version of the corresponding orbitals $\tilde{\boldsymbol{\Phi}}^{n+1}$, while the matrix elements themselves are evaluated in the original non-orthonormal basis. This requires a temporary set $\bar{\boldsymbol{\Phi}}$, constructed e.g. through a Gram–Schmidt process, so that $\langle\bar{\varphi}_i|\bar{\varphi}_j\rangle = \delta_{ij}$. In this basis we can define the Coulomb and Exchange operators as

$$\hat{J}^{n+1}\varphi = \sum_j \varphi(r) \int \frac{\bar{\varphi}_j^{n+1}(r')\bar{\varphi}_j^{n+1}(r')}{|r-r'|}\,\mathrm{d}r', \tag{44}$$

$$\hat{K}^{n+1}\varphi = \sum_j \bar{\varphi}_j^{n+1}(r) \int \frac{\varphi(r')\bar{\varphi}_j^{n+1}(r')}{|r-r'|}\,\mathrm{d}r', \tag{45}$$

which are used when computing the potential updates in Eqs. (39) and (43).

We want to emphasize that these expressions are formally exact, but they *require* that the orbitals of the new set $\tilde{\boldsymbol{\Phi}}^{n+1}$ are related to the orbitals of the old set $\boldsymbol{\Phi}^n$ exactly through the application of the Helmholtz operator in Eq. (35). Otherwise the application of the kinetic energy operator cannot be avoided to obtain the Fock matrix.

5.2 Calculation of energy

The Hamiltonian and the Fock operator differ only by a factor two in the two-electron contribution. The expectation value of the Hamiltonian defined in Eq. (13) can therefore be obtained by taking the trace of the Fock matrix and subtracting the two-electron contribution

$$E = \sum_i F_{ii} - \sum_i \langle \varphi_i | \hat{J} - \hat{K} | \varphi_i \rangle, \tag{46}$$

which means that the kinetic energy operator can be avoided also for the expectation value.

5.3 Orbital orthonormalization

As already mentioned, the basic iteration of the integral operators in Eq. (35) to all orbitals $\{\varphi_i\}$ does not preserve orthonormality. We thus need to explicitly enforce orthonormality in each iteration, but here it is important to avoid projective approaches like the Gram–Schmidt procedure, because these will not allow us to carry over the Fock matrix to the new orbitals. Instead, we can make use of the overlap matrix $\tilde{S} = \langle \tilde{\boldsymbol{\Phi}} | \tilde{\boldsymbol{\Phi}} \rangle$ in a Löwdin transformation [32]

$$\bar{\boldsymbol{\Phi}} = \tilde{\boldsymbol{\Phi}}\tilde{S}^{-1/2}, \qquad \langle \bar{\boldsymbol{\Phi}} | \bar{\boldsymbol{\Phi}} \rangle = I, \tag{47}$$

which allows us to employ the same transformation to the Fock matrix, thus keeping it consistent with the new orthonormal orbitals.

In order to speed up convergence it is useful to augment the Löwdin transformation with another rotation M that brings the orbitals to a particular form. For small systems this can be a diagonalization of the Fock matrix, but for larger systems it is often beneficial to localize the orbitals, e.g. in a Foster–Boys [33, 34] transformation that maximizes the separation between the orbitals from the functional $L_{FB} = \sum_i \langle \varphi_i | \mathbf{r} | \varphi_i \rangle^2$. In practice it will not be necessary to diagonalize/localize in every SCF iteration, so the matrix M can be chosen as the identity for many intermediate steps. The combined transformation matrix then becomes $\tilde{U} = \tilde{S}^{-1/2}M_X$, with $X = \{C, L, I\}$ for canonical, localized or identity, and the new orbital vector and Fock matrix are obtained with

$$\bar{\boldsymbol{\Phi}} = \tilde{\boldsymbol{\Phi}}\tilde{U}, \tag{48}$$

$$\bar{F} = \tilde{U}^{\dagger}\tilde{F}\tilde{U}. \tag{49}$$

6 Implementation

The general algorithm for the SCF optimization for many-electron systems is summarized in Algorithm 1. At each iteration, the input is an orbital vector $\boldsymbol{\Phi}^n$ with the corresponding Fock matrix F^n, which may or may not be diagonal. The orbitals are used to construct the full potential operator $\hat{V}^n$ with updated two-electron contributions. The diagonal part of the Fock matrix is extracted into another matrix Λ^n, and its elements are used to construct the Helmholtz operator for the corresponding orbitals. The Helmholtz operator is applied to each orbital separately, where the argument is corrected with the off-diagonal terms of the Fock matrix, in case non-canonical orbitals are used. This results in a non-orthonormal set of orbitals $\tilde{\boldsymbol{\Phi}}^{n+1}$, and the convergence is judged by the norm of the difference between the input and output orbitals at this point. The Fock matrix corresponding to the new set of orbitals is computed as a pure update from the previous one, as described in Sect. 5.1. It is here important to note that the updated potential operator is computed from an orthonormalized version of the new orbitals, while the matrix elements are computed using the original non-orthonormal set. The final step of the algorithm is to perform an orbital rotation with the Löwdin orthonormalization matrix, optionally combined with another transformation that brings the orbitals to either canonical or localized form, as described in Sect. 5.3.

Algorithm 1 Iterative SCF optimization of many-electron systems.

```
1: Given initial guess for the orbitals $\Phi^0$ and Fock matrix $F^0$
2: while $\delta >$ threshold do
3:     Update potential operator $\hat{V}^n = \hat{V}_{nuc} + \hat{J}^n - \hat{K}^n$ with current orbitals $\Phi^n$
4:     Define diagonal matrix $\Lambda^n_{ij} = F^n_{ij}\delta_{ij}$
5:     for each orbital $\varphi^n_i$ in $\Phi^n$ do
6:         Build Helmholtz operator $\hat{G}^{\mu_i}$ using $\mu_i = \sqrt{-2\Lambda^n_{ii}}$
7:         Apply Helmholtz operator $\tilde{\varphi}^{n+1}_i = -2\hat{G}^{\mu_i}\left[\hat{V}^n\varphi^n_i + \sum_j \varphi_j \left(\Lambda^n_{ji} - F^n_{ji}\right)\right]$
8:     end for
9:     Compute orbital update $\Delta\tilde{\Phi}^n = \tilde{\Phi}^{n+1} - \Phi^n$ and $\delta = \|\Delta\tilde{\Phi}^n\|$
10:    Compute Fock matrix $\tilde{F}^{n+1} = F^n + \Delta\tilde{S}^n_1 F^n + \Delta\tilde{S}^n_2 \Lambda^n + \Delta\tilde{F}^n_{pot}$
11:    Compute rotation matrix $\tilde{U} = \tilde{S}^{-1/2} M_X$ to diagonalize or localize
12:    Rotate orbitals $\Phi^{n+1} = \tilde{\Phi}^{n+1}\tilde{U}$ and Fock matrix $F^{n+1} = \tilde{U}^\dagger \tilde{F}^{n+1}\tilde{U}$
13: end while
```

7 Results

To illustrate the features of the framework exposed in the previous sections, we present two simple examples: the Hydrogen atom as a prototype, one-electron system, and the Beryllium atom as a minimal many-electron system. We have used the Multiwavelet library MRCPP [35] through its Python interface VAMPyR [36] for the implementation, and we have prepared Jupyter Notebooks [37] that reproduce all the results presented below, which can be run freely on Binder [38]. The many-electron implementation is a fully general HF solver. Its limitations are mostly outside the scope of the present work: the missing features to extend it to larger systems are the possibility to deal with open-shell systems, a robust starting guess generator, an iteration accelerator such as the Krylov Accelerated Inexact Newton (KAIN) method [39], and the computational resources which are invariably limited on a Binder distribution. Nevertheless these two simple examples are sufficient to highlight the main features our our implementation. Within the Multiwavelet library, all mathematical objects are represented within a given numerical precision. This means that all functions and operators are truncated accordingly in their compressed representation, i.e. Eqs. (6) and (11), and the operators are applied with bandwidth thresholds that are consistent with the target precision. It is then expected that the obtained solution is exact with respect to the complete basis set limit up to the given precision, but not beyond, even if the equations might be converged further than this.

7.1 Hydrogen atom

The Hydrogen atom is a simple one-electron system, where the Schrödinger equation can be solved by straightforward power iteration of Eqs. (32) and (33), with an additional normalization step for the wavefunction after each iterations. The potential operator $\hat{V}$ contains only the fixed nuclear potential in this case. It should be noted, however, that even if the potential does not depend on the wavefunction, Eq. (32) must still be solved iteratively. This is in contrast to a traditional fixed-basis approach where the corresponding matrix equation can be readily inverted in a single step.

Figure 1 shows a remarkably uniform convergence of the optimization for the Hydrogen atom: the norm of the wavefunction update is almost exactly halved between each iteration, while the energy update is divided by four. This behavior is expected, since the error in the energy should be quadratic with respect to the error of the wavefunction. The initial guess for the orbital was for convenience chosen as a simple Gaussian function, which was projected onto the numerical grid before entering the iterative procedure. The underlying numerical precision is kept at 10^{-6} throughout, and we do not expect the final solution to be more accurate than this, relative to the true eigenfunction. Indeed, when comparing to the known exact solution for Hydrogen we see that error in the orbital stabilizes just below this threshold, while the energy is several orders of magnitude more precise than the set threshold.

7.2 Beryllium atom

The Beryllium atom, being a closed-shell two-orbital system, requires the general many-electron HF procedure as outlined in Algorithm 1. Figure 2 shows the convergence from a simple starting guess of two linearly independent Gaussians, with just a simple Löwdin orthonormalization between each iteration, i.e., no additional orbital rotation was performed to obtain either canonical or localized orbitals, as described in Sect. 5.3.

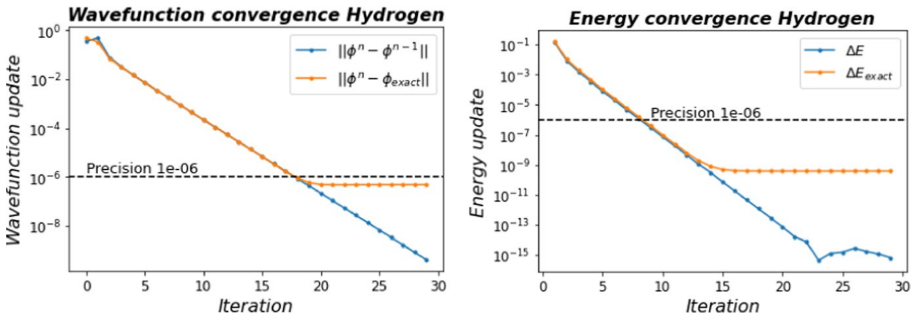


Fig. 1 nvergence of the Hydrogen wavefunction and energy by direct power iteration of Eqs. (32) and (33). The wavefunction is normalized between each iteration. The plots show both the size of the updates relative to the previous iteration, as well as the error with respect to the analytical solution. The overall numerical precision is kept at 10^{-6}

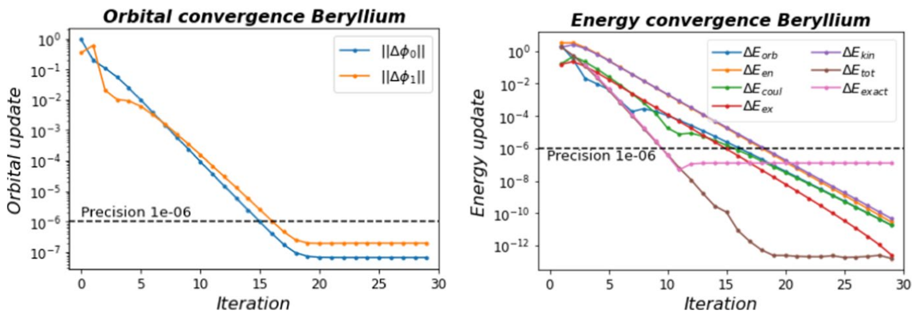


Fig. 2 Convergence of the Beryllium orbitals and energies by iteration of Algorithm 1. The kinetic energy is computed indirectly, by subtracting all other contributions from the total energy, which in turn was computed by the kinetic-free expression in Eq. (46). ΔE_{tot} represents the update in total energy w.r.t. the previous iteration, while ΔE_{exact} represents the difference w.r.t. the Hartree-Fock limit (-14.57302317 Ha) from Thakkar et al. [40]. The overall numerical precision is kept at 10^{-6}

The convergence pattern for Beryllium is very similar to Hydrogen, until the requested numerical precision is achieved; thereafter the orbital convergence tails off. In contrast to Hydrogen, the orbital mixing caused by the orthonormalization step is introducing numerical noise at the order of the truncation threshold, and the orbitals are randomly perturbed in every iteration. This in turn prevents further convergence in the norm of the orbital error. The total energy, on the other hand, shows again a quadratic convergence relative to the orbital errors, as illustrated in the right-hand panel of Fig. 2. However, each energy contribution separately is just linear: it is only their combined sum which exhibits quadratic convergence. It's important to note, though, that even if the total energy converges rapidly and reaches a numerical limit at around 10^{-12} (around the square of the orbital error), its accuracy relative to the HF limit is still bounded by the overall numerical precision in the calculation, in this case 10^{-6}. The reason for this is that in the MW framework, *every* component is approximated according to this precision threshold, including the nuclear, Coulomb and exchange operators. We clearly see this bound when comparing the total energy with the very precise reference from Thakkar et al. [40], displayed as ΔE_{exact} in the figure. In practice, this means that it is not really useful to converge the orbitals and energies all the way to their numerical limits; the SCF can be considered converged whenever the energy update drops below the truncation threshold, in this case after 10-12 iterations.

8 Conclusions

We have presented an implementation of a MW-based SCF solver for the HF equations. The formalism is general and able to deal both closed-shell and open-shell systems alike. The extension to DFT [41], although not considered here, is straightforward by including the exchange and correlation potential. We note however that for DFT derivative operators can be avoided only for local density approximation (LDA) functionals.

Table 1 Levels of contribution:

major, support.

	SRJ	AD	MB	PW	TF	LF
Conceptualization	□	□			□	■
Investigation	■	■	□	□		□
Data curation	□		■	□		
Analysis	■	□	□		□	
Supervision					■	■
Writing – original draft	■	■			□	□
Writing – revisions	■		□			■
Funding acquisition						■
Project administration	□					■

We have shown how it is possible to compute the Fock matrix and the electronic energy by exploiting the formal relation between the level-shifted Laplacian and the bound-state Helmholtz kernel, thus avoiding any reference to the kinetic energy operator. This is an advantage within a MW formalism, because differential operators are formally ill-defined [14], although recent developments have shown that good results can still be achieved [26], also for the kinetic energy expectation value.

We have shown that we are able to obtain high-precision results (basis-set limit within an arbitrary, predefined threshold), and the robust convergence pattern is consistent with the fact that the integral formulation can be viewed as a preconditioned steepest descent [29] method, in contrast to the differential formulation which is instead a steepest descent method.

To illustrate the theoretical framework and demonstrate its applicability we detailed the algorithm and presented two simple examples (Hydrogen and Beryllium atoms), showing that the convergence achieved is consistent with the expected behavior. In particular we have seen that convergence within the predefined threshold is achieved both for the orbital norm and for the energy. Moreover the total energy converges quadratically with respect to the norm of the orbital error. Following an open science paradigm, our algorithms are made freely and readily available for inspection and testing through the Binder platform (Table 1).

Acknowledgements This work has been supported by the Research Council of Norway through a Centre of Excellence Grant (Grant No. 262695) and through a FRIPRO Grant (Grant No. 324590). The simulations were performed on resources provided by Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway (Grant No. NN4654K). LF thanks Prof. Reinhold Schneider for fruitful discussions about the challenges of the HF minimization problem.

Author contributions We use the CRediT taxonomy of contributor roles [42, 43]. The “Investigation” role also includes the “Methodology”, “Software”, and “Validation” roles. The “Analysis” role also includes the “Formal analysis” and “Visualization” roles. The “Funding acquisition” role also includes the “Resources” role. We visualize contributor roles in the following authorship attribution matrix, as suggested in Ref. [44].

Funding Open access funding provided by UiT The Arctic University of Norway (incl University Hospital of North Norway).

References

1. T. Helgaker, P. Jorgensen, J. Olsen, *Molecular Electronic-Structure Theory* (Wiley, New York, 2008)
2. D. Moncrie, S. Wilson, Computational linear dependence in molecular electronic structure calculations using universal basis sets. Int. J. Quantum Chem. **101**, 363–371 (2005)
3. G. Kresse, J. Furthmuller, Effcient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. Phys. Rev. B **54**, 11169–11186 (1996)
4. D. Vanderbilt, Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. Phys. Rev. B **41**, 7892–7895 (1990)
5. G. Kresse, D. Joubert, From ultrasoft pseudopotentials to the projector augmented-wave method. Phys. Rev. B **59**, 1758–1775 (1999)
6. D.J. Singh, L. Nordstrom, *Planewaves, Pseudopotentials, and the LAPW Method* (Springer, New York, 2006)
7. G.Y. Sun et al., Performance of the Vienna ab initio simulation package (VASP) in chemical applications. J. Mol. Struct. Theochem **624**, 37–45 (2003)
8. E. Briggs, D. Sullivan, J. Bernholc, Large-scale electronic-structure calculations with multigrid acceleration. Phys. Rev. B **52**, R5471–R5474 (1995)
9. J.E. Pask, P.A. Sterne, Finite element methods in ab initio electronic structure calculations. Mod. Simul. Mater. Sci. Eng. **13**, R71 (2005)
10. L. Genovese et al., Daubechies wavelets as a basis set for density functional pseudopotential calculations. J. Chem. Phys. **129**, 014109 (2008)
11. J. Pipek, S. Nagy, The kinetic energy operator in the subspaces of wavelet analysis. J. Math. Chem. **46**, 261–282 (2009)
12. R. Harrison et al., Multiresolution quantum chemistry: basic theory and initial applications. J. Chem. Phys. **121**, 11587 (2004)
13. T. Yanai et al., Multiresolution quantum chemistry in multiwavelet bases: Hartree-Fock exchange. J. Chem. Phys. **121**, 6680 (2004)
14. B. Alpert et al., Adaptive solution of partial differential equations in multiwavelet bases. J. Comput. Phys. **182**, 149–190 (2002)
15. B.K. Alpert, A class of bases in $L^2$for the sparse representation of integral operators. SIAM J. Math. Anal. **24**, 246–262 (1999)
16. F.A. Bischoff, R.J. Harrison, E.F. Valeev, Computing many-body wave functions with guaranteed precision: the first-order Møller-Plesset wave function for the ground state of Helium atom. J. Chem. Phys. 104103 (2012)
17. A. Durdek et al., Adaptive order polynomial algorithm in a multiwavelet representation scheme. Appl. Num. Math. **92**, 40–53 (2015)
18. G. Beylkin, R.J. Harrison, K.E. Jordan, Singular operators in multi-wavelet bases. IBM J. Res. Dev. **48**, 161–171 (2004)

19. G. Beylkin, Fast adaptive algorithms in the non-standard form for multidimensional problems. Appl. Comput. Harmon. Anal. **24**, 354–377 (2008)
20. S. Høst et al., The augmented Roothaan-Hall method for optimizing Hartree-Fock and Kohn-Sham density matrices. J. Chem. Phys. **129**, 124106 (2008)
21. L. Frediani et al., Fully adaptive algorithms for multivariate integral equations using the non-standard form and multiwavelets with applications to the Poisson and bound-state Helmholtz kernels in three dimensions. Mol. Phys. **111**, 1143–1160 (2013)
22. G. Beylkin, V. Cheruvu, F. Perez, Fast adaptive algorithms in the non-standard form for multidimensional problems. Appl. Comput. Harmon. Anal. **24**, 354–377 (2008)
23. G. Beylkin, M. Mohlenkamp, Numerical operator calculus in higher dimensions. Proc. Natl. Acad. Sci. **99**, 10246 (2002)
24. D. Gines, G. Beylkin, J. Dunn, LU factorization of non-standard forms and direct multiresolution solvers. Appl. Comput. Harmon. Anal. **5**, 156–201 (1998)
25. S.R. Jensen et al., Linear scaling Coulomb interaction in the multiwavelet basis, a parallel implementation. Int. J. Model Simul. Sci. Comput. **05**, 1441003 (2014)
26. J. Anderson et al., On derivatives of smooth functions represented in multiwavelet bases. J. Comput. Phys. X **4**, 100033 (2019)
27. F.A. Bischoff, Regularizing the molecular potential in electronic structure calculations. I. SCF methods. J. Chem. Phys. **141**, 184105 (2014)
28. F. Jensen, *Introduction to Computational Chemistry* (Wiley, New York, 2013)
29. R. Schneider et al., Direct minimization for calculating invariant subspaces in density functional computations of the electronic structure. J. Comput. Math. **27**, 360–387 (2008)
30. P. Pulay, Convergence acceleration of iterative sequences. The case of SCF iteration. Chem. Phys. Lett. **73**, 393–398 (1980)
31. M.H. Kalos, Monte Carlo calculations of the ground state of three-and four-body nuclei. Phys. Rev. **128**, 1791 (1962)
32. P.-O. Löwdin, On the non-orthogonality problem connected with the use of atomic wave functions in the theory of molecules and crystals. J. Chem. Phys. **18**, 365–375 (1950)
33. S.F. Boys, Construction of some molecular orbitals to be approximately invariant for changes from one molecule to another. Rev. Mod. Phys. **32**, 296–299 (1960)
34. J.M. Foster, S.F. Boys, Canonical configurational interaction procedure. Rev. Mod. Phys. **32**, 300–302 (1960)
35. R. Bast, et al., MRCPP: MultiResolution computation program package (2021). https://github.com/MRChemSoft/mrcpp/tree/release/1.4,version v1.4.0.
36. E. Battistella, et al., VAMPyR: very accurate multiresolution python routines (2021). https://github.com MRChemSoft/vampyr/tree/v0.2rc0,versionv0.2rc0.
37. M. Bjørgve, S. R. Jensen, Kinetic-energy-free algorithms for atoms. https://github.com/MRChemSoft/Kinetic-energy-free-HF
38. The binder project. https://mybinder.org/
39. R. Harrison, Krylov subspace accelerated inexact Newton method for linear and nonlinear equations. J. Comput. Chem. **25**, 328–334 (2004)
40. T. Koga et al., Improved Roothaan-Hartree-Fock wave functions for atoms and ions with N $\leq$ 54. J. Chem. Phys. **103**, 3000–3005 (1995)
41. R.G. Parr, W. Yang, Density-functional theory of the electronic structure of molecules. Annu. Rev. Phys. Chem. **46**, 701–728 (1995)
42. L. Allen et al., Publishing: credit where credit is due. Nature **508**, 312–313 (2014)
43. A. Brand et al., Beyond authorship: attribution, contribution, collaboration, and credit. Learn. Publ. **28**, 151–155 (2015)
44. Researchers are embracing visual tools to give fair credit for work on papers, pp. 5–3 (2021). https://www.natureindex.com/news-blog/researchers-embracing visual-tools-contribution-matrix-give-fair-credit-authorsscientific-papers

5.3 Paper III: Future Perspective for Core-Electron Spectroscopy: Breit Hamiltonian in the Multiwavelets Framework

Abstract

The evolution of core-electron spectroscopy hinges on novel methodologies capable of elucidating structures in powders of strongly correlated materials, including f-element oxides, where single crystals are not available. To address this, sophisticated quantum chemical methods are required to compute core spectroscopy attributes accurately. In this work, we introduce a significant advancement by adapting our flexible real-space multiwavelet basis to the 4-component Dirac-Coulomb-Breit Hamiltonian. Our results demonstrate that multiwavelets can replicate one-dimensional grid-based approaches while presenting a fully three-dimensional methodology, which paves the way for expansion to molecule and material scale applications. Notably, the Multiwavelet framework shows consistent precision, unaffected by nuclear model selection, contingent upon a sufficiently small error threshold and an adequately expansive polynomial basis. Additionally, magnetic and gauge effects from s-orbitals in two-electron systems are found to be equivalent and reconcile well with experimental data observed in K and L edge spectra.

Personal Contributions:

- Actively participated in implementing the `ReMRChem` code, which was crucial for the progression of the project.
- Extended and refined the `VAMPyR` library to streamline operations and augment development geared towards achieving the paper's objectives.

Full Breit Hamiltonian in the Multiwavelets Framework

Christian Tantardini,[*,†,‡] Roberto Di Remigio Eikås,[¶,†] Magnar Bjørgve,[†] Stig Rune Jensen,[†] and Luca Frediani[*,†]

†*Hylleraas center, UiT The Arctic University of Norway, PO Box 6050 Langnes, N-9037 Tromsø, Norway*

‡*Department of Materials Science and NanoEngineering, Rice University, Houston, Texas 77005, United States of America*

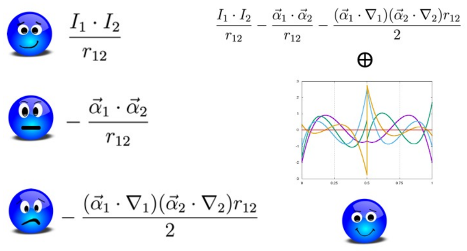
¶*Algorithmiq Ltd., Kanavakatu 3C, FI-00160, Helsinki, Finland*

E-mail: christiantantardini@ymail.com; luca.frediani@uit.no

Abstract

New techniques in core-electron spectroscopy are necessary to resolve the structures of oxides of f-elements and other strongly correlated materials that are present only as powders and not as single crystals. Thus, accurate quantum chemical methods need to be developed to calculate core spectroscopic properties in such materials. In this contribution, we present an important development in this direction, extending our fully adaptive real-space multiwavelet basis framework to tackle the 4-component Dirac-Coulomb-Breit Hamiltonian. We show that Multiwavelets are able to reproduce one-dimensional grid-based approaches. They are however a fully three-dimensional approach which can later on be extended to molecules and materials. Our Multiwavelet implementation attained precise results irrespective of the chosen nuclear model, provided that the error threshold is tight enough and the chosen polynomial basis is

sufficiently large. Furthermore, our results confirmed that in two-electron species, the magnetic and Gauge contributions from s-orbitals are identical in magnitude and can account for the experimental evidence from K and L edges.


$\frac{I_1 \cdot I_2}{r_{12}}$
$- \frac{\vec{\alpha}_1 \cdot \vec{\alpha}_2}{r_{12}}$
$- \frac{(\vec{\alpha}_1 \cdot \nabla_1)(\vec{\alpha}_2 \cdot \nabla_2) r_{12}}{2}$
$\frac{I_1 \cdot I_2}{r_{12}} - \frac{\vec{\alpha}_1 \cdot \vec{\alpha}_2}{r_{12}} - \frac{(\vec{\alpha}_1 \cdot \nabla_1)(\vec{\alpha}_2 \cdot \nabla_2) r_{12}}{2}$
$\oplus$

1 Introduction

Core-electron spectroscopies like X-ray photoelectron spectroscopy, X-ray absorption spectroscopy and electron energy loss spectroscopy are powerful tools to investigate the electronic structure of transition-metal and rare-earth materials. For example, multi-layered transition metal carbides and carbonitrides $M_{n+1}AX_n$, where M is an early transition metal, A is an A-group element (mostly groups 13 and 14), X is C or/and N and n is 1 to 3.[1] These materials can be employed for energy storage systems, such as lithium-ion batteries,[1–5] lithium-ion capacitors,[6] aqueous pseudocapacitors,[7,8] and transparent conductive films.[9] Additionally, rare earths are contained in transparent conducting oxides which are considered the new frontier in the area of optoelectronics.[10–12] These materials have the unique behaviour of being both optically transparent and electrically conducting which makes them key components in many optoelectronic devices such as solar cells, flat panel displays, thin film transistors, and light emitting diodes.[10–12]

Unfortunately, their spectra are not straightforwardly interpretable due to relativistic effects. All relativistic effects such as spin-orbit interactions, electron-electron interaction in the valence shell, and between core and valence electrons, will play a role in the core electron spectra.[13–22] A computational approach based on first-principle calculations that will take into account both relativity and electron correlation could help the interpretation of such spectra. A recent, promising approach in quantum chemistry is based on multiresolution analysis (MRA), by making use of multiwavelets (MWs).[23] This method has gained momentum in recent years, and has been applied to compute complete basis set (CBS) limit results for energies and linear response properties of a large number of compounds both within Hartree-Fock (HF) and density functional theory (DFT).[24–28] A variational treatment of relativistic effects into MRA will allow modelling the spectra of transition metal and rare earth materials. An important step in this direction was presented to tackle the mean-field atomic and molecular Dirac-Coulomb problem in an adaptive, 4-component multiwavelets basis.[29,30] In such a model the electrons are considered static charges where the average interaction between electrons is modelled with the Coulomb-like term only. This is the lowest-order relativistic approximation for the two-electron interaction, which disregards the magnetic interactions, such as spin-other-orbit, and the retardation effects due to the finite speed of light. These effects are important and must be taken into account for a realistic modelling of core-electron spectroscopies. Therefore,[31–34] the Breit interaction terms must thus be included.[35–39] The Breit Hamiltonian adds two negative terms, called Gaunt and Gauge, respectively:

$$\begin{aligned}\hat{H}^{Coulomb} + \hat{H}^{Breit} &= \hat{H}^{Coulomb} + \hat{H}^{Gaunt} + \hat{H}^{Gauge} \\ &= \frac{I_1 \cdot I_2}{r_{12}} - \frac{\vec{\alpha}_1 \cdot \vec{\alpha}_2}{r_{12}} - \frac{(\vec{\alpha}_1 \cdot \nabla_1)(\vec{\alpha}_2 \cdot \nabla_2) r_{12}}{2}\end{aligned} \tag{1}$$

The first term in Eq. (1) is the non-relativistic Coulomb interaction. The second term, called Gaunt, can be seen, in the non-relativistic limit, as the scalar product between the curl of two spin orbitals: $\vec{\alpha}_i \sim \nabla \times \phi_i$.[37] $\vec{\alpha}$ denotes a Cartesian vector collecting the 4×4 Dirac matrices α_x, α_y and α_z (*vide infra*). When $\vec{\alpha}$ acts on a 4-component orbital, it couples its components, as detailed later on in this contribution. This means that the spin rotation of one electron on its axis generates a vector potential that will interact with the vector potentials generated by all other electrons present in the system,[37] resulting in a scalar potential. Finally, the third term, called Gauge, describes the retardation effects due to the reciprocal interaction between the rotational vector fields $(\alpha_i \cdot \nabla_i)$ of two electrons.[37] These contributions cannot be neglected in systems that contain heavy or super-heavy elements, especially in the calculation of core spectroscopic properties.[31–34]

In this contribution, we will present the adaptive MRA multiwavelet implementation of the *full* Breit interaction as a perturbative correction on top of a 4-component Dirac-Coulomb-Hartree-Fock wavefunction. We will demonstrate the precision of our implementation by comparing ground-state energies of highly-charged helium-like ions with increasing Z, $\mathrm{X}^{(\mathrm{Z}-2)+}$, performed with our Python code, *VAMPyR* (Very Accurate Multiresolution Python Routines)[40] with numerical radial integration in *GRASP*[41] and Gaussian basis set calculations with the *DIRAC*[42] software.

2 Theory and Implementation

2.1 Multiresolution Analysis and Multiwavelets

Multiresolution analysis[43] is constructed by considering a set of orthonormal functions called *scaling* functions $\phi_i(x)$ supported on the interval $[0, 1]$. They can be dilated and translated to obtain a corresponding basis in subintervals of $[0, 1]$. The most common procedure is a dyadic subdivision, such that at scale n there will be 2^n intervals defined by a translation index $l = 0, 2^n - 1$ such that the scaling functions in the l-th interval $[l/2^n, (l + 1)/2^n]$ are obtained as:

$$\phi^n_{il} = 2^{n/2}\phi_i(2^n x - l) \tag{2}$$

Additionally, functions at subsequent scales are connected by the *two-scale relationships* which allow to obtain the scaling function at scale n as a linear combination of scaling functions at scale $n - 1$.

This construction leads to a ladder of scaling spaces in a telescopic sequence which is dense in L^2:

$$V_0^k \subset V_1^k \subset .....V_n^k \subset .... \subset L^2 \tag{3}$$

The *multiwavelet functions* are then obtained as the orthogonal complement of the scaling functions at scale $n+1$ with respect to the ones at scale n.

$$V_n^k \oplus W_n^k = V_{n+1}^k, \quad W_n^k \perp V_n^k \tag{4}$$

In the construction of Alpert,[44] the scaling functions are a simple set of polynomials, and the wavelet functions are then piecewise polynomial functions. The possibility to construct efficient algorithms, with precise error control, relies on the combination of several properties of such a construction. Here, it will suffice to say that the most important aspects concern the disjoint support of the basis, which enables function-based adaptivity, the vanishing moments of the wavelet functions, which guarantees fast decay of the representation coefficients, the non-standard (NS) form of operators,[45] which uncouples scales during operator application thus preserving adaptivity, the separated representation of integral kernels, which leads to low-scaling algorithms. The interested reader is referred to the available literature for details about those aspects.[23,44,46,47]

2.2 Mean-field two-electron operators in a Multiwavelets Basis

We will summarise the main methodological developments enabling the results in this contribution. We first recall that in a relativistic framework, molecular orbitals are vectors with four complex components. We will use indices:

- $u, w \in \{x, y, z\}$ for Cartesian components,
- $p, q, \ldots$ for occupied 4-component orbitals,
- $A, B, \ldots \in \{1, 2, 3, 4\}$ for orbital components.

Furthermore, Greek capital letters will be used for the 4-component orbitals and their lowercase counterparts will be used for the corresponding components:

$$\Phi_p = \begin{pmatrix} \varphi_p^1 \\ \varphi_p^2 \\ \varphi_p^3 \\ \varphi_p^4 \end{pmatrix} \tag{5}$$

The corresponding Hermitian conjugate (transposed and complex conjugate) orbital is:

$$\Phi_p^\dagger = \begin{pmatrix} \overline{\varphi}_p^1 & \overline{\varphi}_p^2 & \overline{\varphi}_p^3 & \overline{\varphi}_p^4 \end{pmatrix} \tag{6}$$

with † denoting Hermitian conjugation and overline complex conjugation of a component.

To avoid confusion we will also refer to the instantaneous electron interaction (first term in Eq. 1 as the *Coulomb* term, whereas we will use the terms *direct* and *exchange* to refer to the two parts of each term, arising from the fermionic nature of the electrons.

For the Coulomb operator $g^{Coulomb}(\vec{r}_1, \vec{r}_2) = \frac{I_1 \cdot I_2}{r_{12}}$, the direct and exchange operators are straightforward and shown in Eqs. (7a) and (7b) in the Supporting Information, respectively. In practice, these operators are applied as convolutions. Efficient and accurate convolution with an integral operator is implemented in a separated representation (see Ref. 47 for details). We underline that the Coulomb part of the two-electron interaction is in this framework *diagonal*, in the sense that it is not coupling the four components of the spinor. In a Gaussian Type Orbital (GTO) framework, the exchange part would instead couple the four components of the spinor, because the formalism is tied to the atomic orbital (AO) densities, thus generating an artificial coupling once the exchange operation is performed.[48]

We proceed similarly for the Gaunt operator $g^{Gaunt}(\vec{r}_1, \vec{r}_2) = -\frac{\vec{\alpha}_1 \cdot \vec{\alpha}_2}{r_{12}}$. Note that the $\vec{\alpha}$ appearing in the numerator are Cartesian vectors whose components are 4×4 anti-diagonal block matrices:

$$\alpha_u = \begin{pmatrix} 0 & \sigma_u \\ \sigma_u & 0 \end{pmatrix} \tag{7}$$

with σ_u, $u \in \{x, y, z\}$, the Pauli matrices. Applying α_u on a 4-component orbital, in practice reorders the components, possibly multiplied by a phase factor.

The two-electron energy for the Gaunt operator is thus:

$$E^{Gaunt} = -\frac{1}{2}\sum_{pq}\int d\vec{r}_1 \int d\vec{r}_2 \frac{\vec{j}_{pp}(\vec{r}_1)\cdot\vec{j}_{qq}(\vec{r}_2)}{r_{12}} \tag{8}$$

$$+\frac{1}{2}\sum_{pq}\int d\vec{r}_1 \int d\vec{r}_2 \frac{\vec{j}_{pq}(\vec{r}_1)\cdot\vec{j}_{qp}(\vec{r}_2)}{r_{12}} \tag{9}$$

where we have introduced the current density Cartesian vector, with components:

$$j_{pq;u} = \sum_{AB} \overline{\varphi}_p^A \alpha_u^{AB} \varphi_q^B, \tag{10}$$

to rewrite the expression more compactly. The corresponding mean-field, effective one-electron, direct and exchange operators are:

$$J^{Gaunt}\Phi_k = \sum_u \left[\int d\vec{r}_2 \frac{\sum_q \Phi_q^\dagger(\vec{r}_2)\alpha_u\Phi_q(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}\right]\alpha_u\Phi_k(\vec{r}_1) = \left[\int d\vec{r}_2 \frac{\vec{j}(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}\right]\cdot[\vec{\alpha}\Phi_k] \tag{11a}$$

$$K^{Gaunt}\Phi_k = \sum_q\sum_u \alpha_u\Phi_q(\vec{r}_1)\left[\int d\vec{r}_2 \frac{j_{qk;u}(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}\right] = \sum_q [\vec{\alpha}\Phi_q]\cdot\vec{V}_{qk}^{Gaunt}, \tag{11b}$$

$\vec{j}$ is the trace of the matrix collecting the orbital-pair current densities $j_{pq;u}$.

The Gaunt direct and exchange operators use the same primitive as the Coulomb operators for the convolution with the inverse-distance kernel. Thus:

1. Although the expressions for the Gaunt mean-field operators appear more complicated than those stemming from the Coulomb interaction, their computational load is only three times higher, because each component of the $\vec{\alpha}$ vector only has four non-zero elements.

2. For each Cartesian component, one can compute a "Gaunt potential" which is then multiplied by the $\vec{\alpha}$-transformed orbital, exactly as for the Coulomb interaction.

Turning our attention to the gauge two-electron potential, we follow the suggestion of Sun *et al.*[49] $-\nabla_1 \frac{1}{r_{12}} \equiv \frac{\vec{r}_{12}}{r_{12}^3} \equiv \nabla_2 \frac{1}{r_{12}}$, and rewrite it as:

$$g^{Gauge}(\vec{r}_1, \vec{r}_2) = \frac{1}{2} \frac{(\vec{\alpha}_1 \cdot \vec{r}_{12})(\vec{\alpha}_2 \cdot \vec{r}_{12})}{r_{12}^3} \tag{12}$$

$$= -\frac{1}{2} \left[\vec{\alpha}_1 \cdot \left(\mp \nabla_{1,2} \frac{1}{r_{12}} \right) \right] (\vec{\alpha}_2 \cdot \vec{r}_1) + \frac{1}{2} \left[\vec{\alpha}_1 \cdot \left(\mp \nabla_{1,2} \frac{1}{r_{12}} \right) \right] (\vec{\alpha}_2 \cdot \vec{r}_2) \tag{13}$$

where the sign/index pairs ($-\nabla_1$ or $+\nabla_2$) can be chosen independently for each of the two terms, giving rise to *four* equivalent expression.

The energy expressions corresponding to each of the above forms can be considerably simplified using integration by parts, thus avoiding the need for differentiating the inverse-distance kernel. However, of the four forms presented above, the energy expression obtained by choosing $+\nabla_2$ in both terms of Eq. (13) is the most compact *and* computationally parsimonious:

$$E^{Gauge} = \frac{1}{2} \sum_{pq} \int d\vec{r}_1 \int d\vec{r}_2 \frac{\left(\vec{j}_{pp}(\vec{r}_1) \cdot \vec{r}_1 \right) \left(\nabla_2 \cdot \vec{j}_{qq}(\vec{r}_2) \right)}{2r_{12}} \tag{14}$$

$$- \frac{1}{2} \sum_{pq} \int d\vec{r}_1 \int d\vec{r}_2 \frac{\left(\vec{j}_{pp}(\vec{r}_1) \cdot \vec{r}_2 \right) \left(\nabla_2 \cdot \vec{j}_{qq}(\vec{r}_2) \right)}{2r_{12}} \tag{15}$$

$$- \frac{1}{2} \sum_{pq} \int d\vec{r}_1 \int d\vec{r}_2 \frac{\vec{j}_{pp}(\vec{r}_1) \cdot \vec{j}_{qq}(\vec{r}_2)}{2r_{12}} \tag{16}$$

$$- \frac{1}{2} \sum_{pq} \int d\vec{r}_1 \int d\vec{r}_2 \frac{\left(\vec{j}_{pq}(\vec{r}_1) \cdot \vec{r}_1 \right) \left(\nabla_2 \cdot \vec{j}_{qp}(\vec{r}_2) \right)}{2r_{12}} \tag{17}$$

$$+ \frac{1}{2} \sum_{pq} \int d\vec{r}_1 \int d\vec{r}_2 \frac{\left(\vec{j}_{pq}(\vec{r}_1) \cdot \vec{r}_2 \right) \left(\nabla_2 \cdot \vec{j}_{qp}(\vec{r}_2) \right)}{2r_{12}} \tag{18}$$

$$+ \frac{1}{2} \sum_{pq} \int d\vec{r}_1 \int d\vec{r}_2 \frac{\vec{j}_{pq}(\vec{r}_1) \cdot \vec{j}_{qp}(\vec{r}_2)}{2r_{12}} \tag{19}$$

The former three terms are the direct contributions and the latter three the exchange contributions. The use of the inverse-distance kernel is the most significant advantage of this formulation, since that is already an efficient and robust computational primitive in a multiwavelet basis. Note that the calculation of the divergence of the orbital current densities

$$\nabla \cdot \vec{j}_{pq} \equiv \frac{\partial j_{pq;x}}{\partial x} + \frac{\partial j_{pq;y}}{\partial y} + \frac{\partial j_{pq;z}}{\partial z}$$

is both efficient and precise in a multiwavelet basis.[50]

Finally, we present the expressions for the direct and exchange Gauge mean-field operators:

$$\begin{aligned}
J^{Gauge}\Phi_k = \frac{1}{2}\Bigg\{ &\left[\int d\vec{r}_2 \frac{\nabla_2 \cdot \vec{j}(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}\right] [(\vec{\alpha}\Phi_k) \cdot \vec{r}_1] - \left[\int d\vec{r}_2 \frac{\vec{j}(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}\right] \cdot [\vec{\alpha}\Phi_k] && (20a)\\
&- \left[\int d\vec{r}_2 \frac{\vec{r}_2 \left(\nabla_2 \cdot \vec{j}(\vec{r}_2)\right)}{|\vec{r}_1 - \vec{r}_2|}\right] \cdot [\vec{\alpha}\Phi_k] \Bigg\}\\
K^{Gauge}\Phi_k = \frac{1}{2}\sum_q \Bigg\{ &\left[\int d\vec{r}_2 \frac{\nabla_2 \cdot \vec{j}_{qk}(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}\right] [(\vec{\alpha}\Phi_k) \cdot \vec{r}_1] - [\vec{\alpha}\Phi_q] \cdot \vec{V}^{Gaunt}_{qk} && (20b)\\
&- \left[\int d\vec{r}_2 \frac{\vec{r}_2 \left(\nabla_2 \cdot \vec{j}_{qk}(\vec{r}_2)\right)}{|\vec{r}_1 - \vec{r}_2|}\right] \cdot [\vec{\alpha}\Phi_k] \Bigg\}.
\end{aligned}$$

All terms in both the direct and exchange operators are applied using the inverse-distance integral operator only.

For completeness, we report also the expressions for the Gauge term when using the inverse-cube-distance form for the operator:

$$g^{Gauge}(\vec{r}_1, \vec{r}_2) = -\frac{(\vec{\alpha}_1 \cdot \vec{r}_{12})\,(\vec{\alpha}_2 \cdot \vec{r}_{12})}{2r_{12}^3}, \tag{21}$$

The two-electron energy reads:

$$E^{Gauge} = -\frac{1}{2}\sum_{pq}\int d\vec{r}_1 \int d\vec{r}_2 \frac{\left(\vec{j}_{pp}(\vec{r}_1)\cdot\vec{r}_{12}\right)\left(\vec{j}_{qq}(\vec{r}_2)\cdot\vec{r}_{12}\right)}{r_{12}^3} \tag{22}$$

$$+\frac{1}{2}\sum_{pq}\int d\vec{r}_1 \int d\vec{r}_2 \frac{\left(\vec{j}_{pq}(\vec{r}_1)\cdot\vec{r}_{12}\right)\left(\vec{j}_{qp}(\vec{r}_2)\cdot\vec{r}_{12}\right)}{r_{12}^3}. \tag{23}$$

While this is arguably more compact than the sum of all six terms in the previous equation (Eqs. (14)- (19)), it has two main disadvantages. First, it is harder to appreciate the physical content of the expression at a glance. Second, it requires the application of a different convolution operator. The latter point is apparent when looking at the expressions for the direct and exchange operators:

$$\begin{aligned}
J^{Gauge}\Phi_k &= \sum_{uw}\left[\int d\vec{r}_2 \frac{(\vec{r}_1-\vec{r}_2)_u(\vec{r}_1-\vec{r}_2)_w}{|\vec{r}_1-\vec{r}_2|^3} j_w(\vec{r}_2)\right]\alpha_u\Phi_k \\
&= \left[\int d\vec{r}_2 \mathbb{G}(\vec{r}_1,\vec{r}_2)\vec{j}(\vec{r}_2)\right]\cdot\vec{\alpha}\Phi_k \qquad (24a)\\
K^{Gauge}\Phi_k &= \sum_q\sum_{uw}\alpha_u\Phi_q\left[\int d\vec{r}_2 \frac{(\vec{r}_1-\vec{r}_2)_u(\vec{r}_1-\vec{r}_2)_w}{|\vec{r}_1-\vec{r}_2|^3} j_{qk;w}(\vec{r}_2)\right] \\
&= \sum_q [\vec{\alpha}\Phi_q]\cdot\left[\int d\vec{r}_2 \mathbb{G}(\vec{r}_1,\vec{r}_2)\vec{j}_{qk}(\vec{r}_2)\right], \qquad (24b)
\end{aligned}$$

The new convolution operator, $\mathbb{G}$, is a *matrix* convolution operator with 6 unique elements, each of which must be implemented by approximating the integral representation of the inverse-cube-distance kernel[51] as a finite exponential sum:[52]

$$\frac{(\vec{r}_1-\vec{r}_2)_u(\vec{r}_1-\vec{r}_2)_w}{|\vec{r}_1-\vec{r}_2|^3} \simeq \sum_\kappa a_\kappa(\vec{r}_1-\vec{r}_2)_u(\vec{r}_1-\vec{r}_2)_w \exp(-b_\kappa|\vec{r}_1-\vec{r}_2|^2). \tag{25}$$

Each term, though anisotropic, can be applied in each Cartesian direction separately. Coefficients and exponents in the sum are obtained similarly to those for the inverse-distance convolution operator, see Ref. [47] for details. This form has been tested in our code, but it turned out to be less stable numerically and significantly more demanding computationally.

3 Computational Details

DIRAC calculations were performed using a nuclear point-charge model and a threshold of 10^{-7} on the norm of the error vector (electronic gradient) was chosen as the convergence criterion for the SCF procedure. The chosen basis set for He, Ne^{8+}, Ar^{16+}, Kr^{34+}, Xe^{52+} and Rn^{84+} was dyall-aug-cvqz.[38,53,54] Furthermore, the calculations were performed using default settings for 4-center integral screening and replacing $(SS|SS)$ integrals by a simple Coulombic correction. In our MW implementation it is not possible to perform such a correction, because 4-center integrals do not appear in the formalism. We investigated whether this could impact our perturbative/variational comparisons: with the full two-electron integral tensors the total energy computed with *DIRAC* changes slightly and computational cost increases *significantly*. However, the *relative error* with respect to both our implementation in *VAMPyR* and in *GRASP* was practically unaffected. This shows that the error is dominated by the intrinsic limitation of the basis set.

4 Results and Discussion

We present results for closed-shell, helium-like species: the core $1s$-orbitals are doubly occupied and our code explicitly enforces Kramers' time-reversal symmetry (TRS),[55,56] such that the 4-component $1s^{\alpha}$ is related to $1s^{\beta}$ by a quaternionic unitary transformation.[57]

In a mean-field treatment – *e.g.* HF and Kohn-Sham DFT – the Coulomb two-electron operator is replaced by the corresponding *Direct* and *Exchange* terms, indicated with J and K, respectively. Further inclusion of the Gaunt and Gauge interactions in Eq. (1) will result into additional J- and K-like terms. Making use of Kramers TRS has a significant impact on the computational cost: the Coulomb interaction will only encompass the *direct* term, whereas *exchange* one will be equal to zero. The Gaunt and Gauge interactions will give rise to both *direct* and *exchange* terms but several contributions will either vanish or be identical to each other.

Previous work by Anderson *et al.*[30] on *full* 4-component Dirac-Coulomb relativistic calculations used smeared nuclear charge models.[58] In particular for the isolated atoms they used the Fermi nuclear model.[58] This was done to mitigate numerical issues treating core orbitals with a point-charge model and improve precision. The Fermi model represents the nuclear charge using the Fermi-Dirac distribution for the nuclear charge density, introducing two parameters: the skin thickness and the half-charge radius. The former is set to 2.30 fm (2.30×10^{-5} Å) for all nuclei.[58] The latter is the radius of a sphere containing half of the total nuclear charge. This parameter depends on the atomic mass of the nucleus M_N, with one expression used when $M_N \leq 5$ atomic mass units and another for $M_N > 5$.[58] The Fermi model for the nuclear charge is smooth and is thus more physically meaningful. Furthermore, it avoids singularities at the nuclei, in contrast to a point-like model. However, the results of Anderson *et al.*[30] showed that the achieved precision of multiwavelet methods with respect to the grid-based approach available in *GRASP* decreases with increasing Z, even though a more physically motivated nuclear model was used.

Our multiwavelet implementation in *VAMPyR* uses two parameters to tune the precision of the calculation: the tolerance, ε, and the polynomial order, k. Furthermore, both point-charge and Fermi models are available for the nuclei. In order to validate our Dirac-Coulomb Hartree-Fock (DCHF) implementation and reassess the impact of the nuclear model, we performed DCHF calculations with a point-charge model and increasingly tighter precision settings. We report the comparison of our results with *GRASP* in Figure 1. The relative errors obtained at looser precision settings, as shown in Fig. 1, are not consistent with the user-requested ϵ for heavy elements. The desired precision is user-selected through the settings for ϵ and k and should, in principle, be achieved irrespective of the nuclear model. However, our results show that a point-charge nuclear model can reproduce grid-based results from *GRASP* only when a very tight tolerance is chosen, see Fig. 1 and Table 1 in Supporting Information (SI). At the opposite end, SCF convergence could not be achieved for $k = 6$, $\epsilon = 10^{-4}$ for Kr^{34+} and heavier elements.

One possible explanation is the choice of point-like nuclear potential, which is nonphysical and not suitable for fully relativistic calculations, but only for nonrelativistic ones. Thus, calculations with a point nucleus require a significant tighter *tolerance* and consequently a higher polynomial order to achieve the same precision of grid-based results from *GRASP*.

After assessing the validity of our method for the DCHF equation, we developed the Gaunt and Gauge two-electron terms in the Breit Hamiltonian as a perturbative correction, as done in *GRASP*. The Gaunt term contains the vector operator $\vec{\alpha}$: it is a Cartesian vector of 4×4 matrices whose antidiagonal blocks are the Pauli matrices for the corresponding Cartesian direction. As we have previously mentioned in the *Introduction*, it can be seen as the curl of a spinorbital in the classical limit.[59] $\vec{\alpha}$ acting on a 4-component orbital mixes its components to give the the current density generated by the rotation of the spin around its axis.[59]

We first compared DCHF results from *DIRAC* with those obtained with *VAMPyR* at high precision (*i.e.* $k = 10$, $\epsilon = 10^{-8}$), see Table 2 in SI. These results confirm and extend to the *full* 4-component regime the observations of Jensen *et al.*: MWs can attain higher precision than large Gaussian atomic basis sets.[60]

Thereafter, we compared our perturbative Gaunt correction, implemented in *VAMPyR*, with the variational implementation available in the *DIRAC* code, see Figure 2. The inclusion of the Gaunt term in the variational self-consistent field procedure is not expected to significantly affect the ground state as previously shown[61] and both results can be compared, see Figure 2. In fact, the logarithm of the unsigned relative errors for the spinorbit energies, see Fig. 2.1, and the Gaunt terms, see Fig. 2.2, between *VAMPyR* and *DIRAC* have the same order of magnitude.

The perturbative Gauge correction only involves the inverse interelectronic distance ker-

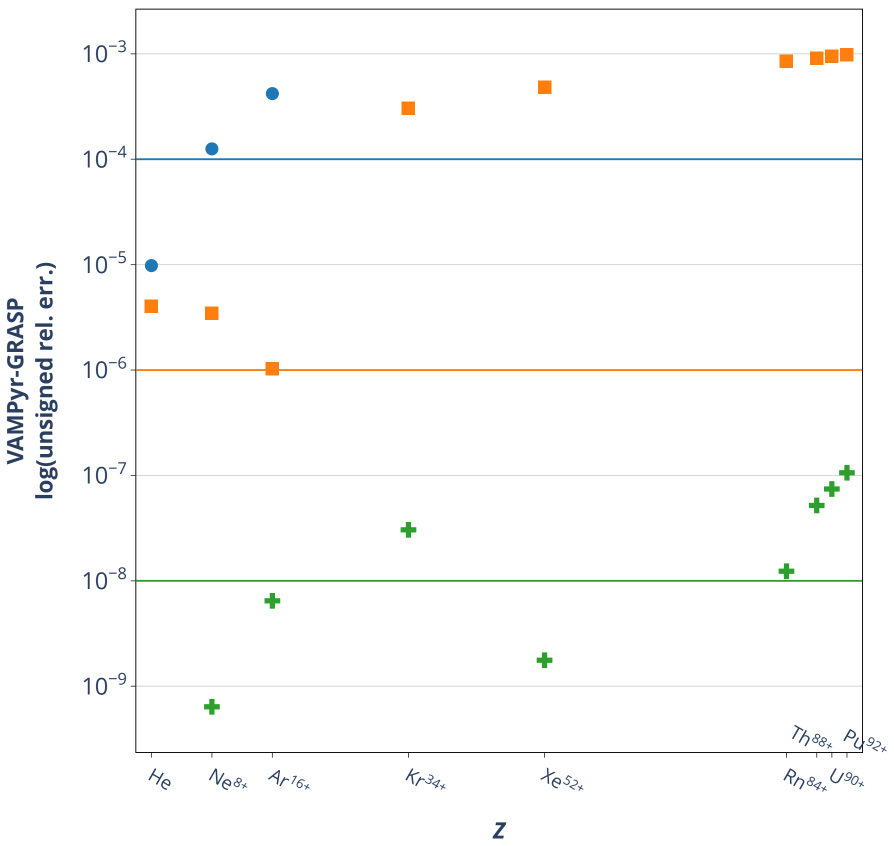
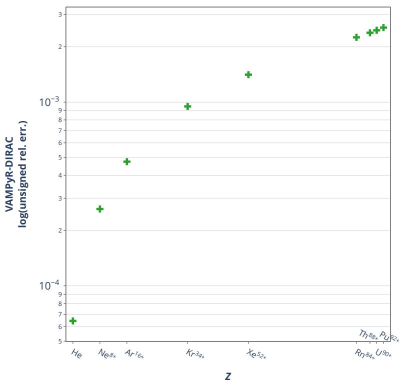
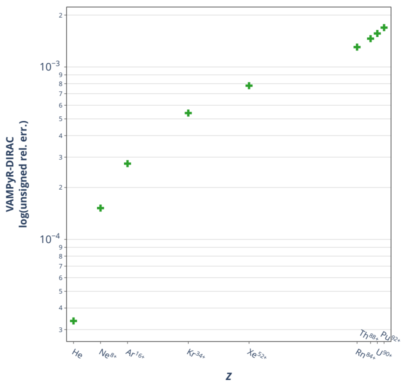


Figure 1: Logarithm of unsigned relative error between the Dirac-Coulomb-Hartree-Fock ground-state energy calculations from *VAMPyR* and *GRASP*. All species are in the electronic configuration $1s^2$. The *VAMPyR* calculations were done with different choices of Legendre polynomial order k and tolerance ϵ: blue circle, $k = 6$, $\epsilon = 10^{-4}$; orange square, $k = 8$, $\epsilon = 10^{-6}$; green cross, $k = 10$, $\epsilon = 10^{-8}$. Both codes have used nuclear point charge model as described in Ref. 58

nel, as shown in by Eqs. (14)- (19), from which it is evident how the magnetic energy term arises as half of the Gaunt term, since both the direct and exchange Gauge contributions (third and sixth terms) contain half of the Gaunt term.



(2.1)



(2.2)

Figure 2: Comparison between the spinorbit energies (a) and Gaunt terms (b) coming from *VAMPyR* and *DIRAC* for selected systems in electronic configuration $1s^2$. The y axis shows the logarithm of the unsigned relative difference between *VAMPyR* and *DIRAC* results. The *VAMPyR* calculations were done with Legendre polynomial order $k = 10$ and tolerance $\epsilon = 10^{-8}$. All codes have used a nuclear point charge model as described in Ref. [58]

For the specific case of $1s^2$ systems, the terms involving a gradient in the Gauge energy (i.e., 1st Eq. (14), 2nd Eq. (15), 4th Eq. (17) and 5th Eq. (18)) are either zero or cancel each other out, up to the chosen numerical precision ε. Thus, the ratio between the Gauge term (E^{Gauge}) and the magnetic interaction energy, which corresponds to half of the Gaunt term ($E^{Mag} = \frac{1}{2}E^{Gaunt}$), should be one (*i.e.*, identical Magnetic and Gauge terms). This was verified comparing the Breit energy corrections from *VAMPyR* and *GRASP* results, see Figure 3 and Table 5 in the SI.

The E^{Gauge}/E^{Mag} ratio was calculated previously using Gaussian atomic orbital basis sets for several atoms from $Z = 9$ to $Z = 79$.[49] It was shown to range between 0.90 (Fluorine) and 0.80 for $Z > 56$, converging asymptotically. In Table 5 of the SI, where we have considered $1s^2$ systems exclusively, we have obtained a unitary ratio between Gauge and magnetic term. Furthermore, the magnitude of the Gauge term from our results in Table 5 in SI confirms what was previously found by Halbert *et al.*[62] that in core-electron spectroscopy the Gauge

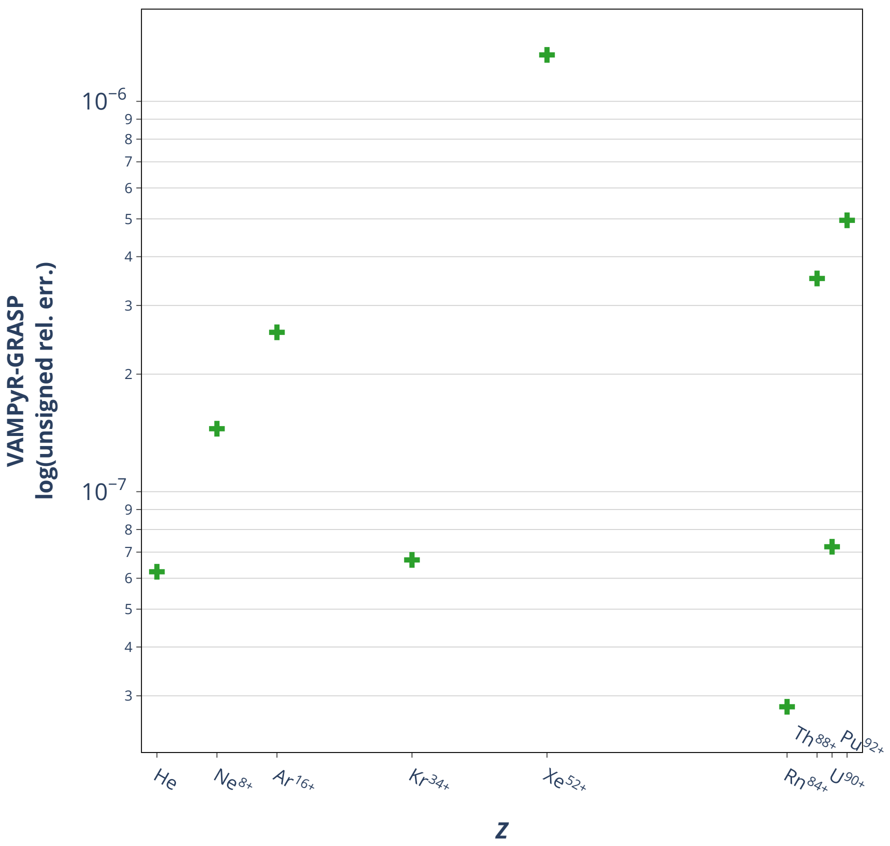


Figure 3: Comparison between the Breit perturbative corrections computed *VAMPyR* and *GRASP* for noble gases and actinides in electronic configuration $1s^2$. The y axis shows the logarithm of the unsigned relative difference between *VAMPyR* and *GRASP* results. The *VAMPyR* calculations were done with Legendre polynomial order $k = 10$ and tolerance $\epsilon = 10^{-8}$. All codes have used a nuclear point charge model as described in Ref. [58]

term remains quite significant for the K and L edges, and it must be accounted for, especially for $1s$ to $2s$ transitions.[63]

5 Conclusions

We have shown that the 4-component Dirac-Coulomb-Hartree-Fock equations can be solved self-consistently with a fully adaptive MW basis irrespective of the chosen nuclear model, as required with Gaussian basis sets.[64,65]

The use of multiresolution analysis (MRA) with a MW basis to solve the KS-DFT equations allows to separate model errors from discretization (*i.e.* basis set) errors, with the latter precisely quantifiable. Thus, the use of a MW basis provides fundamental insight to understand the range of applicability of KS-DFT with localized basis sets. This issue is especially relevant for 4-component relativistic calculations on heavy elements where the description of the core electrons is challenging due the nature of the Dirac equation combined with the extremely high nuclear charge and a reduced availability of GTO bases.

We have shown that the DCHF ground state combined with the Breit Hamiltonian as a perturbative correction can reproduce grid-based calculations performed with *GRASP*. Albeit not performed in this work, the fully variational inclusion of the Gaunt and Gauge terms can be obtained by making use of the corresponding operator expressions (Equations (11a) and (11b) and Equations (20a) and (20b) for Gaunt and Gauge, respectively). This has not been done for the current work both to simplify the comparison with the *GRASP* code and because of excessive memory demands of the current pilot implementation. The latter is indeed the main challenge for future extensions to general molecular systems where the simplifications that enabled our results (time-reversal symmetry, spherical symmetry of the $1s$ orbital) will no longer hold. Work is in progress in our group to overcome these hurdles.

The unitary E^{Gauge}/E^{Mag} ratio for s-orbitals explains how neither Gaunt nor Gauge terms can be neglected for core-electron spectroscopy and explains the importance of considering both these terms when x-ray photoelectron spectra are calculated to fit the experimental ones.[62,63,66] Our results confirm the validity of the MW approach for future development of core-electron spectroscopy to resolve the structures of oxides of f-elements and other strongly correlated systems.

Acknowledgement

We would like to thank Prof. Trond Saue from the CNRS/Université de Toulouse, France, Dr. Jon Grumer from Uppsala University, Sweden, and Dr. Michal Repisky from UiT, The Arctic University of Norway for useful discussions. We acknowledge support from the Research Council of Norway through its Centres of Excellence scheme (262695), through the FRIPRO grant ReMRChem (324590), and from NOTUR – The Norwegian Metacenter for Computational Science through grant of computer time (nn4654k).

Supporting Information Available

All data generated or analyzed during this study are included in the graph showed in this article and tables showed in supporting information.

References

(1) Naguib, M.; Kurtoglu, M.; Presser, V.; Lu, J.; Niu, J.; Heon, M.; Hultman, L.; Gogotsi, Y.; Barsoum, M. W. Two-Dimensional Nanocrystals Produced by Exfoliation of Ti3AlC2. *Advanced Materials* **2011**, *23*, 4248–4253.

(2) Naguib, M.; Come, J.; Dyatkin, B.; Presser, V.; Taberna, P.-L.; Simon, P.; Barsoum, M. W.; Gogotsi, Y. MXene: a promising transition metal carbide anode for lithium-ion batteries. *Electrochemistry Communications* **2012**, *16*, 61–64.

(3) Naguib, M.; Halim, J.; Lu, J.; Cook, K. M.; Hultman, L.; Gogotsi, Y.; Barsoum, M. W. New Two-Dimensional Niobium and Vanadium Carbides as Promising Materials for Li-Ion Batteries. *Journal of the American Chemical Society* **2013**, *135*, 15966–15969.

(4) Mashtalir, O.; Naguib, M.; Mochalin, V. N.; Dall'Agnese, Y.; Heon, M.; Bar-

soum, M. W.; Gogotsi, Y. Intercalation and delamination of layered carbides and carbonitrides. *Nature communications* **2013**, *4*, 1716.

(5) Tang, Q.; Zhou, Z.; Shen, P. Are MXenes Promising Anode Materials for Li Ion Batteries? Computational Studies on Electronic Properties and Li Storage Capability of Ti3C2 and Ti3C2X2 (X = F, OH) Monolayer. *Journal of the American Chemical Society* **2012**, *134*, 16909–16916.

(6) Come, J.; Naguib, M.; Rozier, P.; Barsoum, M. W.; Gogotsi, Y.; Taberna, P.-L.; Morcrette, M.; Simon, P. A Non-Aqueous Asymmetric Cell with a Ti2C-Based Two-Dimensional Negative Electrode. *Journal of The Electrochemical Society* **2012**, *159*, A1368.

(7) Ghidiu, M.; Lukatskaya, M. R.; Zhao, M.-Q.; Gogotsi, Y.; Barsoum, M. W. Conductive two-dimensional titanium carbide 'clay' with high volumetric capacitance. *Nature* **2014**, *516*, 78–81.

(8) Lukatskaya, M. R.; Mashtalir, O.; Ren, C. E.; Dall'Agnese, Y.; Rozier, P.; Taberna, P. L.; Naguib, M.; Simon, P.; Barsoum, M. W.; Gogotsi, Y. Cation Intercalation and High Volumetric Capacitance of Two-Dimensional Titanium Carbide. *Science* **2013**, *341*, 1502–1505.

(9) Halim, J.; Lukatskaya, M. R.; Cook, K. M.; Lu, J.; Smith, C. R.; Näslund, L.-A.; May, S. J.; Hultman, L.; Gogotsi, Y.; Eklund, P.; Barsoum, M. W. Transparent Conductive Two-Dimensional Titanium Carbide Epitaxial Thin Films. *Chemistry of Materials* **2014**, *26*, 2374–2381.

(10) Chandiramouli, R.; Jeyaprakash, B. Review of CdO thin films. *Solid State Sciences* **2013**, *16*, 102–110.

(11) Liu, C. P.; Ho, C. Y.; Dos Reis, R.; Foo, Y.; Guo, P. F.; Zapien, J. A.; Walukiewicz, W.; Yu, K. M. Room-temperature-synthesized high-mobility transparent amorphous $CdO-$

Ga_2O_3 alloys with widely tunable electronic bands. *ACS applied materials & interfaces* **2018**, *10*, 7239–7247.

(12) Dixon, S. C.; Scanlon, D. O.; Carmalt, C. J.; Parkin, I. P. n-Type doped transparent conducting binary oxides: an overview. *Journal of Materials Chemistry C* **2016**, *4*, 6946–6961.

(13) Grobe, R.; Eberly, J. H. Observation of coherence transfer by electron-electron correlation. *Phys. Rev. A* **1993**, *48*, 623–627.

(14) Lundqvist, B. I. Characteristic structure in core electron spectra of metals due to the electron-plasmon coupling. *Physik der kondensierten Materie* **1969**, *9*, 236–248.

(15) Wendin, G.; Ohno, M. Strong Dynamical Effects of Many-Electron Interactions in Photoelectron Spectra from 4s and 4p Core Levels. *Physica Scripta* **1976**, *14*, 148.

(16) Brus, L. E. Electron–electron and electron-hole interactions in small semiconductor crystallites: The size dependence of the lowest excited electronic state. *The Journal of chemical physics* **1984**, *80*, 4403–4409.

(17) Pines, D. Electron interaction in solids. *Canadian Journal of Physics* **1956**, *34*, 1379–1394.

(18) Gusev, A.; Reznik, I.; Tsitrin, V. Electron-electron interaction and antishielding constants of core shells of atoms. *Journal of Physics: Condensed Matter* **1995**, *7*, 4855.

(19) Mulazzi, M.; Chainani, A.; Katayama, N.; Eguchi, R.; Matsunami, M.; Ohashi, H.; Senba, Y.; Nohara, M.; Uchida, M.; Takagi, H.; others Absence of nesting in the charge-density-wave system 1 T-VS 2 as seen by photoelectron spectroscopy. *Physical Review B* **2010**, *82*, 075130.

(20) Lee, S.; Park, K.; Park, J.; Choi, J. B.; Yang, S.-R. E.; Yoo, K.-H.; Kim, J.; Park, S.;

Kim, K. Single-electron spectroscopy in a coupled triple-dot system: Role of interdot electron-electron interactions. *Physical Review B* **2000**, *62*, R7735.

(21) Kahk, J.; Poll, C.; Oropeza, F.; Ablett, J.; Céolin, D.; Rueff, J.; Agrestini, S.; Utsumi, Y.; Tsuei, K.; Liao, Y.; others Understanding the electronic structure of IrO 2 using hard-x-ray photoelectron spectroscopy and density-functional theory. *Physical review letters* **2014**, *112*, 117601.

(22) Glatzel, P.; Bergmann, U. High resolution 1s core hole X-ray spectroscopy in 3d transition metal complexes—electronic and structural information. *Coordination chemistry reviews* **2005**, *249*, 65–95.

(23) Harrison, R. J.; Fann, G. I.; Yanai, T.; Beylkin, G. Multiresolution Quantum Chemistry in Multiwavelet Bases. Computational Science — ICCS 2003. Berlin, Heidelberg, 2003; pp 103–110.

(24) Yanai, T.; Harrison, R. J.; Handy, N. C. Multiresolution quantum chemistry in multiwavelet bases: time-dependent density functional theory with asymptotically corrected potentials in local density and generalized gradient approximations. *Molecular Physics* **2005**, *103*, 413–424.

(25) Vence, N.; Harrison, R.; Krstić, P. Attosecond electron dynamics: A multiresolution approach. *Phys. Rev. A* **2012**, *85*, 033403.

(26) Yanai, T.; Fann, G. I.; Beylkin, G.; Harrison, R. J. Multiresolution quantum chemistry in multiwavelet bases: excited states from time-dependent Hartree–Fock and density functional theory via linear response. *Phys. Chem. Chem. Phys.* **2015**, *17*, 31405–31416.

(27) Jensen, S. R.; Flå, T.; Jonsson, D.; Monstad, R. S.; Ruud, K.; Frediani, L. Magnetic properties with multiwavelets and DFT: the complete basis set limit achieved. *Phys. Chem. Chem. Phys.* **2016**, *18*, 21145–21161.

(28) Brakestad, A.; Jensen, S. R.; Wind, P.; D'Alessandro, M.; Genovese, L.; Hopmann, K. H.; Frediani, L. Static Polarizabilities at the Basis Set Limit: A Benchmark of 124 Species. *Journal of Chemical Theory and Computation* **2020**, *16*, 4874–4882.

(29) Dirac, P. A. M. The Quantum Theory of the Electron. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* **1928**, *117*, 610–624.

(30) Anderson, J.; Sundahl, B.; Harrison, R.; Beylkin, G. Dirac-Fock calculations on molecules in an adaptive multiwavelet basis. *The Journal of Chemical Physics* **2019**, *151*, 234112.

(31) Mussard, B.; Sharma, S. One-Step Treatment of Spin–Orbit Coupling and Electron Correlation in Large Active Spaces. *Journal of Chemical Theory and Computation* **2018**, *14*, 154–165.

(32) Petrov, A. N.; Mosyagin, N. S.; Titov, A. V.; Tupitsyn, I. I. Accounting for the Breit interaction in relativistic effective core potential calculations of actinides. *Journal of Physics B: Atomic, Molecular and Optical Physics* **2004**, *37*, 4621.

(33) Vidal, M. L.; Pokhilko, P.; Krylov, A. I.; Coriani, S. Equation-of-Motion Coupled-Cluster Theory to Model L-Edge X-ray Absorption and Photoelectron Spectra. *The Journal of Physical Chemistry Letters* **2020**, *11*, 8314–8321.

(34) Kasper, J. M.; Stetina, T. F.; Jenkins, A. J.; Li, X. Ab initio methods for L-edge x-ray absorption spectroscopy. *Chemical Physics Reviews* **2020**, *1*.

(35) Breit, G. An Interpretation of Dirac's Theory of the Electron. *Proceedings of the National Academy of Sciences* **1928**, *14*, 553–559.

(36) Breit, G. Dirac's Equation and the Spin-Spin Interactions of Two Electrons. *Phys. Rev.* **1932**, *39*, 616–624.

(37) Moss, R. *Advanced molecular quantum mechanics: an introduction to relativistic quantum mechanics and the quantum theory of radiation*; Springer Science & Business Media, 2012.

(38) Dyall, K. G.; Fægri Jr, K. *Introduction to relativistic quantum chemistry*; Oxford University Press, 2007.

(39) Helgaker, T.; Coriani, S.; Jørgensen, P.; Kristensen, K.; Olsen, J.; Ruud, K. Recent Advances in Wave Function-Based Methods of Molecular-Property Calculations. *Chemical Reviews* **2012**, *112*, 543–631.

(40) Battistella, E.; Bjorgve, M.; Di Remigio, R.; Gerez, G.; Jensen, S. R. VAMPyR: Very Accurate Multiresolution Python Routines. 2021.

(41) Jönsson, P.; Gaigalas, G.; Bieroń, J.; Fischer, C. F.; Grant, I. P. New version: Grasp2K relativistic atomic structure package. *Comput. Phys. Commun.* **2013**, *184*, 2197–2203.

(42) Saue, T.; Bast, R.; Gomes, A. S. P.; Jensen, H. J. A.; Visscher, L.; Aucar, I. A.; Di Remigio, R.; Dyall, K. G.; Eliav, E.; Fasshauer, E.; Fleig, T.; Halbert, L.; Hedegård, E. D.; Helmich-Paris, B.; Iliaš, M.; Jacob, C. R.; Knecht, S.; Laerdahl, J. K.; Vidal, M. L.; Nayak, M. K.; Olejniczak, M.; Olsen, J. M. H.; Pernpointner, M.; Senjean, B.; Shee, A.; Sunaga, A.; van Stralen, J. N. P. The DIRAC code for relativistic molecular calculations. *J. Chem. Phys.* **2020**, *152*, 204104.

(43) Alpert, B.; Beylkin, G.; Gines, D.; Vozovoi, L. Adaptive Solution of Partial Differential Equations in Multiwavelet Bases. *Journal of Computational Physics* **2002**, *182*, 149–190.

(44) Alpert, B.; Beylkin, G.; Coifman, R.; Rokhlin, V. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM journal on Scientific Computing* **1993**, *14*, 159–184.

(45) Beylkin, G.; Cramer, R.; Fann, G.; Harrison, R. J. Multiresolution separated representations of singular and weakly singular operators. *Applied and Computational Harmonic Analysis* **2007**, *23*, 235–253.

(46) Beylkin, G.; Mohlenkamp, M. J. Algorithms for Numerical Analysis in High Dimensions. *SIAM J. Sci. Comput.* **2005**, *26*, 2133–2159.

(47) Frediani, L.; Fossgaard, E.; Flå, T.; Ruud, K. Fully adaptive algorithms for multivariate integral equations using the non-standard form and multiwavelets with applications to the Poisson and bound-state Helmholtz kernels in three dimensions. *Molecular Physics* **2013**, *111*, 1143–1160.

(48) Reiher, M.; Wolf, A. *Relativistic Quantum Chemistry: The Fundamental Theory of Molecular Science*; John Wiley & Sons, 2014.

(49) Sun, S.; Ehrman, J.; Sun, Q.; Li, X. Efficient evaluation of the Breit operator in the Pauli spinor basis. *The Journal of Chemical Physics* **2022**, *157*, 064112.

(50) Anderson, J.; Harrison, R. J.; Sekino, H.; Sundahl, B.; Beylkin, G.; Fann, G. I.; Jensen, S. R.; Sagert, I. On derivatives of smooth functions represented in multiwavelet bases. *Journal of Computational Physics: X* **2019**, *4*, 100033.

(51) Shiozaki, T. Communication: An efficient algorithm for evaluating the Breit and spin–spin coupling integrals. *The Journal of Chemical Physics* **2013**, *138*, 111101.

(52) Hackbusch, W.; Khoromskij, B. N. Low-rank Kronecker-product Approximation to Multi-dimensional Nonlocal Operators. Part I. Separable Approximation of Multivariate Functions. *Computing* **2006**, *76*, 177–202.

(53) Dyall, K. G. Relativistic and nonrelativistic finite nucleus optimized double zeta basis sets for the 4p, 5p and 6p elements. *Theoretical Chemistry Accounts* **1998**, *99*, 366–371.

(54) Dyall, K. G. Relativistic and nonrelativistic finite nucleus optimized triple-zeta basis sets for the 4 p, 5 p and 6 p elements. *Theoretical Chemistry Accounts* **2002**, *108*, 335–340.

(55) Kramers, H. Theotie ga&&le de la rotation paramagnetique dans les cristaux. Proc. Royal Acad. Amsterdam. 1930; p 959.

(56) Wigner, E. Über die Operation der Zeitumkehr in der Quantenmechanik, Gott. 1932.

(57) Saue, T.; Jensen, H.-J. *personal communication* **1996**,

(58) Visscher, L.; Dyall, K. DIRAC–FOCK ATOMIC ELECTRONIC STRUCTURE CALCULATIONS USING DIFFERENT NUCLEAR CHARGE DISTRIBUTIONS. *Atomic Data and Nuclear Data Tables* **1997**, *67*, 207–224.

(59) Saue, T. Relativistic Hamiltonians for Chemistry: A Primer. *ChemPhysChem* **2011**, *12*, 3077–3094.

(60) Jensen, S. R.; Saha, S.; Flores-Livas, J. A.; Huhn, W.; Blum, V.; Goedecker, S.; Frediani, L. The Elephant in the Room of Density Functional Theory Calculations. *The Journal of Physical Chemistry Letters* **2017**, *8*, 1449–1457.

(61) Thierfelder, C.; Schwerdtfeger, P. Quantum electrodynamic corrections for the valence shell in heavy many-electron atoms. *Physical Review A* **2010**, *82*, 062503.

(62) Halbert, L.; Vidal, M. L.; Shee, A.; Coriani, S.; Severo Pereira Gomes, A. Relativistic EOM-CCSD for Core-Excited and Core-Ionized State Energies Based on the Four-Component Dirac-Coulomb(-Gaunt) Hamiltonian. *Journal of Chemical Theory and Computation* **2021**, *17*, 3583–3598.

(63) Boudjemia, N.; Jänkälä, K.; Gejo, T.; Nagaya, K.; Tamasaku, K.; Huttula, M.; Piancastelli, M. N.; Simon, M.; Oura, M. Deep core photoionization of iodine in CH3I

and CF3I molecules: how deep down does the chemical shift reach? *Phys. Chem. Chem. Phys.* **2019**, *21*, 5448–5454.

(64) Ishikawa, Y.; Quiney, H. M. On the use of an extended nucleus in Dirac–Fock Gaussian basis set calculations. *International Journal of Quantum Chemistry* **1987**, *32*, 523–532.

(65) Visser, O.; Aerts, P.; Hegarty, D.; Nieuwpoort, W. The use of gaussian nuclear charge distributions for the calculation of relativistic electronic wavefunctions using basis set expansions. *Chemical Physics Letters* **1987**, *134*, 34–38.

(66) Oura, M.; Gejo, T.; Nagaya, K.; Kohmura, Y.; Tamasaku, K.; Journel, L.; Piancastelli, M. N.; Simon, M. Hard x-ray photoelectron spectroscopy on heavy atoms and heavy-element containing molecules using synchrotron radiation up to 35 keV at SPring-8 undulator beamlines. *New Journal of Physics* **2019**, *21*, 043015.

5.4 Paper IV: Cavity-free continuum solvation: implementation and parametrization in a multiwavelet framework

Abstract

We introduce a novel approach to continuum solvation models via a multiwavelet-based framework that integrates a quantum/classical polarizable continuum model. Departing from traditional methods, this model applies a diffuse interface between the solute and solvent, characterized by a variable permittivity, thereby obviating the sharp-boundary assumption prevalent in several extant solvation models. Our implementation effectively incorporates both surface and volume polarization effects within the quantum/classical interface, adhering to tight precision constraints facilitated by the adaptability of the multiwavelet technique. The model excels in simulating intricate solvent interactions without necessitating *a posteriori* volumetric polarization adjustments. Comparative analyses with established sharp-boundary continuum models on the Minnesota solvation database demonstrate our model's alignment, particularly in the calculated polarization energies.

Personal Contributions:

- Contributed to enhancing the `VAMPyR` library by adding critical features crucial for pioneering the first version of the solvation model code.
- Collaborated in conceptualizing how the `VAMPyR` library would be utilized in the implementation, signifying a sound methodological foundation for the research.

JCTC Journal of Chemical Theory and Computation

pubs.acs.org/JCTC Article

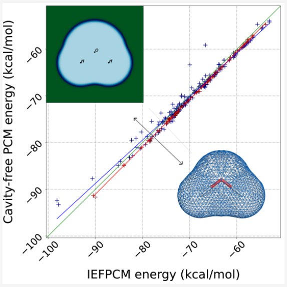
Cavity-Free Continuum Solvation: Implementation and Parametrization in a Multiwavelet Framework

Gabriel A. Gerez S, Roberto Di Remigio Eikås,* Stig Rune Jensen, Magnar Bjørgve, and Luca Frediani*

Cite This: *J. Chem. Theory Comput.* 2023, 19, 1986−1997 Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: We present a multiwavelet-based implementation of a quantum/classical polarizable continuum model. The solvent model uses a diffuse solute−solvent boundary and a position-dependent permittivity, lifting the sharp-boundary assumption underlying many existing continuum solvation models. We are able to include both surface and volume polarization effects in the quantum/classical coupling, with guaranteed precision, due to the adaptive refinement strategies of our multiwavelet implementation. The model can account for complex solvent environments and does not need *a posteriori* corrections for volume polarization effects. We validate our results against a sharp-boundary continuum model and find a very good correlation of the polarization energies computed for the Minnesota solvation database.



1. INTRODUCTION

Continuum solvation models have been used in quantum chemistry for half a century.[1−4] Their use is motivated by the need to simulate the effect of a large solvent environment on a molecular solute, keeping at the same time the computational cost to a minimum.

Several models and flavors have throughout the years been developed. Common to essentially all such models are two basic assumptions: 1) the solvent degrees of freedom can be conveniently described in terms of a continuum, parametrized using macroscopic properties of the solvent; 2) the quantum system is confined inside a cavity and the solute−solvent interaction is described in terms of functions (charge density/potential) supported on the cavity surface. Whereas the former assumption is a physical one, giving a prescription for the underlying physical laws,[5] the latter is a convenient mathematical formulation, which reduces the computational cost transforming a three-dimensional problem in the whole space to a two-dimensional one on the boundary of the molecular cavity. Despite the convenience, a sharp boundary between neighboring molecules assumes that no electronic density is present beyond the cavity surface. This is not physically sound, because electronic densities of solute and solvent in reality overlap. Initially, this issue has been dealt with by simple renormalization procedures:[3] more elaborate corrections have later been proposed,[6−8] and for the Integral Equation Formalism (IEF) formulation of the polarizable continuum model (PCM) it can be shown that a first-order correction is already included in the model.[9] A full account of this issue is, however, not practical in terms of a surface model, and the ever increasing basis sets employed in routine calculations, including very diffuse functions, aggravate the problem further by allowing more and more of the electron density to "escape" the cavity.

Neglecting electronic charge overlap between solute and solvent does not only impact the electrostatic energy: excitation energies depend on the charge distribution in the excited states, which is invariably more diffuse than in the ground state, and other interaction terms, such as the repulsion energy, depend explicitly on the overlap between solute and solvent densities.[10]

The parametric description of the cavity surface also presents challenges, not only from a formal point of view to define the correct cavity boundary,[2,3] but also from a technical standpoint, especially for larger molecules. The development of stable cavity generators is still an active area of research.[11−20]

In recent years, several real-space methods for quantum chemistry have been developed,[21−26] and with these, the treatment of solvation as a three-dimensional problem has become a feasible alternative. The advantage is a seamless integration with the quantum mechanical implementation: the electrostatic potential is no longer computed in a vacuum but in the generalized dielectric medium with a position-dependent

Received: November 4, 2022
Published: March 18, 2023

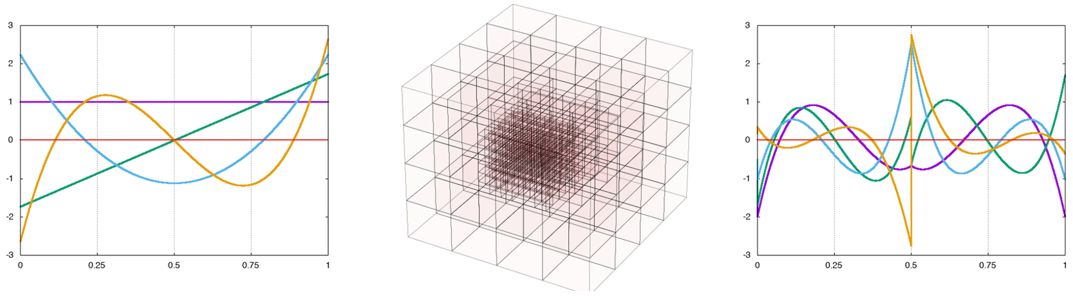


Figure 1. Left panel: scaling functions of order $k = 3$ defined in the interval [0,1] are simple polynomials. Right panel: the corresponding wavelet functions are piece-wise polynomials with four vanishing moments (orthogonal to polynomials up to the cubic one). Central panel: adaptive grids are constructed on demand to minimize storage and meet precision requirements.

permittivity. Several real-space codes have so far adopted this strategy.[27−32] Another advantage of this approach is an increased flexibility: no constraints are placed on the form of the permittivity function, and complex environments consisting of surfaces, droplets, membranes, can be treated without the need of ad-hoc implementations, which are often limited to a handful of special cases.[33−35]

In this contribution, we will present our implementation, which makes use of a multiwavelet (MW) framework[36−39] to solve both the Kohn−Sham (KS) equations of density functional theory (DFT)[40,41] and the Generalized Poisson Equation (GPE)[27] for the solvent reaction potential. We will also show a set of benchmark calculations to showcase the implementation's theoretical correctness, parametrization, and flexibility. MWs constitute a basis that can give accurate results up to a user-defined precision, thanks to an automatic adaptive refinement.[39] Our implementation is included in the open-source MW computational chemistry software package MRChem.[26] The combination of MW-based KS-DFT and GPE solver provides a methodology for the assessment of solvent effects with controlled precision.

2. THEORY

In the theoretical framework adopted in this work, molecules are described through quantum mechanics, whereas the solvent is modeled as a classical entity, described by macroscopic properties. The two subsystems are connected by the solute−solvent interaction, which describes the mutual polarization of the two subsystems.[2,3] Such an interaction is described by classical electrostatics. In almost all implementations, the quantum and the classical problem are solved with very different methods: the most widely used approach makes use of Boundary Element Method (BEM)[42] techniques to solve the electrostatic problem (environment) and Gaussian Type Orbital (GTO) bases[43,44] to describe the quantum problem. The use of Multiwavelets offers a unique opportunity to treat both problems with the same tools and methods. We will here recap the basic concepts of multiresolution analysis (MRA) and how it is employed to solve the electrostatic and the quantum problem.

2.1. Multiresolution Analysis and Multiwavelets.

MRA is a mathematical framework that considers a space spanned by a basis of functions with self-similarity and regularity properties.[45] In practice, all basis functions are constructed by simple translation and dilation of a small set of starting functions $\phi(x)$:

$$\phi_l^n(x) = 2^{n/2}\phi(2^n x - l) \tag{1}$$

The core idea of MRA is that the space spanned by the basis functions at a given scale n is a subspace of those at scale $n + 1$. Such a *ladder of spaces* can be extended indefinitely, and its limit is by construction dense in L^2. Successive refinements thus provide a systematic strategy to reach completeness, with a handful of predefined functions. This is in stark contrast with traditional GTO methods, where extending a basis requires a complete reparametrization of the basis set, atom by atom. The *wavelet* functions are obtained by taking the difference between two consecutive scaling spaces, and they convey information about the error incurred at each scale n due to neglecting the refinement at scale $n + 1$, see Figure 1 for a 1-dimensional illustration.

As long as the fundamental properties of *self-similarity* and *completeness* are preserved, the choice of a specific basis set can be guided by numerical considerations to obtain compact representation of functions and efficient application of operators.

Alpert's Multiwavelets[36] constitute a practical realization of MRA by considering a set of polynomial functions (e.g., Legendre or Interpolating polynomials) defined on an interval. The main advantages of Multiwavelets are the simplicity of the original basis (a polynomial set) and the disjoint support (basis functions are zero outside their support node).[37] The latter enables adaptive refinement of functions to minimize the storage needs and the computational overhead. The extension to three-dimensional functions is obtained by tensor-product methods, and operators are efficiently applied in a separated form.[38]

Multiwavelets are an ideal framework to deal with integral operators, and this allows both the KS equations for the quantum system[40] and the Poisson equation for the solvent polarization[27] to be solved within the same formalism, once the equations are converted from the conventional differential form to the appropriate integral form. Functions are projected/computed on an adaptive grid to guarantee the requested precision. All operations (operator applications, algebraic manipulations) are defined within the requested precision, in such a way that the developer can easily implement new algorithms with little effort[46] and the end-user only needs to specify the requested precision.[26,47−49]

For details about how to solve the KS equations within a MW framework, we refer to the literature.[39−41] Concerning the

GPE, we will expose the derivation and the implementation details in the remainder of this section.

2.2. Electrostatics of Continuous Media. Any material is a bound aggregate of nuclei and electrons: at microscopic level these charged particles obey the microscopic Maxwell equations. We are, however, interested in the *macroscopic* behavior of the material in the presence of external sources of charge $\rho(\boldsymbol{r})$ and current $\boldsymbol{j}(\boldsymbol{r})$. Following Jackson,[5] we can perform a spatial average to arrive at the *macroscopic* Maxwell equations:

$$\begin{cases} \nabla\cdot\boldsymbol{D} = 4\pi\rho \\ \nabla\times\boldsymbol{H} - \dfrac{1}{c}\dfrac{\partial D}{\partial t} = \dfrac{4\pi}{c}\boldsymbol{j} \\ \nabla\times\boldsymbol{E} + \dfrac{1}{c}\dfrac{\partial\boldsymbol{B}}{\partial t}t = 0 \\ \nabla\cdot\boldsymbol{B} = 0 \end{cases} \tag{2}$$

These equations are expressed in terms of the usual electric and magnetic fields, $\boldsymbol{E}$ and $\boldsymbol{B}$, and additionally the *displacement* $\boldsymbol{D}$ and *magnetization* $\boldsymbol{H}$ fields appear as a result of the spatial averaging. In the quasistatic limit, the electric field has zero curl and can thus be written in terms of a scalar potential function: $\boldsymbol{E} = -\nabla V$, where V is the electrostatic potential. To relate the external sources to the potential it is first necessary to relate the fields $\boldsymbol{E}$ and $\boldsymbol{D}$ with a *constitutive relation*,[5,50] which is, in general, a nonlinear and space-time nonlocal relationship between the fields. For linear and local continuous media the constitutive relation is

$$\boldsymbol{D} = \boldsymbol{\varepsilon}(\boldsymbol{r})\boldsymbol{E} \tag{3}$$

where the permittivity $\boldsymbol{\varepsilon}(\boldsymbol{r})$ is a position-dependent, rank-3 symmetric tensor. Upon inserting the constitutive relation into the first of Maxwell's equations, we obtain the GPE:

$$\nabla\cdot[\boldsymbol{\varepsilon}(\boldsymbol{r})\nabla V] = -4\pi\rho \tag{4}$$

In the following, we will further specialize to the isotropic case $\boldsymbol{\varepsilon}(\boldsymbol{r}) = \varepsilon(\boldsymbol{r})\boldsymbol{I}$, with $\boldsymbol{I}$ the rank-3 identity:

$$\nabla\cdot[\varepsilon(\boldsymbol{r})\nabla V] = -4\pi\rho \tag{5}$$

We remark that the permittivity is still position-dependent, in contrast to the usual PCM treatment. The solution to eq 5 can be partitioned as

$$V = V_\rho + V_R = \int_{\mathbb{R}^3}\frac{\rho(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}\mathrm{d}\boldsymbol{r}' + V_R \tag{6}$$

where V_ρ is the electrostatic potential in a vacuum and V_R is the *reaction potential*. The *polarization energy* is then defined as

$$U_{\mathrm{pol}} = \frac{1}{2}\int \mathrm{d}\boldsymbol{r}\rho(\boldsymbol{r})V_R[\rho](\boldsymbol{r}) \tag{7}$$

We write the reaction potential as a functional of the charge density: the functional dependence is linear.[9]

2.3. The Quantum-Classical Coupling. Our quantum mechanical treatment of the system will be based on KS-DFT. For an N-electron system coupled with a classical polarizable continuum environment, the KS-DFT *free* energy[2] functional[51,52] reads:

$$\begin{aligned} \mathcal{G}[\rho] &= T_s[\rho_e] + V_{\mathrm{Ne}}[\rho_e] + J[\rho_e] - \zeta K[\rho_e] + E_{\mathrm{xc}}[\rho_e, \nabla\rho_e] \\ &\quad + U_{\mathrm{pol}}[\rho] + \frac{1}{2}\sum_{\alpha\neq\beta}^{N_{\mathrm{nuclei}}}\frac{Z_\alpha Z_\beta}{|\boldsymbol{R}_\alpha - \boldsymbol{R}_\beta|} \\ &= \int\mathrm{d}\boldsymbol{r}\left[-\frac{1}{2}\nabla^2\rho_1(\boldsymbol{r},\boldsymbol{r}')\right]_{\boldsymbol{r}'=\boldsymbol{r}} + \int\mathrm{d}\boldsymbol{r}V_{\mathrm{Ne}}(\boldsymbol{r})\rho_e \\ &\quad + \frac{1}{2}\int\mathrm{d}\boldsymbol{r}\int\mathrm{d}\boldsymbol{r}'\frac{\rho_e(\boldsymbol{r})\rho_e(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} - \frac{\zeta}{2}\int\mathrm{d}\boldsymbol{r} \\ &\quad \int\mathrm{d}\boldsymbol{r}'\frac{\rho_1(\boldsymbol{r},\boldsymbol{r}')\rho_1(\boldsymbol{r}',\boldsymbol{r})}{|\boldsymbol{r}-\boldsymbol{r}'|} + E_{\mathrm{xc}}[\rho_e, \nabla\rho_e] \\ &\quad + \frac{1}{2}\int\mathrm{d}\boldsymbol{r}\rho(\boldsymbol{r})V_R[\rho](\boldsymbol{r}) + \frac{1}{2}\sum_{\alpha\neq\beta}^{N_{\mathrm{nuclei}}}\frac{Z_\alpha Z_\beta}{|\boldsymbol{R}_\alpha - \boldsymbol{R}_\beta|} \end{aligned} \tag{8}$$

The molecular charge density is separated into electronic and nuclear components:

$$\rho(\boldsymbol{r}) = \rho_e(\boldsymbol{r}) + \sum_{\alpha}^{N_{\mathrm{nuclei}}} Z_\alpha\delta(\boldsymbol{r} - \boldsymbol{R}_\alpha) \tag{9}$$

$E_{\mathrm{xc}}[\rho_e, \nabla\rho_e]$ is a GGA exchange-correlation functional, and the nuclear-electron potential is defined as

$$V_{\mathrm{Ne}}(\boldsymbol{r}) = -\sum_{\alpha=1}^{N_{\mathrm{nuclei}}}\frac{Z_\alpha}{|\boldsymbol{R}_\alpha - \boldsymbol{r}|} \tag{10}$$

ζ is a scalar factor influencing the portion of exact exchange included in the energy. The 1-body reduced density matrix (RDM) and electronic density function appear in the energy expression:

$$\rho_1(\boldsymbol{r},\boldsymbol{r}') = \sum_{i=1}^{N}\phi_i(\boldsymbol{r})\phi_i^*(\boldsymbol{r}'), \qquad \rho_e(\boldsymbol{r}) \equiv \rho_1(\boldsymbol{r},\boldsymbol{r}) \tag{11}$$

The minimum is found by constrained optimization, to enforce idempotency and normalization of the RDM:

$$\min_{\rho_e}\mathcal{G}[\rho] \text{ such that } \begin{cases} \displaystyle\int\mathrm{d}\boldsymbol{r}''\rho_1(\boldsymbol{r},\boldsymbol{r}'')\rho_1(\boldsymbol{r}'',\boldsymbol{r}') = \rho_1(\boldsymbol{r},\boldsymbol{r}') \\ \displaystyle\int\mathrm{d}\boldsymbol{r}\rho_e = N \end{cases} \tag{12}$$

and leads to the variational condition:[51,53]

$$[F, \rho_e] = 0 \tag{13}$$

where the effective one-electron Fock operator appears:

$$\begin{aligned} F(\boldsymbol{r},\boldsymbol{r}') &= \frac{\delta\mathcal{G}}{\delta\rho_1(\boldsymbol{r},\boldsymbol{r}')} \\ &= \left[-\frac{1}{2}\nabla^2 + V_{\mathrm{Ne}}(\boldsymbol{r})\right]\delta(\boldsymbol{r}-\boldsymbol{r}') + \delta(\boldsymbol{r}-\boldsymbol{r}') \\ &\quad \left[\int\mathrm{d}\boldsymbol{r}'\frac{\rho_e(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}\right] \\ &\quad - \frac{\zeta}{|\boldsymbol{r}-\boldsymbol{r}'|}\rho_1(\boldsymbol{r},\boldsymbol{r}') + \frac{\delta E_{\mathrm{xc}}}{\delta\rho_1(\boldsymbol{r},\boldsymbol{r}')} + V_R[\rho(\boldsymbol{r})] \end{aligned} \tag{14}$$

2.4. Solving the Generalized Poisson Equation. The solution to the GPE is a function supported on the entire space

$\mathbb{R}^3$. Apparent surface charge formulations of continuum solvation models do not solve eq 5 directly, but rather reformulate it as a boundary integral equation and solve it by boundary-element discretization. The apparent surface charge, supported on the closed solute–solvent boundary, is the sought-after quantity to compute the polarization energy.[9] Such a procedure is generally based on two underlying assumptions: (1) the charge density is entirely contained inside the cavity boundary, and (2) the permittivity is unitary inside the cavity and constant outside the cavity, with a jump condition that defines the electrostatic potential and field across the cavity boundary. With a real-space approach both assumptions can be relaxed and the equation can be solved directly. We recap here the procedure outlined by Fosso-Tande and Harrison.[27]

We rewrite eq 5 in terms of the Laplacian of the potential V:

$$\nabla^2 V = -\frac{4\pi\rho}{\varepsilon(\boldsymbol{r})} - \frac{\nabla\varepsilon(\boldsymbol{r})\cdot\nabla V}{\varepsilon(\boldsymbol{r})} \tag{15}$$

The second term on the right-hand side contains both the gradient of the permittivity and the gradient of the potential. When the permittivity is not constant, the equation cannot be solved in one step by inversion of the Laplacian, i.e., by convolution of the right-hand side with the Laplacian's Green's function. An iterative strategy must be employed instead.

Let us then define the effective charge:

$$\rho_{\text{eff}} = \frac{\rho}{\varepsilon} \tag{16}$$

and the polarization function:

$$\gamma = \frac{1}{4\pi}\frac{\nabla\varepsilon\cdot\nabla V}{\varepsilon} = \frac{\nabla\log\varepsilon\cdot\nabla V}{4\pi} \tag{17}$$

such that eq 15 becomes

$$\nabla^2 V = -4\pi(\rho_{\text{eff}} + \gamma) \tag{18}$$

We can now formally solve eq 15 in terms of the Laplacian's Green's function:

$$V(\boldsymbol{r}) = \int \mathrm{d}\boldsymbol{r}'\frac{\rho_{\text{eff}}(\boldsymbol{r}') + \gamma(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} = \frac{1}{|\boldsymbol{r}-\boldsymbol{r}'|}*(\rho_{\text{eff}} + \gamma) \tag{19}$$

However, both the polarization energy in eq 7 and the solute–solvent interaction term in the Fock operator are expressed in terms of the reaction potential, rather than the total electrostatic potential. By making use of the partition of V in eq 6 and recalling that $\nabla^2 V_\rho = -4\pi\rho$ one obtains

$$\nabla^2 V_{\text{R}} = -4\pi\left[\rho\left(\frac{1-\varepsilon}{\varepsilon}\right) + \gamma\right] \tag{20}$$

which can be formally inverted using the Poisson kernel:

$$V_{\text{R}} = \frac{1}{|\boldsymbol{r}-\boldsymbol{r}'|}*\left[\rho\left(\frac{1-\varepsilon}{\varepsilon}\right) + \gamma\right] \tag{21}$$

We stress that γ is a function of $V = V_\rho + V_{\text{R}}$ and eq 21 must therefore be solved iteratively.

3. IMPLEMENTATION

In this section we present details about our specific choice of parametrization for the permittivity and how we compute the electrostatic potential between solute and solvent. We also show how we couple this to a standard self-consistent field (SCF) optimization procedure.

3.1. The Permittivity Function Parametrization. We partition space into two regions: a cavity containing the solute, and the remainder. The cavity surface is defined as the union set of a collection of interlocking spheres centered on the nuclei. Their radii are parametrized by using the corresponding van der Waals radii times a factor. This factor is often set to either 1.1 or 1.2,[2] but it might vary, e.g., depending on the charge of the solute. For standard continuum models the cavity boundary is the support of the electrostatic problem for the solute–solvent interaction. In the current model it serves as a support to define the parametrization of the position-dependent $\varepsilon(\boldsymbol{r})$. In Section 4 the appropriate parametrization of the cavity for the present model will be discussed.

Following Fosso-Tande and Harrison, we write the permittivity as a function of the molecular cavity function:[27]

$$\varepsilon(\boldsymbol{r}) = \varepsilon_{\text{in}}\exp\left[\left(\log\frac{\varepsilon_{\text{out}}}{\varepsilon_{\text{in}}}\right)(1 - C(\boldsymbol{r}))\right] \tag{22}$$

The exponential parametrization proves convenient in light of the definition of γ in eq 17, which lets us define its gradient using the cavity function, $C(\boldsymbol{r})$, only.

The molecular cavity function is constructed as follows. For each sphere α centered at $\boldsymbol{r}_\alpha$ with radius R_α, we can measure the signed normal distance of any point in space as

$$s_\alpha(\boldsymbol{r}) = |\boldsymbol{r} - \boldsymbol{r}_\alpha| - R_\alpha \tag{23}$$

Given $s_\alpha(\boldsymbol{r})$, we define a smoothed boundary of the sphere as

$$C_\alpha(\boldsymbol{r}) = \frac{1}{2}\left[1 + \operatorname{erf}\left(-\frac{s_\alpha(\boldsymbol{r})}{\sigma}\right)\right] \tag{24}$$

where σ is a user-defined smoothing parameter: C_α approaches the Heaviside step function as $\sigma \to 0$. The molecular cavity function is then a product of all N spheres:

$$C(\boldsymbol{r}) = 1 - \prod_{\alpha=1}^{N_{\text{sph}}}(1 - C_\alpha(\boldsymbol{r})) \tag{25}$$

see Figure 2 for an example.

The log-derivative of the permittivity in eq 17 is then:

$$\nabla\log\varepsilon(\boldsymbol{r}) = \left(\log\frac{\epsilon_{\text{in}}}{\epsilon_{\text{out}}}\right)\nabla C(\boldsymbol{r}) \tag{26}$$

requiring evaluation of the gradient of the cavity function. For interlocking-spheres cavities, a closed-form analytical expression is available, see Appendix A, and is implemented in our code. Note, however, that, in a real-space, multiwavelet framework, we can compute this gradient by direct application of the derivative operator,[54] which allows one to use more complex or even numerical definitions of the boundary, e.g., as isodensity surfaces.

3.2. The Self-Consistent Reaction Field. The self-consistent reaction field (SCRF) is the iterative procedure to solve the GPE for any given molecular density. At convergence, the iterations produce the reaction potential V_{R}, which can be directly employed in the solution of the KS-DFT equations.

Algorithm 1 shows the iterative procedure implemented to solve the GPE within the SCF iterations. The input parameters at iteration n are the charge density $\rho^{[n]}$, the permittivity $\varepsilon(\boldsymbol{r})$, a

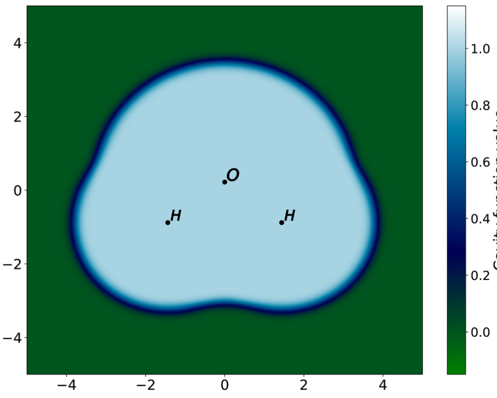


Figure 2. Cross-section in the xy plane of the cavity function $C(\boldsymbol{r})$ for the water molecule. Atom positions are indicated by their symbol. Coordinates are in atomic units. We can observe the smooth boundary of the cavity function.

guess for the reaction potential ${V_R}^{[n,0]}$, and a threshold parameter δ. Before iterating, the effective density ${\rho_{\mathrm{eff}}}^{[n]}$ and the potential ${V_\rho}^{[n]}$ are computed. At each microiteration i, the reaction potential ${V_R}^{[n,i]}$ is computed in four steps as outlined in lines 5−8 of Algorithm 1, and convergence in the norm of the reaction potential is checked against the threshold δ. At the first SCF iteration, the starting guess for the reaction potential is set to zero (${V_R}^{[0,0]} = 0$). At all subsequent iterations, the starting guess is set to the converged reaction potential from the previous iteration: ${V_R}^{[n,0]} = {V_R}^{[n-1]}$.

Algorithm 1 Self-consistent optimization of the reaction field. n is the SCF iteration index.

```
1: procedure SCRF MICROITERATION($\rho^{[n]}$, $\varepsilon(\boldsymbol{r})$, $V_{\mathrm{R}}^{[n,0]}$, $\delta$)
2:     $\rho_{\mathrm{eff}}^{[n]} = \frac{\rho^{[n]}}{\varepsilon}$
3:     $V_\rho^{[n]} = \frac{1}{|\boldsymbol{r}-\boldsymbol{r}'|} \star \rho^{[n]}$
4:     while $i < N_{\mathrm{micro}}$ do
5:         $V = V_{\mathrm{R}}^{[n,i]} + V_\rho^{[n]}$
6:         $\gamma = \frac{\nabla \log \varepsilon(\boldsymbol{r}) \cdot \nabla V^{[n,i]}}{4\pi}$
7:         $V_{\mathrm{R}} = \frac{1}{|\boldsymbol{r}-\boldsymbol{r}'|} \star \left[\rho_{\mathrm{eff}}^{[n]} - \rho^{[n]} + \gamma\right]$
8:         $V_{\mathrm{R}}^{[n,i+1]} = \mathrm{KAIN}(V_{\mathrm{R}}, V_{\mathrm{R}}^{[n,i]}, \dots V_{\mathrm{R}}^{[n,i-k]})$
9:         if $||V_{\mathrm{R}}^{[n,i+1]} - V_{\mathrm{R}}^{[n,i]}|| < \delta$ then
10:            return $V_{\mathrm{R}}^{[n]} := V_{\mathrm{R}}^{[n,i+1]}$
11:        end if
12:    end while
13: end procedure
```

A straightforward implementation of the microiterations suffers from slow convergence of the reaction potential, thus adding a significant prefactor to each SCF iteration. We use the Krylov-accelerated inexact Newton (KAIN) method,[41] which is a convergence acceleration technique, similar to Pulay's DIIS[55] and Anderson's mixing.[56] At each microiteration i, the updated reaction potential ${V_R}^{[i+1]}$ is constructed as a linear combination, with constraints, of N previous iterates. The KAIN history length N impacts both convergence and memory: $N = 5$ is generally a good compromise between fast convergence (fewer iterations) and acceptable memory footprint.

The KAIN acceleration is combined with an adaptive threshold to improve the convergence rate of the microiterations: instead of converging the reaction potential to the same predefined threshold ϵ used for the orbitals, we make use of a threshold, δ, chosen to be the norm of the orbital update in the parent SCF macroiteration. δ is thus updated during the SCF procedure. There are two parameters that affect the convergence pattern of the reaction potential, V_R:

1. The guess for V_R at the start of the microiterations:

(**A**) ${V_R}^{[n,0]} = 0$, or (**B**) ${V_R}^{[n,0]} = {V_R}^{[n-1]}$ (and zero for the first microiteration embedded in the first macroiteration).

2. The convergence threshold for the microiterations: (**C**) fixed threshold δ, or (**D**) dynamic threshold $\delta^{[n]} = |\Delta\rho^{[n]}|$.

These lead to four possible convergence regimes: **AC**, **BC**, **AD**, **BD**; the latter being our default.

Figure 3 illustrates how the number of microiterations evolves. A dynamic precision threshold **D** reduces the number

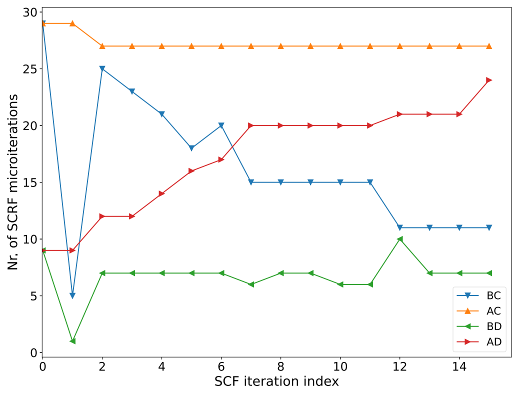


Figure 3. Convergence regimes for the SCRF algorithm. MW calculations with global precision 10^{-5} for acetamide (C_2H_5NO, identifier `0233ethb` from the Minnesota Solvent Descriptor Database). Four possible convergence scenarios are presented: static (**A**) or dynamic (**B**) precision threshold for the microiterations; zero initial guess (**C**) or guess from previous macroiteration (**D**). A dynamic threshold (green and red curves) reduces the number of microiterations at the beginning of the SCF procedure. A starting guess from the previous SCF macroiteration (green and blue curves) is effective close to convergence. Combining the two (green curve) is the optimal strategy. The dip observed for the blue and green curves at macroiteration 1 is due to the fact that the macroiteration 0 is a preliminary step and the orbital are not changed progressing from macroiteration 0 to macroiteration 1, but the convergence threshold is tightened. This results in an almost converged reaction potential as a starting guess for the microiterations nested in macroiteration 1.

of microiterations in the beginning of the SCF procedure, simply because the threshold for convergence is looser. Using the converged V_R from the previous macroiteration **B** helps close to SCF convergence, because the orbitals do not change much and the starting guess for the microiterations is also better. Combining those two choices results in the optimal convergence pattern: the convergence threshold is progressively tighter, while at the same time the starting guess for the reaction potential improves. The opposite choice (**AC** instead of **BD**) requires a large number of microiterations throughout, whereas the intermediate choices (**AD** and **BC**) result in a large number of iterations at the beginning (**BC**) or at the end (**AD**). We underline that all four choices converge to the same result for the example in Figure 3, but we can envisage cases

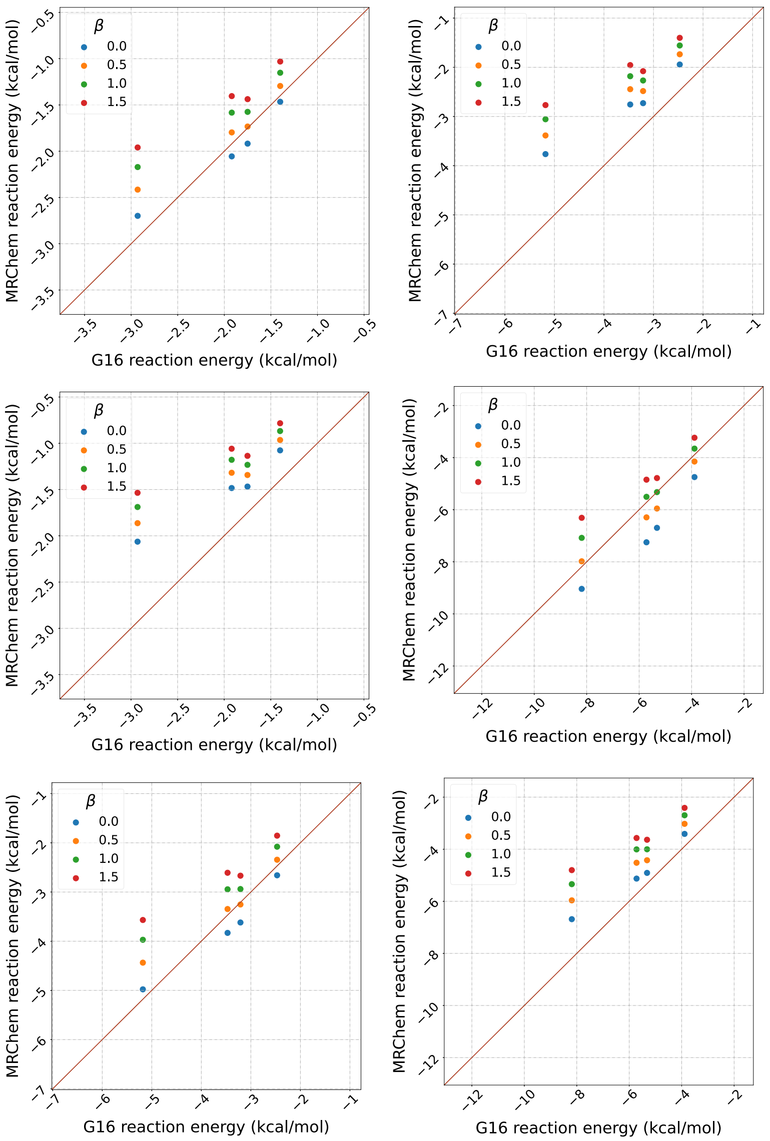


Figure 4. Results for the cavity parametrization. Left column: $\alpha = 1.1$. Right column: $\alpha = 1.2$. On each row a different permittivity is used: from top to bottom: $\varepsilon = 2.0$, 4.0, 80.0. For each plot there are four sets of data, corresponding to $\beta = 0.0$, 0.5, 1.0, 1.5. Each point on the set represents a molecule. x-Axis: the reaction energy calculated using Gaussian16. y-Axis: the reaction energy calculated using MRChem. Values are in Hartree.

where convergence could potentially be prevented by choices **A** and **C**.

4. RESULTS

For all systems, the solvation energies have been computed with both Gaussian16[57] and MRChem. Gaussian16 features the Integral Equation Formalism PCM (IEFPCM)[58] with a sharp cavity boundary. MRChem features the solvation model described in the previous sections.

Two sets of calculations have been performed. The aim of the first set was to determine a good parametrization for the cavity surface in terms of the atomic radii and the cavity surface thickness. Once a satisfactory parametrization was achieved, an extensive benchmark of solvation energies was performed, by considering the Minnesota Solvent Descriptor Database (MSDD) of Marenich et al.[59]

All calculations reported are KS-DFT using the PBE0 functional.[60] Gaussian16 results are obtained with the `Def2-TZVP`,[61−63] basis set, except where otherwise stated. MRChem

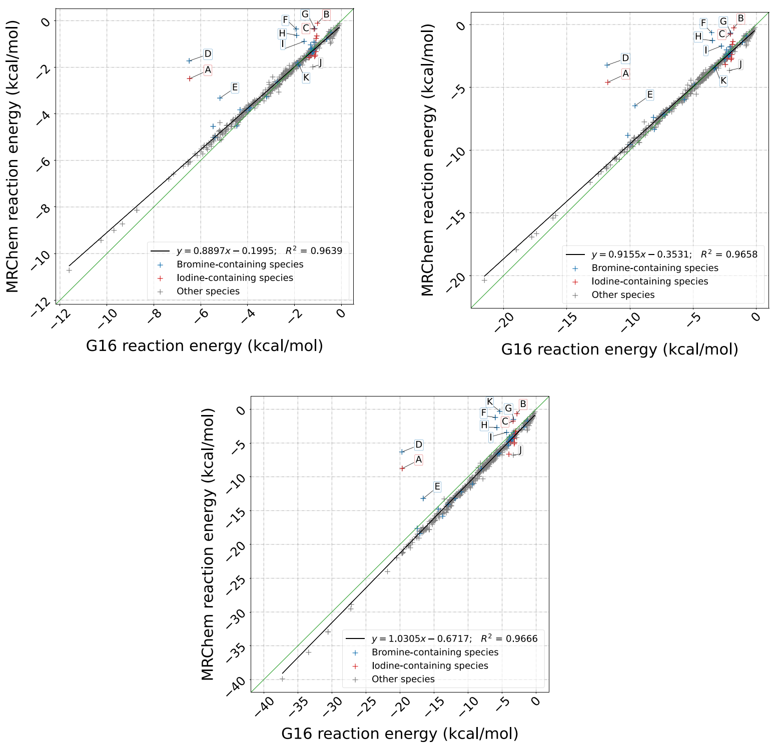


Figure 5. Correlation plots of reaction energies computed with Gaussian16 and MRChem for all neutral species in the MSDD[59] for ε = 2.0, 4.0, 80.0. All cavities are atom-centered, with Bondi radii.[66,67] Radii are scaled by 1.1. in Gaussian 16. For the MRChem calculations, we used default values: α = 1.1, β = 0.5, σ = 0.2 au Linear regression line shown in black. Outlier species are marked in blue and red when containing bromine and iodine, respectively. The labels refer to A. 5-bromouracil, $H_3C_4N_2O_2Br$ (`n203`); B. 5-bromo-3-s-butyl-6-methyl-uracil, $H_{13}C_9N_2O_2Br$ (`test1013`); C. 2-bromoanisole, H_7C_7OBr (`test5008`); D. Bromobenzene, H_5C_6Br (`0186bro`); E. 4-bromopyridine, H_4C_5NBr (`0573bro`); F. 1-bromo-2-chloroethane, H_4C_2ClBr (`0202bro`); G. 5-iodouracil, $H_3C_4N_2O_2I$ (`test2018`); H. iodomethane, H_3CI (`test4003`); I. iodobenzene, H_5C_6I (`test4001`); J. 1,4-dichlorobenzene, $H_4C_6Cl_2$ (`0176pdi`); K. 3-bromoanisole, H_7C_7OBr (`test5009`).

results are obtained setting the global precision parameter to 10^{-5}. In other words, the obtained absolute energy is correct with at least five digits with respect to the complete basis set (CBS) limit.[47] This is not to be confused with the convergence threshold of a SCF calculation performed with an atomic basis set, which will guarantee the "exact" result within the chosen basis, but where the precision compared to the CBS is limited by the choice of basis.

4.1. Cavity Parametrization.

For the parametrization calculations, 4 molecules of different levels of polarity were chosen: water, ethanol, formaldehyde, and ethyne (geometries taken from the MSDD,[59] file names `0217wat`, `0045eth`, `0069met`, and `0030eth`). No geometry optimization was performed. They were chosen to give a minimal set of neutral (polar and apolar) systems, to allow for a reliable yet simple data set to identify a good choice of the parameters defining the cavity.

In Gaussian16, the external iteration procedure[64,65] was used to extract the reaction energy from the total energy.[a] The spheres used for the cavities were atom-centered and used the atoms' Bondi radii[66] scaled by a factor of 1.1, as is standard for Gaussian16. Three different permittivities have been employed: 2.0, 4.0, and 80.0.

In MRChem, the cavity is also built from atom-centered spheres, with each radius R_i parametrized as

$$R_i = \alpha_i R_i^{\text{vdW}} + \beta_i \sigma_i \tag{27}$$

where R_i^{vdW} is the Bondi radius[66,67] of the i-th atom, σ_i is the width of the cavity boundary, and α_i and β_i are adjustable parameters. We allowed for granular, sphere-by-sphere flexibility in our implementation of the cavity function. By default, one value is used for each parameter (α, β, σ) for *all* spheres. The combination α = 1.1 and β = 0.0 would yield matching radii between MRChem and Gaussian16. In the following, we explored results when α values were 1.0, 1.1, 1.2, 1.3 and for β values of 0.0, 0.5, 1.0, 1.5. In all MRChem calculations the width parameter was fixed to σ = 0.2 au.

The aim of the parametrization is to see how the cavity width σ affects the results of our calculations, compared to a sharp-boundary method, and to choose the combination of α and β coefficients that provides a good correlation between our method and a sharp boundary implementation. The goal is not to replicate results from Gaussian16 implementation: our

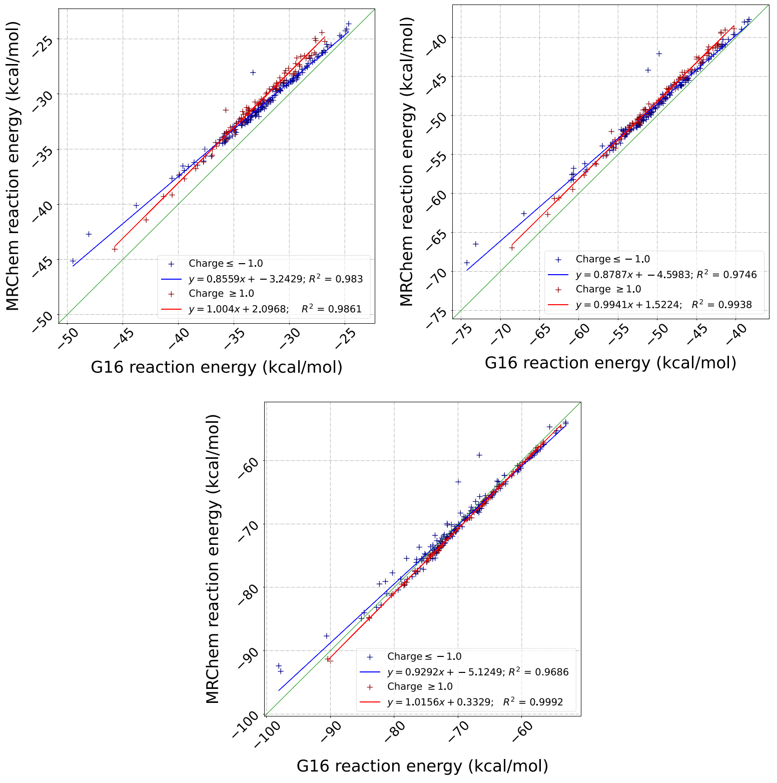


Figure 6. Correlation plots of reaction energies computed with Gaussian16 and MRChem for all positive (red) and negative (blue) ions in the MSDD[59] for ε = 2.0, 4.0, 80.0. All cavities are atom-centered, with Bondi radii.[66,67] Radii are scaled by 1.1. in Gaussian 16. For the MRChem calculations, we used default values: α = 1.1, β = 0.5, σ = 0.2 au linear regression lines are shown in red (blue) for positive (negative) ions, respectively.

method has a diffuse cavity layer, whereas the cavity of IEFPCM is a 2-dimensional boundary. This will lead to contributions and errors that are not equivalent.

Figure 4 shows the results for the cavity parametrization for α = 1.1 and α = 1.2. Results for α = 1.0 and α = 1.3 are not shown, because they largely overestimate (α = 1.0) or underestimate (α = 1.3) solvation energies, but they are available in the data package available online on DataVerse.[68]

We conclude that a cavity parametrization with α = 1.1 and β = 0.5 provides a good correlation with sharp-boundary IEFPCM for all reasonable values of the permittivity and default value of cavity width. This choice of α and β with σ = 0.2 au is the current default in MRChem.

4.2. Model Benchmarking against the Minnesota Solvent Descriptor Database.

The geometries from the MSDD were used to compile a comprehensive benchmark of our model against a sharp-boundary cavity implementation. MSDD holds solvation-related quantities, for a wide variety of solvents and solutes.[59] From the conclusions in the previous section, all MRChem results reported in this section employ the cavity parameters α = 1.1, β = 0.5, and σ = 0.2 au.

Figures 5 and 6 summarize our results, for neutral and charged species, respectively. As for the results in Section 4.1, the figures visualize the correlation between the reaction energies computed with Gaussian16 (x-axis) and MRChem (y-axis).

For low permittivity (ε = 2.0), Figure 5.1 shows that for neutral species our data is quite close to the main diagonal for small energies, but has a slight systematic deviation for more negative reaction energies (bottom left corner). For ions, Figure 6.1 shows a systematic overestimation with respect to Gaussian16, and a clear distinction between cations and anions. For ε = 4.0 (Figure 5.2 and 6.2), we see a similar trend, although most data points appear to be closer to the diagonal. For high permittivity ε = 80, Figure 5.3 for neutral species and Figure 6.3 for ionic ones show that the values are now mostly below the diagonal, that is, solvation energies are underestimated compared to Gaussian16. In Figure 5, we can see a set of outlying point with respect to the rest of the data. These points have been identified as species containing bromine[b] or iodine,[c] with only one outlier containing chlorine instead.[d59] There may be multiple, concomitant reasons for these discrepancies: (a) Bromine and iodine are the only atoms from the fourth and fifth period of the periodic table present in the set; (b) the radii used in the definition of the cavities for these elements might not be appropriate; (c) the different treatment of volume polarization in the two implementations (full account in our model and implicit first-order correction in the IEFPCM model[8,9]) might affect the description of these molecules, where a more delocalized electronic density is expected. It would be interesting to disentangle the effects of surface and volume polarization, but it is not straightforward to do so and it goes beyond the scope of the present work.

The fact that molecules are quite close to the line, especially as the reaction energy becomes small (top right corner), is not surprising. We chose the cavity parameters from a limited set of small molecules. On the other hand, the observed deviations for larger solvation energy are to a large degree systematic, which shows that they could be accounted for, with a more refined parametrization.

Cations tend to have less diffuse density than anions. Therefore, the size of the cavity with respect to the spatial extent of the electronic density is larger for cations than for anions. According to the simple Born model, solvation energy of ions is inversely proportional to the radius of the cavity, which explains the better correlation observed for cations: when the charge distribution is better confined inside the cavity, the difference between a sharp interface formally not accounting for volume polarization and a diffuse one including it, becomes smaller.

4.3. Performance. The current code is a prototype, and we have therefore not yet dedicated attention to improving its performance in terms of computational time and memory footprint. A few general considerations can however be made. The solution of the GPE is technically similar to that of the Helmholtz equation, which we employ to solve the SCF equations.[40,69] It should therefore be possible to achieve linear scaling with respect to the system size once the code is fully optimized.[70] This is a feature of MRA,[37] which is designed to decouple the long- and short-range interactions automatically thanks to the adaptive refinement scheme coupled with the use of the nonstandard form of operators.[38] In this sense, the algorithm should be competitive with implementations of sharp-cavity models that employ the fast multipole method (FMM) to accelerate the matrix-free solution of the PCM equations.[71]

A qualitative comparison with the domain decomposition (DD) family of algorithms[9,72,73] is also in order. DD approaches to implicit solvation are, by construction, linear scaling. Furthermore, they are easily recast in a matrix-free form that both reduces the memory footprint and lends itself to further performance boosting via the FMM.[74] However, in our understanding of the algorithm, these advantages of the method are not straightforwardly extended to cavities with diffuse boundaries. Furthermore, when dealing with quantum mechanical source densities, the quantum-classical coupling must rely on volume integrations, e.g., using a DFT grid, to correctly represent the escaped charge.[75]

Our algorithm achieves formal simplicity and, in principle, algorithmic efficiency. Real-space methods for the reaction potential can be coupled with GTO methods for the electronic-structure problem,[76] thus making our method of interest *beyond* multiwavelet-based quantum chemistry. Currently the main bottleneck is constituted by the memory footprint of the functions describing the cavity and the solvent reaction potential, since they extend throughout the whole computational domain. Work is currently in progress to deal with such functions in an efficient way.

5. CONCLUSIONS

We have implemented, parametrized, and benchmarked a continuum solvation model based on a position dependent permittivity $\varepsilon(\boldsymbol{r})$.[27] Our algorithm performs microiterations, nested within each SCF cycle, to obtain the solvent reaction potential. We overcome convergence issues using KAIN convergence acceleration and an adaptive convergence threshold. Our implementation is robust and introduces only a modest computational overhead.

With a simple parametrization, we have obtained a good correlation with respect to the IEFPCM implemented in Gaussian16, for an extensive library of geometries and a wide range of permittivities. Some systematic deviations have been observed, suggesting that a more sophisticated cavity parametrization could yield even better agreement. An alternative option, which is often challenging for standard solvation models, is to parametrize the permittivity by making use of an isodensity cavity as support. This choice would forego the radius parametrization altogether, but it might pose other challenges, because the cavity gradient must be computed numerically, and the coupling with the density functional must be taken into account.

The performance and stability might be further improved, by considering a different approach to the SCRF microiterations: a square-root parametrization of the electrostatic potential, as suggested by Fisicaro et al., might prove useful.[30]

The flexibility of the method will allow for several additional developments, such as the inclusion of charged particles outside the cavity, as well as other contributions to the solvation energy, such as cavitation, dispersion, and repulsion.

APPENDIX A: ANALYTICAL DERIVATIVES OF THE PERMITTIVITY AND CAVITY FUNCTIONS

A.1: The gradient

The gradient of the permittivity function can be determined analytically. Differentiating eq 22:

$$\begin{aligned}\nabla\varepsilon(\boldsymbol{r}) &= -\varepsilon_{\text{in}}\exp\left[(1 - C(\boldsymbol{r}))\log\left(\frac{\varepsilon_{\text{out}}}{\varepsilon_{\text{in}}}\right)\right]\log\left(\frac{\varepsilon_{\text{out}}}{\varepsilon_{\text{in}}}\right)\nabla C(\boldsymbol{r}) \\ &= \log\left(\frac{\varepsilon_{\text{out}}}{\varepsilon_{\text{in}}}\right)\varepsilon(\boldsymbol{r})\nabla C(\boldsymbol{r})\end{aligned} \tag{28}$$

which only requires to compute the analytical gradient of the interlocking sphere cavity function $C(\boldsymbol{r})$.

The analytical gradient of the interlocking sphere cavity is as defined by Fosso-Tande and Harrison:[27]

$$\nabla C(\boldsymbol{r}) = \left[\prod_{\alpha=1}^{N_{\text{sph}}}(1 - C_\alpha(\boldsymbol{r}))\right]\sum_{\alpha=1}^{N_{\text{sph}}}\frac{\nabla C_\alpha(\boldsymbol{r})}{1 - C_\alpha(\boldsymbol{r})} \tag{29}$$

The gradient of a single sphere cavity function C_α is

$$\nabla C_\alpha(\boldsymbol{r}) = -\frac{1}{\sigma\sqrt{\pi}}\exp\left(\frac{{s_\alpha}^2(\boldsymbol{r})}{\sigma^2}\right)\nabla s_\alpha(\boldsymbol{r}) \tag{30}$$

and finally the gradient of the signed normal distance is

$$\nabla s_\alpha(\boldsymbol{r}) = \begin{pmatrix} \dfrac{x - x_\alpha}{|\boldsymbol{r} - \boldsymbol{r}_\alpha|} \\ \dfrac{y - y_\alpha}{|\boldsymbol{r} - \boldsymbol{r}_\alpha|} \\ \dfrac{z - z_\alpha}{|\boldsymbol{r} - \boldsymbol{r}_\alpha|} \end{pmatrix} \tag{31}$$

In the implementation, we use a cutoff of 10^{-12} for the denominator, in order to avoid numerical discontinuities.

■ ASSOCIATED CONTENT

Data Availability Statement

Input and output files for the Gaussian16 and MRChem calculations reported in this work are available on the Norwegian instance of the Dataverse data repository: 10.18710/TFSWLC. The data package also includes the Jupyter notebooks used to produce the graphs in this paper.

■ AUTHOR INFORMATION

Corresponding Authors

Roberto Di Remigio Eikås − *Algorithmiq Ltd, FI-00160 Helsinki, Finland;* orcid.org/0000-0002-5452-9239; Email: roberto@algorithmiq.fi

Luca Frediani − *Hylleraas Centre for Quantum Molecular Sciences, Department of Chemistry, UiT The Arctic University of Norway, N-9037 Tromsø, Norway;* orcid.org/0000-0003-0807-682X; Email: luca.frediani@uit.no

Authors

Gabriel A. Gerez S − *Hylleraas Centre for Quantum Molecular Sciences, Department of Chemistry, UiT The Arctic University of Norway, N-9037 Tromsø, Norway;* orcid.org/0000-0002-9866-6630

Stig Rune Jensen − *Hylleraas Centre for Quantum Molecular Sciences, Department of Chemistry, UiT The Arctic University of Norway, N-9037 Tromsø, Norway;* orcid.org/0000-0002-2175-5723

Magnar Bjørgve − *Hylleraas Centre for Quantum Molecular Sciences, Department of Chemistry, UiT The Arctic University of Norway, N-9037 Tromsø, Norway*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.2c01098

Author Contributions

We use the CRediT taxonomy of contributor roles.[77,78] The "Investigation" role also includes the "Methodology", "Software", and "Validation" roles. The "Analysis" role also includes the "Formal analysis" and "Visualization" roles. The "Funding acquisition" role also includes the "Resources" role. G.A.G.S. major: Investigation, Data curation, Analysis, Writing − original draft, Writing − revisions; support: Project administration. R.D.R.E. major: Conceptualization, Writing − original draft, Writing − revisions; support: Investigation, Analysis, Supervision. S.R.J. major: −; support: Investigation, Analysis, Supervision, Writing − original draft, Writing − revisions. M.B. major: −; support: Investigation, Supervision, Writing − revisions. L.F. major: Conceptualization, Supervision, Writing − revisions, Funding acquisition, Project administration; support: Investigation, Analysis, Writing − original draft. A graphical representation of this information can be found in the arXiv preprint of this paper: https://arxiv.org/abs/2211.02461.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

We acknowledge support from the Research Council of Norway through its Centres of Excellence scheme, project number 262695, and through the FRIPRO grant ReMRChem (324590), from the Tromsø Research Foundation (SURFINT, A32543) and from Notur—The Norwegian Metacenter for Computational Science through grant of computer time, no. nn4654k. R.D.R.E. acknowledges support from the European High-Performance Computing Joint Undertaking under Grant Agreement No. 951732 and partial support by the Research Council of Norway through its Mobility Grant scheme, project number 261873. We thank Simone Brugiapaglia (Concordia University) for helpful discussions on a point raised by one of the reviewers.

■ ADDITIONAL NOTES

[a]We later learned that the, undocumented, keyword `PrintResultsTable` achieves the same purpose. We used this for one molecule in the benchmark set, the singly charged negative peroxide ion O_2^- (identifier: `i091`), where the external iteration procedure failed to terminate.

[b]Molecules and corresponding filenames in the database: A. 5-bromouracil, $H_3C_4N_2O_2Br$ (`n203`); B. 5-bromo-3-s-butyl-6-methyl-uracil, $H_{13}C_9N_2O_2Br$ (`test1013`); C. 2-bromoanisole, H_7C_7OBr (`test5008`); D. Bromobenzene, H_5C_6Br (`0186bro`); E. 4-bromopyridine, H_4C_5NBr (`0573bro`); F. 1-bromo-2-chloroethane, H_4C_2ClBr (`0202bro`); K. 3-bromoanisole, H_7C_7OBr (`test5009`).

[c]Molecules and corresponding filenames in the database: G. 5-iodouracil, $H_3C_4N_2O_2I$ (`test2018`); H. Iodomethane, H_3CI (`test4003`); I. Iodobenzene, H_5C_6I (`test4001`).

[d]Molecule and corresponding filename in the database: J. 1,4-dichlorobenzene, $H_4C_6Cl_2$ (`0176pdi`).

■ REFERENCES

(1) Miertuš, S.; Scrocco, E.; Tomasi, J. Electrostatic interaction of a solute with a continuum. A direct utilizaion of AB initio molecular potentials for the prevision of solvent effects. *Chem. Phys.* **1981**, *55*, 117−129.

(2) Tomasi, J.; Mennucci, B.; Cammi, R. Quantum mechanical continuum solvation models. *Chem. Rev.* **2005**, *105*, 2999−3093.

(3) Tomasi, J.; Persico, M. Molecular Interactions in Solution: An Overview of Methods Based on Continuous Distributions of the Solvent. *Chem. Rev.* **1994**, *94*, 2027−2094.

(4) Cramer, C.; Truhlar, D. Implicit solvation models: Equilibria, structure, spectra, and dynamics. *Chem. Rev.* **1999**, *99*, 2161−2200.

(5) Jackson, J. D. *Classical Electrodynamics*; Wiley, 1998.

(6) Klamt, A.; Jonas, V. Treatment of the outlying charge in continuum solvation models. *J. Chem. Phys.* **1996**, *105*, 9972−9981.

(7) Chipman, D. M. Charge penetration in dielectric models of solvation. *J. Chem. Phys.* **1997**, *106*, 10194−10206.

(8) Chipman, D. M. Reaction field treatment of charge penetration. *J. Chem. Phys.* **2000**, *112*, 5558−5565.

(9) Cancès, E.; Mennucci, B. Comment on "Reaction field treatment of charge penetration" [J. Chem. Phys. 112, 5558 (2000)]. *J. Chem. Phys.* **2001**, *114*, 4744−4745.

(10) Amovilli, C.; Mennucci, B. Self-Consistent-Field Calculation of Pauli Repulsion and Dispersion Contributions to the Solvation Free

Energy in the Polarizable Continuum Model. *J. Phys. Chem. B* **1997**, *101*, 1051−1057.
(11) Silla, E.; Villar, F.; Nilsson, O.; Pascual-Ahuir, J. L.; Tapia, O. Molecular volumes and surfaces of biomacromolecules via GEPOL: A fast and efficient algorithm. *J. Mol. Graphics* **1990**, *8*, 168−172.
(12) Silla, E.; Tuñón, I.; Pascual-Ahuir, J. L. GEPOL: An improved description of molecular surfaces II. Computing the molecular area and volume. *Journal Of Computational Chemistry* **1991**, *12*, 1077−1088.
(13) Pascual-Ahuir, J. L.; Silla, E. GEPOL: An improved description of molecular surfaces. I. Building the spherical surface set. *Journal Of Computational Chemistry* **1990**, *11*, 1047−1060.
(14) Pomelli, C. S.; Tomasi, J. DefPol: New procedure to build molecular surfaces and its use in continuum solvation methods. *Journal Of Computational Chemistry* **1998**, *19*, 1758−1776.
(15) Pomelli, C. S.; Tomasi, J.; Cossi, M.; Barone, V. Effective generation of molecular cavities in polarizable continuum model by DefPol procedure. *Journal Of Computational Chemistry* **1999**, *20*, 1693−1701.
(16) Connolly, M. L. Analytical molecular surface calculation. *J. Appl. Crystallogr.* **1983**, *16*, 548−558.
(17) Connolly, M. L. The molecular surface package. *J. Mol. Graphics* **1993**, *11*, 139−141.
(18) Foresman, J. B.; Keith, T. A.; Wiberg, K. B.; Snoonian, J.; Frisch, M. J. Solvent Effects. 5. Influence of Cavity Shape, Truncation of Electrostatics, and Electron Correlation on ab Initio Reaction Field Calculations. *Journal Of Physical Chemistry* **1996**, *100*, 16098−16104.
(19) Quan, C.; Stamm, B. Mathematical analysis and calculation of molecular surfaces. *J. Comput. Phys.* **2016**, *322*, 760−782.
(20) Quan, C.; Stamm, B. Meshing molecular surfaces based on analytical implicit representation. *J. Mol. Graph. Model.* **2017**, *71*, 200−210.
(21) Losilla, S. A.; Sundholm, D.; Jusélius, J. The direct approach to gravitation and electrostatics method for periodic systems. *J. Chem. Phys.* **2010**, *132*, 024102.
(22) Genovese, L.; Videau, B.; Ospici, M.; Deutsch, T.; Goedecker, S.; Mehaut, J.-F. Daubechies wavelets for high performance electronic structure calculations: The BigDFT project. *Comptes Rendus Mecanique* **2011**, *339*, 149−164.
(23) Andrade, X.; Strubbe, D.; De Giovannini, U.; Larsen, A. H.; Oliveira, M. J. T.; Alberdi-Rodriguez, J.; Varas, A.; Theophilou, I.; Helbig, N.; Verstraete, M. J.; Stella, L.; Nogueira, F.; Aspuru-Guzik, A.; Castro, A.; Marques, M. A. L.; Rubio, A. Real-space grids and the Octopus code as tools for the development of new simulation approaches for electronic systems. *Phys. Chem. Chem. Phys.* **2015**, *17*, 31371−31396.
(24) Blum, V.; Gehrke, R.; Hanke, F.; Havu, P.; Havu, V.; Ren, X.; Reuter, K.; Scheffler, M. Ab initio molecular simulations with numeric atom-centered orbitals. *Comput. Phys. Commun.* **2009**, *180*, 2175−2196.
(25) Harrison, R. J.; Beylkin, G.; Bischoff, F. A.; Calvin, J. A.; Fann, G. I.; Fosso-Tande, J.; Galindo, D.; Hammond, J. R.; Hartman-Baker, R.; Hill, J. C.; Jia, J.; Kottmann, J. S.; Yvonne Ou, M.-J.; Pei, J.; Ratcliff, L. E.; Reuter, M. G.; Richie-Halford, A. C.; Romero, N. A.; Sekino, H.; Shelton, W. A.; Sundahl, B. E.; Thornton, W. S.; Valeev, E. F.; Vázquez-Mayagoitia, Á.; Vence, N.; Yanai, T.; Yokoi, Y. MADNESS: A Multiresolution, Adaptive Numerical Environment for Scientific Simulation. *SIAM journal on scientific computing* **2016**, *38*, S123−S142.
(26) Bast, R.; Bjørgve, M.; Brakestad, A.; Di Remigio, R.; Frediani, L.; Gerez, G.; Jensen, S. R.; Wind, P. *MRChem: MultiResolution Chemistry*. https://github.com/MRChemSoft/mrchem/tree/v1.1.0, DOI: 10.5281/zenodo.7113393.
(27) Fosso-Tande, J.; Harrison, R. J. Implicit Solvation Models in a Multiresolution Multiwavelet Basis. *Chem. Phys. Lett.* **2013**, *561-562*, 179−184.
(28) Fosso-Tande, J.; Harrison, R. J. Confinement effects of solvation on a molecule physisorbed on a polarizable continuum particle. *Computational and Theoretical Chemistry* **2013**, *1017*, 22−30.
(29) Fisicaro, G.; Genovese, L.; Andreussi, O.; Mandal, S.; Nair, N. N.; Marzari, N.; Goedecker, S. *Soft-Sphere Continuum Solvation in Electronic-Structure Calculations*. 2017.
(30) Fisicaro, G.; Genovese, L.; Andreussi, O.; Marzari, N.; Goedecker, S. A generalized Poisson and Poisson-Boltzmann solver for electrostatic environments. *J. Chem. Phys.* **2016**, *144*, 014103.
(31) Andreussi, O.; Fisicaro, G. Continuum embeddings in condensed-matter simulations. *International Journal Of Quantum Chemistry* **2019**, *119*, No. e25725.
(32) Womack, J. C.; Anton, L.; Dziedzic, J.; Hasnip, P. J.; Probert, M. I. J.; Skylaris, C.-K. DL_MG: A Parallel Multigrid Poisson and Poisson−Boltzmann Solver for Electronic Structure Calculations in Vacuum and Solution. *J. Chem. Theory Comput.* **2018**, *14*, 1412.
(33) Frediani, L.; Pomelli, C. S.; Tomasi, J. n-alkyl alcohols at the water/vapour and water/benzene interfaces: a study on phase transfer energies. *Phys. Chem. Chem. Phys.* **2000**, *2*, 4876−4883.
(34) Corni, S.; Frediani, L. In *Continuum Solvation Models in Chemical Physics: From Theory to Applications*; Mennucci, B., Cammi, R., Eds.; Wiley, 2008; pp 300−312.
(35) Di Remigio, R.; Mozgawa, K.; Cao, H.; Weijo, V.; Frediani, L. A polarizable continuum model for molecules at spherical diffuse interfaces. *J. Chem. Phys.* **2016**, *144*, 124103.
(36) Alpert, B. K. A class of bases in L 2 for the sparse representations of integral operators. *Siam Journal on Mathematical Analysis* **1993**, *24*, 246−262.
(37) Alpert, B.; Beylkin, G.; Gines, D.; Vozovoi, L. Adaptive Solution of Partial Differential Equations in Multiwavelet Bases. *J. Comput. Phys.* **2002**, *182*, 149−190.
(38) Beylkin, G.; Cheruvu, V.; Perez, F. Fast adaptive algorithms in the non-standard form for multidimensional problems. *Appl. Comput. Harmon A* **2008**, *24*, 354−377.
(39) Frediani, L.; Fossgaard, E.; Flå, T.; Ruud, K. Fully adaptive algorithms for multivariate integral equations using the non-standard form and multiwavelets with applications to the Poisson and bound-state Helmholtz kernels in three dimensions. *Mol. Phys.* **2013**, *111*, 1143−1160.
(40) Harrison, R. J.; Fann, G. I.; Yanai, T.; Gan, Z.; Beylkin, G. Multiresolution quantum chemistry: Basic theory and initial applications. *J. Chem. Phys.* **2004**, *121*, 11587.
(41) Harrison, R. J. Krylov subspace accelerated inexact Newton method for linear and nonlinear equations. *Journal Of Computational Chemistry* **2004**, *25*, 328−334.
(42) Sauter, S. A.; Schwab, C. Boundary Element Methods; *Springer Series in Computational Mathematics*; Springer: Berlin, Heidelberg, 2011.
(43) Jensen, F. *Introduction to Computational Chemistry*; John Wiley & Sons, 2013.
(44) Jensen, F. Atomic orbital basis sets. *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2013**, *3*, 273−295.
(45) Keinert, F. *Wavelets and Multiwavelets*; CRC Press, 2003.
(46) Bjørgve, M.; Battistella, E.; Jensen, S. R.; Frediani, L. Manuscript in preparation.
(47) Jensen, S. R.; Saha, S.; Flores-Livas, J. A.; Huhn, W.; Blum, V.; Goedecker, S.; Frediani, L. The Elephant in the Room of Density Functional Theory Calculations. *Journal Of Physical Chemistry Letters* **2017**, *8*, 1449−1457.
(48) Jensen, S. R.; Flå, T.; Jonsson, D.; Monstad, R. S.; Ruud, K.; Frediani, L. Magnetic properties with multiwavelets and DFT: the complete basis set limit achieved. *Phys. Chem. Chem. Phys.* **2016**, *18*, 21145−21161.
(49) Brakestad, A.; Jensen, S. R.; Wind, P.; D'Alessandro, M.; Genovese, L.; Hopmann, K. H.; Frediani, L. Static Polarizabilities at the Basis Set Limit: A Benchmark of 124 Species. *Journal Of Chemical Theory And Computation* **2020**, *16*, 4874−4882.
(50) Norman, P.; Ruud, K.; Saue, T. *Principles and Practices of Molecular Properties: Theory, Modeling, and Simulations*; John Wiley & Sons, 2018.

(51) G. Parr, R.; Yang, W. *Density-functional theory of atoms and molecules*; Oxford University Press Clarendon Press: New York, Oxford England, 1994.
(52) Lin, L.; Lu, J. *A Mathematical Introduction to Electronic Structure Theory*; SIAM Spotlights; SIAM, 2019.
(53) McWeeny, R. Some Recent Advances in Density Matrix Theory. *Rev. Mod. Phys.* **1960**, *32*, 335−369.
(54) Anderson, J.; Harrison, R. J.; Sekino, H.; Sundahl, B.; Beylkin, G.; Fann, G. I.; Jensen, S. R.; Sagert, I. On derivatives of smooth functions represented in multiwavelet bases. *Journal of Computational Physics: X* **2019**, *4*, 100033.
(55) Pulay, P. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.* **1980**, *73*, 393−398.
(56) Anderson, D. G. Iterative Procedures for Non-linear Integral Equations. *J. ACM* **1965**, *12*, 547−560.
(57) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; Li, X.; Caricato, M.; Marenich, A. V.; Bloino, J.; Janesko, B. G.; Gomperts, R.; Mennucci, B.; Hratchian, H. P.; Ortiz, J. V.; Izmaylov, A. F.; Sonnenberg, J. L.; Williams-Young, D.; Ding, F.; Lipparini, F.; Egidi, F.; Goings, J.; Peng, B.; Petrone, A.; Henderson, T.; Ranasinghe, D.; Zakrzewski, V. G.; Gao, J.; Rega, N.; Zheng, G.; Liang, W.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Throssell, K.; Montgomery, J. A., Jr.; Peralta, J. E.; Ogliaro, F.; Bearpark, M. J.; Heyd, J. J.; Brothers, E. N.; Kudin, K. N.; Staroverov, V. N.; Keith, T. A.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A. P.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Millam, J. M.; Klene, M.; Adamo, C.; Cammi, R.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Farkas, O.; Foresman, J. B.; Fox, D. J. *Gaussian16, Revision C.01*; Gaussian, 2016.
(58) Scalmani, G.; Frisch, M. J. Continuous surface charge polarizable continuum models of solvation. I. General formalism. *J. Chem. Phys.* **2010**, *132*, 114110.
(59) Marenich, A. V.; Kelly, C. P.; Thompson, J. D.; Hawkins, G. D.; Chambers, C. C.; Giesen, D. G.; Winget, P.; Cramer, C. J.; Truhlar, D. G. *Minnesota Solvation Database (MNSOL)*, version 2012; University of Minnesota: Minneapolis, 2020.
(60) Adamo, C.; Barone, V. Toward reliable density functional methods without adjustable parameters: The PBE0 model. *J. Chem. Phys.* **1999**, *110*, 6158.
(61) Weigend, F.; Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy. *Phys. Chem. Chem. Phys.* **2005**, *7*, 3297.
(62) Weigend, F. Accurate Coulomb-fitting basis sets for H to Rn. *Phys. Chem. Chem. Phys.* **2006**, *8*, 1057.
(63) Peterson, K. A.; Figgen, D.; Goll, E.; Stoll, H.; Dolg, M. Systematically convergent basis sets with relativistic pseudopotentials. II. Small-core pseudopotentials and correlation consistent basis sets for the post-d group 16−18 elements. *J. Chem. Phys.* **2003**, *119*, 11113−11123.
(64) Improta, R.; Barone, V.; Scalmani, G.; Frisch, M. J. A state-specific polarizable continuum model time dependent density functional theory method for excited state calculations in solution. *J. Chem. Phys.* **2006**, *125*, 054103.
(65) Improta, R.; Scalmani, G.; Frisch, M. J.; Barone, V. Toward effective and reliable fluorescence energies in solution by a new state specific polarizable continuum model time dependent density functional theory approach. *J. Chem. Phys.* **2007**, *127*, 074504.
(66) Bondi, A. Van der Waals volumes and Radii, The J. of Ph. *Journal Of Physical Chemistry* **1964**, *68*, 441−451.
(67) Mantina, M.; Chamberlin, A. C.; Valero, R.; Cramer, C. J.; Truhlar, D. G. Consistent van der Waals Radii for the Whole Main Group. *J. Phys. Chem. A* **2009**, *113*, 5806−5812.
(68) Gerez Sazo, G. A.; Di Remigio Eikås, R.; Frediani, L. *Supporting Data for: Cavity-free continuum solvation: implementation and parametrization in a multiwavelet framework.* 2022; DOI: 10.18710/TFSWLC.
(69) Jensen, S. R.; Durdek, A.; Bjørgve, M.; Wind, P.; Flå, T.; Frediani, L. Kinetic energy-free Hartree−Fock equations: an integral formulation. *J. Math. Chem.* **2023**, *61*, 343−361.
(70) Wind, P.; Bjørgve, M.; Brakestad, A.; Gerez S, G. A.; Jensen, S. R.; Di Remigio Eikås, R.; Frediani, L. MRChem Multiresolution Analysis Code for Molecular Electronic Structure Calculations: Performance and Scaling Properties. *J. Chem. Theory Comput.* **2023**, *19*, 137−146.
(71) Scalmani, G.; Barone, V.; Kudin, K. N.; Pomelli, C. S.; Scuseria, G. E.; Frisch, M. J. Achieving linear-scaling computational cost for the polarizable continuum model of solvation. *Theor. Chem. Acc.* **2004**, *111*, 90−100.
(72) Stamm, B.; Cancès, E.; Lipparini, F.; Maday, Y. A new discretization for the polarizable continuum model within the domain decomposition paradigm. *J. Chem. Phys.* **2016**, *144*, 054101.
(73) Stamm, B.; Lagardère, L.; Scalmani, G.; Gatto, P.; Cancès, E.; Piquemal, J.-P.; Maday, Y.; Mennucci, B.; Lipparini, F. How to make continuum solvation incredibly fast in a few simple steps: A practical guide to the domain decomposition paradigm for the conductor-like screening model. *Int. J. Quantum Chem.* **2019**, *119*, No. e25669.
(74) Mikhalev, A.; Nottoli, M.; Stamm, B. Linearly scaling computation of ddPCM solvation energy and forces using the fast multipole method. *J. Chem. Phys.* **2022**, *157*, 114103.
(75) Nottoli, M.; Stamm, B.; Scalmani, G.; Lipparini, F. Quantum Calculations in Solution of Energies, Structures, and Properties with a Domain Decomposition Polarizable Continuum Model. *J. Chem. Theory Comput.* **2019**, *15*, 6061−6073.
(76) Howard, J. C.; Womack, J. C.; Dziedzic, J.; Skylaris, C.-K.; Pritchard, B. P.; Crawford, T. D. Electronically Excited States in Solution via a Smooth Dielectric Model Combined with Equation-of-Motion Coupled Cluster Theory. *J. Chem. Theory Comput.* **2017**, *13*, 5572−5581.
(77) Allen, L.; Scott, J.; Brand, A.; Hlava, M.; Altman, M. Publishing: Credit where credit is due. *Nature* **2014**, *508*, 312−313.
(78) Brand, A.; Allen, L.; Altman, M.; Hlava, M.; Scott, J. *Beyond authorship: attribution, contribution, collaboration, and credit*; Learned Publishing, 2015; Vol. *28*, pp 151−155.

Bibliography

[1] F. Jensen. “Atomic orbital basis sets”. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 3.3 (2013), pp. 273–295.

[2] D. Moncrieff and S. Wilson. “Computational linear dependence in molecular electronic structure calculations using universal basis sets”. In: *International Journal of Quantum Chemistry* 101.4 (2005), pp. 363–371.

[3] S. Losilla, D. Sundholm, and J. Jusélius. “The direct approach to gravitation and electrostatics method for periodic systems”. In: *The Journal of chemical physics* 132.2 (2010).

[4] L. Genovese, B. Videau, M. Ospici, T. Deutsch, S. Goedecker, and J.-F. Méhaut. “Daubechies wavelets for high performance electronic structure calculations: The BigDFT project”. In: *Comptes Rendus Mécanique* 339.2-3 (2011), pp. 149–164.

[5] X. Andrade, D. Strubbe, U. De Giovannini, A. H. Larsen, M. J. Oliveira, J. Alberdi-Rodriguez, A. Varas, I. Theophilou, N. Helbig, M. J. Verstraete, et al. “Real-space grids and the Octopus code as tools for the development of new simulation approaches for electronic systems”. In: *Physical Chemistry Chemical Physics* 17.47 (2015), pp. 31371–31396.

[6] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler. “Ab initio molecular simulations with numeric atom-centered orbitals”. In: *Computer Physics Communications* 180.11 (2009), pp. 2175–2196.

[7] B. K. Alpert. “A class of bases in Lˆ2 for the sparse representation of integral operators”. In: *SIAM journal on Mathematical Analysis* 24.1 (1993), pp. 246–262.

[8] P. Wind, M. Bjørgve, A. Brakestad, G. A. Gerez S, S. R. Jensen, R. D. R. Eikås, and L. Frediani. “MRChem Multiresolution Analysis Code for Molecular Electronic Structure Calculations: Performance and Scaling Properties”. In: *Journal of Chemical Theory and Computation* 19.1 (2022), pp. 137–146.

[9] R. J. Harrison, G. I. Fann, T. Yanai, and G. Beylkin. “Lecture Notes in Computer Science. Computational Science-ICCS 2003”. In: (2003).

[10] R. J. Harrison, G. I. Fann, T. Yanai, Z. Gan, and G. Beylkin. “Multiresolution quantum chemistry: Basic theory and initial applications”. In: *The Journal of Chemical Physics* 121.23 (2004), pp. 11587–11598.

[11] T. Yanai, G. I. Fann, Z. Gan, R. J. Harrison, and G. Beylkin. “Multiresolution quantum chemistry in multiwavelet bases: Hartree–Fock exchange”. In: *The Journal of Chemical Physics* 121.14 (2004), pp. 6680–6688.

[12] *MADNESS Github*. Accessed: 2023-10-14. 2023. URL: `https://github.com/m-a-d-n-e-s-s/madness`.

[13] *MADNESS Documentation*. Accessed: 2023-10-14. 2023. URL: `https://madness.readthedocs.io/en/latest/`.

[14] R. J. Harrison, G. Beylkin, F. A. Bischoff, J. A. Calvin, G. I. Fann, J. Fosso-Tande, D. Galindo, J. R. Hammond, R. Hartman-Baker, J. C. Hill, et al. “MADNESS: A multiresolution, adaptive numerical environment for scientific simulation”. In: *SIAM Journal on Scientific Computing* 38.5 (2016), S123–S142.

[15] S. R. Jensen, S. Saha, J. A. Flores-Livas, W. Huhn, V. Blum, S. Goedecker, and L. Frediani. “The elephant in the room of density functional theory calculations”. In: *The Journal of Physical Chemistry letters* 8.7 (2017), pp. 1449–1457.

[16] A. Brakestad, P. Wind, S. R. Jensen, L. Frediani, and K. H. Hopmann. “Multiwavelets applied to metal–ligand interactions: Energies free from basis set errors”. In: *The Journal of Chemical Physics* 154.21 (2021).

[17] S. R. Jensen, T. Flå, D. Jonsson, R. S. Monstad, K. Ruud, and L. Frediani. “Magnetic properties with multiwavelets and DFT: the complete basis set limit achieved”. In: *Physical Chemistry Chemical Physics* 18.31 (2016), pp. 21145–21161.

[18] A. Brakestad, S. R. Jensen, P. Wind, M. D’Alessandro, L. Genovese, K. H. Hopmann, and L. Frediani. “Static polarizabilities at the basis set limit: A benchmark of 124 species”. In: *Journal of Chemical Theory and Computation* 16.8 (2020), pp. 4874–4882.

[19] A. Brakestad, S. R. Jensen, C. Tantardini, Q. Pitteloud, P. Wind, J. Uzulis, A. Gulans, and K. Helen. “Scalar relativistic effects with Multiwavelets: Implementation and benchmark”. In: *arXiv preprint arXiv:2310.02149* (2023).

[20] R. van der Pas, E. Stotzer, and C. Terboven. *Using OpenMP—The Next Step: Affinity, Accelerators, Tasking, and SIMD*. Scientific and engineering computation series. MIT Press, Oct. 2017. ISBN: 9780262534789.

[21] T. G. Mattson, Yun, and A. E. Koniges. *The OpenMP Common Core: Making OpenMP Simple Again*. The MIT Press, Nov. 2019. ISBN: 9780262538862.

[22] M. Bjorgve, C. Tantardini, S. Jensen, G. Gerez, P. Wind, R. Eikås, E. Dinvay, and L. Frediani. "VAMPyR – A high level Python library for mathematical operations in a Multiwavelets representation". In: *Manuscript in preparation* (2023).

[23] G. A. Gerez Sazo. "Combining the Polarizable Continuum Model with Multiresolution analysis: a cavity-free approach to solvent effects". MA thesis. UiT Norges arktiske universitet, 2019.

[24] G. A. Gerez S, R. Di Remigio Eikås, S. R. Jensen, M. Bjørgve, and L. Frediani. "Cavity-free continuum solvation: implementation and parametrization in a multiwavelet framework". In: *Journal of Chemical Theory and Computation* 19.7 (2023), pp. 1986–1997.

[25] C. Tantardini, R. D. Remigio Eikås, M. Bjørgve, S. Jensen, and L. Frediani. "Future Perspective for Core-Electron Spectroscopy: Breit Hamiltonian in the Multiwavelets Framework". In: *Journal of Chemical Theory and Computation* (2023). Submitted.

[26] *Numerical Methods in Quantum Chemistry 2023*. Accessed: 2023-10-22. 2023. URL: `https://mrchemsoft.no/nmqc-2023/`.

[27] *Hylleraas School 2021*. Accessed: 2023-10-22. 2021. URL: `https://www.mn.uio.no/hylleraas/english/news-and-events/events/events-in-oslo/2021/hylleraas-school-2021.html`.

[28] B. K. Alpert, G. Beylkin, D. Gines, and L. Vozovoi. "Adaptive solution of partial differential equations in multiwavelet bases". In: *J. Comput. Physics* 182.1 (2002), pp. 149–190.

[29] G. Beylkin. "On the fast algorithm for multiplication of functions in the wavelet bases". In: *Wavelet Analysis and Applications* (1993), pp. 53–61.

[30] L. Frediani, E. Fossgaard, T. Flå, and K. Ruud. "Fully adaptive algorithms for multivariate integral equations using the non-standard form and multiwavelets with applications to the Poisson and bound-state Helmholtz kernels in three dimensions". In: *Molecular Physics* 111.9-11 (2013), pp. 1143–1160.

[31] J. Anderson, R. J. Harrison, H. Sekino, B. Sundahl, G. Beylkin, G. I. Fann, S. R. Jensen, and I. Sagert. "On derivatives of smooth functions represented in multiwavelet bases". In: *Journal of Computational Physics: X* 4 (2019), p. 100033.

[32] M. Bjørgve. "Separable representations of the Poisson, Helmholtz and complex Helmholtz kernels". MA thesis. UiT Norges arktiske universitet, 2017.

[33] K. Singer. "The use of Gaussian (exponential quadratic) wave functions in molecular problems-I. General formulae for the evaluation of integrals". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 258.1294 (1960), pp. 412–420.

[34] W. Hackbusch and B. N. Khoromskij. “Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. Part I. Separable approximation of multi-variate functions”. In: *Computing* 76 (2006), pp. 177–202.

[35] D. J. Griffiths. *Introduction to Quantum Mechanics.* 2nd ed. Harlow: Pearson, 2014.

[36] M. Born. “Born-oppenheimer approximation”. In: *Ann. Phys* 84 (1927), pp. 457–484.

[37] P. W. Atkins and R. S. Friedman. *Molecular quantum mechanics.* Oxford university press, 2011.

[38] R. P. Feynman, A. R. Hibbs, and D. F. Styer. *Quantum mechanics and path integrals.* Courier Corporation, 2010.

[39] W. Pauli. “The connection between spin and statistics”. In: *Physical Review* 58.8 (1940), p. 716.

[40] F. Jensen. *Introduction to computational chemistry.* John wiley & sons, 2017.

[41] H. P. and K. W. “Inhomogeneous Electron Gas”. In: *Phys. Rev.* 136 (1964), B864–B871.

[42] E. committee. *European Summerschool in Quantum Chemistry 2019. Book II. Simen Reine and Trond Saue, Eds.* 2019.

[43] K. W. and S. L. J. “Self-Consistent Equations Including Exchange and Correlation Effects”. In: *Phys. Rev.* 140 (1965), A1133–A1138.

[44] R. J. Harrison. “Krylov subspace accelerated inexact Newton method for linear and nonlinear equations”. In: *Journal of computational chemistry* 25.3 (2004), pp. 328–334.

[45] P. Pulay. “Convergence acceleration of iterative sequences. The case of SCF iteration”. In: *Chemical Physics Letters* 73.2 (1980), pp. 393–398.

[46] D. G. Anderson and M. Engineering Sciences Laboratory Cambridge. “Iterative Procedures for Non-linear Integral Equations”. English. In: (1965), p. 56. DOI: `10.1145/321296.321305`. URL: `http://scholar.google.comjavascript:void(0)`.

[47] S. R. Jensen. “Real-space all-electron density functional theory with multiwavelets”. In: (2014).

[48] H. Anton and C. Rorres. *Elementary linear algebra: applications version.* John Wiley & Sons, 2013.

[49] S. R. Jensen, A. Durdek, M. Bjørgve, P. Wind, T. Flå, and L. Frediani. “Kinetic energy-free Hartree–Fock equations: an integral formulation”. In: *Journal of Mathematical Chemistry* 61.2 (2023), pp. 343–361.

[50] P.-O. Löwdin. “On the non-orthogonality problem connected with the use of atomic wave functions in the theory of molecules and crystals”. In: *The Journal of Chemical Physics* 18.3 (1950), pp. 365–375.

[51] S. F. Boys. “Construction of some molecular orbitals to be approximately invariant for changes from one molecule to another”. In: *Reviews of Modern Physics* 32.2 (1960), p. 296.

[52] S. Miertuš, E. Scrocco, and J. Tomasi. “Electrostatic interaction of a solute with a continuum. A direct utilizaion of AB initio molecular potentials for the prevision of solvent effects”. In: *Chemical Physics* 55.1 (1981), pp. 117–129.

[53] J. Tomasi, B. Mennucci, and R. Cammi. “Quantum mechanical continuum solvation models”. In: *Chemical reviews* 105.8 (2005), pp. 2999–3094.

[54] J. Tomasi and M. Persico. “Molecular interactions in solution: an overview of methods based on continuous distributions of the solvent”. In: *Chemical Reviews* 94.7 (1994), pp. 2027–2094.

[55] C. J. Cramer, D. G. Truhlar, et al. “Implicit solvation models: equilibria, structure, spectra, and dynamics”. In: *Chemical Reviews* 99 (1999), pp. 2161–2200.

[56] T. Vreven and K. Morokuma. “Hybrid methods: Oniom (qm: mm) and qm/mm”. In: *Annual reports in computational chemistry* 2 (2006), pp. 35–51.

[57] H. M. Senn and W. Thiel. “QM/MM methods for biomolecular systems”. In: *Angewandte Chemie International Edition* 48.7 (2009), pp. 1198–1229.

[58] B. Mennucci. “Modeling environment effects on spectroscopies through QM/classical models”. In: *Physical Chemistry Chemical Physics* 15.18 (2013), pp. 6583–6594.

[59] J. M. Olsen, K. Aidas, and J. Kongsted. “Excited states in solution through polarizable embedding”. In: *Journal of Chemical Theory and Computation* 6.12 (2010), pp. 3721–3734.

[60] F. Lipparini and V. Barone. “Polarizable force fields and polarizable continuum model: A fluctuating charges/PCM approach. 1. Theory and implementation”. In: *Journal of Chemical Theory and Computation* 7.11 (2011), pp. 3711–3724.

[61] D. M. Chipman. “Charge penetration in dielectric models of solvation”. In: *The Journal of chemical physics* 106.24 (1997), pp. 10194–10206.

[62] E. Cancès and B. Mennucci. “Comment on “Reaction field treatment of charge penetration”[J. Chem. Phys. 112, 5558 (2000)]”. In: *The Journal of Chemical Physics* 114.10 (2001), pp. 4744–4745.

[63] R. Bast, M. Bjørgve, A. Brakestad, R. Di Remigio, L. Frediani, G. Gerez, S. R. Jensen, and P. Wind. *MRChem: MultiResolution Chemistry.* 2022. URL: `https://github.com/MRChemSoft/mrchem/tree/v1.1.0`.

[64] J. Fosso-Tande and R. J. Harrison. “Implicit solvation models in a multiresolution multiwavelet basis”. In: *Chemical Physics Letters* 561 (2013), pp. 179–184.

[65] J. Fosso-Tande and R. J. Harrison. “Confinement effects of solvation on a molecule physisorbed on a polarizable continuum particle”. In: *Computational and Theoretical Chemistry* 1017 (2013), pp. 22–30.

[66] G. Fisicaro, L. Genovese, O. Andreussi, S. Mandal, N. N. Nair, N. Marzari, and S. Goedecker. “Soft-sphere continuum solvation in electronic-structure calculations”. In: *Journal of Chemical Theory and Computation* 13.8 (2017), pp. 3829–3845.

[67] G. Fisicaro, L. Genovese, O. Andreussi, N. Marzari, and S. Goedecker. “A generalized Poisson and Poisson-Boltzmann solver for electrostatic environments”. In: *The Journal of Chemical Physics* 144.1 (2016).

[68] O. Andreussi and G. Fisicaro. “Continuum embeddings in condensed-matter simulations”. In: *International Journal of Quantum Chemistry* 119.1 (2019), e25725.

[69] J. C. Womack, L. Anton, J. Dziedzic, P. J. Hasnip, M. I. Probert, and C.-K. Skylaris. “DL_MG: A parallel multigrid Poisson and Poisson–Boltzmann solver for electronic structure calculations in vacuum and solution”. In: *Journal of chemical theory and computation* 14.3 (2018), pp. 1412–1432.

[70] P. Pyykko. “Relativistic effects in structural chemistry”. In: *Chemical Reviews* 88.3 (1988), pp. 563–594.

[71] J. Anderson, R. J. Harrison, B. Sundahl, W. S. Thornton, and G. Beylkin. “Real-space quasi-relativistic quantum chemistry”. In: *Computational and Theoretical Chemistry* 1175 (2020), p. 112711.

[72] J. Anderson, B. Sundahl, R. Harrison, and G. Beylkin. “Dirac-Fock calculations on molecules in an adaptive multiwavelet basis”. In: *The Journal of Chemical Physics* 151.23 (2019).

[73] R. E. Stanton and S. Havriliak. “Kinetic balance: A partial solution to the problem of variational safety in Dirac calculations”. In: *The Journal of chemical physics* 81.4 (1984), pp. 1910–1918.

[74] F. Parpia and I. Grant. “Software for relativistic atomic theory: The GRASP project at Oxford”. In: *Le Journal de Physique IV* 1.C1 (1991), pp. C1–33.

[75] A. Rosén and D. Ellis. “Relativistic molecular wavefunctions: XeF2”. In: *Chemical Physics Letters* 27.4 (1974), pp. 595–599.

[76] J. Desclaux and P. Pyykko. “Relativistic and non-relativistic Hartree-Fock one-centre expansion calculations for the series CH4 to PbH4 within the spherical approximation”. In: *Chemical Physics Letters* 29.4 (1974), pp. 534–539.

[77] D. Sundholm, P. Pyykkö, and L. Laaksonen. "Two-dimensional, fully numerical solutions of second-order Dirac equations for diatomic molecules. part 3". In: *Physica Scripta* 36.3 (1987), p. 400.

[78] J. M. Kasper, T. F. Stetina, A. J. Jenkins, and X. Li. "Ab initio methods for L-edge x-ray absorption spectroscopy". In: *Chemical Physics Reviews* 1.1 (2020).

[79] M. L. Vidal, P. Pokhilko, A. I. Krylov, and S. Coriani. "Equation-of-motion coupled-cluster theory to model L-edge x-ray absorption and photoelectron spectra". In: *The journal of physical chemistry letters* 11.19 (2020), pp. 8314–8321.

[80] A. Petrov, N. Mosyagin, A. Titov, and I. Tupitsyn. "Accounting for the Breit interaction in relativistic effective core potential calculations of actinides". In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 37.23 (2004), p. 4621.

[81] B. Mussard and S. Sharma. "One-step treatment of spin–orbit coupling and electron correlation in large active spaces". In: *Journal of chemical theory and computation* 14.1 (2018), pp. 154–165.

[82] G. Breit. "An interpretation of Dirac's theory of the electron". In: *Proceedings of the National Academy of Sciences* 14.7 (1928), pp. 553–559.

[83] G. Breit. "Dirac's equation and the spin-spin interactions of two electrons". In: *Physical Review* 39.4 (1932), p. 616.

[84] P. A. M. Dirac. "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.

[85] P. A. M. Dirac. "The quantum theory of the electron. Part II". In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 118.779 (1928), pp. 351–361.

[86] B. L. van der Waerden. *Group Theory and Qauntum Mechanics.* Springer, Berlin, 1974.

[87] T. Saue. "Relativistic Hamiltonians for chemistry: A primer". In: *ChemPhysChem* 12.17 (2011), pp. 3077–3094.

[88] J. Blackledge and B. Babajanov. "On the Dirac scattering problem". In: (2013).

[89] *VAMPyR Github.* Accessed: 2023-10-22. 2023. URL: `https://github.com/mrchemSoft/vampyr`.

[90] *VAMPyR Anaconda.* Accessed: 2023-10-22. 2023. URL: `https://anaconda.org/conda-forge/vampyr`.

[91] *VAMPyR Documentation.* Accessed: 2023-10-14. 2023. URL: `https://docs.mrchemsoft.no/projects/vampyr/en/latest/`.

[92] *Binder notebooks VAMPyR.* Accessed: 2023-10-14. 2023. URL: `https://mybinder.org/v2/gh/MRChemSoft/vampyr/master?urlpath=lab%2Ftree%2Fdocs%2Fnotebooks`.

[93] *C++ Templates.* Accessed: 2023-10-22. 2023. URL: `https://en.cppreference.com/w/cpp/language/templates`.

[94] D. Vandevoorde, N. M. Josuttis, and D. Gregor. *C++ Templates: The Complete Guide.* 2 edition. Addison-Wesley, 2018. ISBN: 9780321714121.

[95] D. Beazley. *Understanding the Python GIL.* PyCon 2010, Atlanta, Georgia. ccessed 2023-10-22. 2010. URL: `https://www.dabeaz.com/python/UnderstandingGIL.pdf`.