Exploring Optimization Algorithms in Quantum Chemistry: An Educational Approach Using PyNOF

Juan Felipe Huan Lew-Yee,[*,†] Xabier Lopez,[‡,†] Elixabete Rezabal,[‡,†] Mario Piris,[‡,†,¶] and Jose M. Mercero[*,‡,†]

†*Donostia International Physics Center (DIPC), 20018 Donostia, Euskadi, Spain*

‡*Kimika Fakultatea, Euskal Herriko Unibertsitatea (UPV/EHU), P.K. 1072, 20080 Donostia, Euskadi, Spain*

¶*IKERBASQUE, Basque Foundation for Science, 48013 Bilbao, Euskadi, Spain*

E-mail: felipe.lew.yee@dipc.org; jm.mercero@ehu.eus

Abstract

This work presents an educational workflow designed to teach modern optimization techniques in the context of quantum chemistry, with a focus on the orbital optimization problem as the central example. The project starts with the implementation of several optimization methods for Hartree-Fock (HF), serving as a familiar reference point for comparing optimization strategies. Later, the experiments are expanded to Natural Orbital Functional (NOF) calculations as a straightforward way to incorporate electron correlation. By analyzing the performance of both classical and modern algorithms, such as Steepest Descent, AdaGrad, RMSProp, and Adam, students gain insight into the effects of factors such as the learning rate on the convergence behavior and algorithmic efficiency. Alongside, the use of both the HF and NOF methods illustrates the impact of electronic correlation. These modern algorithms, widely used in

deep learning, remain scarcely applied in electronic structure problems, making their exploration in this context both innovative and pedagogically valuable. The exercises are based on Jupyter Notebooks supported on the PyNOF software, a Python implementation of the Natural Orbital Functional (NOF) Theory, enabling a hands-on approach in which students can integrate the main components into their own written optimization code. Furthermore, the use of Jupyter Notebooks ensures an interactive and accessible learning environment that integrates coding, visualization, and theoretical discussion.

Introduction

Programming has become an increasingly essential skill in the XXI century scientific education, particularly in fields such as physics and chemistry. Problems once considered intractable are now approachable thanks to the exponential growth in computational power. Consequently, numerical techniques are now present in nearly every scientific domain, enabling cross-disciplinary studies in physics, biology, mathematics, economics, and chemistry. However, (under)graduate education, particularly in chemistry, often lags behind these developments.

In chemistry education, quantum mechanics is frequently introduced through systems with analytic solutions (e.g., the particle in a box, the harmonic oscillator, or the hydrogen atom). Although these models are foundational, they do not reflect the complexity of most chemically relevant systems, which often lack closed-form solutions. To address this gap, computational approaches based on numerical techniques are necessary to obtain approximate solutions to the Schrödinger equation and to explore realistic chemical systems.

Optimization methods are especially important in this context. As highlighted by Kochenderfer and Wheeler,[1] optimization lies at the core of many computational tools used for both analysis and synthesis. In synthesis problems, for instance, the goal is to find a system configuration that meets specific performance objectives. This is achieved

by minimizing or maximizing an objective function, often subject to constraints. In analysis problems, the objective is typically to evaluate or interpret the behavior of a given system by identifying parameter values or configurations that best explain the observed data or the underlying physical principles. Classical methods such as Steepest Descent, Newton-Raphson, Conjugate Gradient, and Simplex algorithms[2–5] have long been the mainstay in numerical optimization and are widely covered in (under)graduate curricula.

However, as problem complexity increases and non-convex optimization landscapes become the norm, classical algorithms often struggle with convergence and efficiency. To overcome these limitations, a new class of optimization methods has emerged. Algorithms such as AdaGrad,[6] RMSprop,[7] and Adam[8] were originally designed to train deep neural networks, but have since found applications in broader scientific computing contexts.[9,10] These methods adapt learning rates and incorporate moment-based updates, offering robustness in high-dimensional settings and noisy gradients.

Quantum chemistry provides a rich platform for exploring and comparing optimization methods. In particular, the HF method, which is central to electronic structure theory,[11–13] can be formulated as an optimization task, more precisely as a constrained minimization. It seeks to determine a set of orthonormal orbitals that minimize the total electronic energy of a molecular system. More advanced methods based on NOF theory[14] extend this framework by incorporating fractional orbital occupancies, thereby accounting for electron correlation effects.

In this work, we use the open-source PyNOF library,[15] to introduce undergraduate and graduate students to electronic structure optimization through a practical and interactive environment. Developed as the Python counterpart of the DoNOF[16,17] code (a NOF program originally written in Fortran), PyNOF is designed to provide clarity and accessibility for educational purposes, while DoNOF remains more efficient for large-scale numerical calculations. PyNOF has been adapted for pedagogical use, allowing students to practice their programming skills by implementing, testing, and comparing various optimization strategies within a flexible and interactive environment.

Students can observe convergence behavior step by step, experimenting with different algorithms and parameters.

The goal of this paper is to propose a computational (Jupyter) notebook based teaching project for computational techniques in the context of quantum chemistry studies that allows students to engage with optimization methods through their direct application to HF and NOF calculations. The pedagogical use of HF as a foundation for the introduction of electronic structure theory has been well established in chemical education, providing a conceptually simple yet quantitatively meaningful framework for exploring quantum mechanical approximations and molecular properties,[18–22] while the incorporation of NOF allows explicit treatment of electron correlation (see the H_2 dissociation section).

The educational use of notebooks ensures an interactive and user-friendly learning experience. This approach follows recent trends[23–25] in chemistry education, where notebooks have been successfully adopted to teach quantum chemical methods and promote active learning.

We present how both classical and modern optimization techniques can be effectively integrated into the chemistry curriculum, providing students with tools and insights that align with current research practices. This module has been developed and implemented within the Master's course "Numerical Analysis and Computational Techniques" part of the Erasmus Mundus Theoretical Chemistry and Computational Modelling (TCCM) program.[26] It complements the theoretical content with interactive coding and data analysis sessions in which students not only explore optimization methods but also actively develop their programming skills.

After a concise yet comprehensive overview of the theoretical background, the module begins by examining how different optimization algorithms respond to variations in the learning rate using two simple molecules, H_2O and CO_2.One of these case studies may be moved to the Supporting Information for brevity?.

Based on this preliminary analysis, students attempt to determine suitable learning rates for each algorithm. Although this provides useful initial guidelines, it also reveals

a fundamental point: optimal α values depend on the specific system and algorithm (as discussed in Section Results), encouraging students to reflect on generalizability and transferability in computational modeling.

With these empirically selected learning rates, the algorithms are then tested on a broader set of molecules to assess their performance in more diverse chemical environments. This step allows students to compare the reliability and efficiency of the different methods across systems. To build on this foundation, the module next introduces a more advanced theoretical framework by transitioning from HF to NOF calculations. This progression not only reinforces previously learned concepts but also exposes students to additional variables, such as fractional occupation numbers, which increase the complexity of the optimization. Students explore the convergence behavior of three different molecules using two representative learning rates for each algorithm, gaining exposure to the numerical challenges associated with more sophisticated electronic structure methods.

After observing that the Adam algorithm generally achieves the best convergence across systems, the module concludes with a focused analysis of its dependence on the learning rate. Students observe that smaller values of α lead to smoother and more stable minimization paths, whereas larger values allow a faster initial descent but may overshoot or fail to reach the lowest energy. These findings motivate a final refinement: a modified version of the Adam optimizer with dynamic α, showcasing how thoughtful adaptation of algorithmic parameters can improve practical performance in quantum chemical applications.

Finally, in order for students to perceive the effects of electron correlation, and as an exercise to practice Python programming skills, the dissociation of H_2 is analyzed and discussed using HF and NOF, ideally with Adam. The (restricted) HF approach provides an inadequate description of the hydrogen molecule at large internuclear separations, leading to the well-known issue of incorrect homolytic bond dissociation due to its inability to capture static correlation. NOF theory, through fractional occupation of natural orbitals, overcomes this limitation.

In the following section, only minimal theoretical elements are introduced, with references provided for more complete treatments.

Theoretical Background

Hartree-Fock Theory

Hartree-Fock (HF) theory is a cornerstone of quantum chemistry and serves as the starting point for many electronic structure methods. It approximates the many-electron wavefunction with a single Slater determinant composed of molecular orbitals, which are optimized to minimize the total electronic energy. The HF equations are solved self-consistently, typically through iterative diagonalization of the Fock matrix.

Modern quantum chemistry packages such as Gaussian,[27] Psi4,[28] or ORCA[29] implement highly optimized routines for HF calculations. These codes employ a variety of numerical techniques: direct inversion in the iterative subspace (DIIS), level-shifting, and density matrix mixing, among others, to accelerate convergence.[30–32]

In this work, our aim is not to compete with the performance or scalability of these production-level codes. Rather, our objective is educational: to examine how different optimization strategies affect convergence and efficiency in solving the HF equations.

Moreover, instead of relying on diagonalization, the HF problem is solved through orbital rotations, which inherently satisfy the orthonormality constraints. In this approach, the molecular orbitals are transformed using a unitary matrix $\mathbf{U} = e^{\mathbf{y}}$, where $\mathbf{y_{pq}}$ is an antisymmetric matrix ($y_{pq} = -y_{qp}$). This parameterization makes the HF energy a function of the $N_{bf}(N_{bf}-1)/2$ independent rotation parameters, where N_{bf} is the number of basis functions.[9,33–35]

By expressing the energy as a function of $\mathbf{y}$, we obtain a smooth and unconstrained minimization problem that is particularly suitable for testing different optimization strategies. This parameterization provides a concrete and instructive framework for introducing optimization strategies in a physically meaningful context.

At this point, we can use and compare the performance of classical optimization methods such as Steepest Descent and Conjugate Gradient, alongside adaptive techniques originally developed for machine learning, including AdaGrad, RMSProp, and Adam.

Natural Orbital Functional Theory

NOF theory extends beyond HF by incorporating electronic correlation through natural orbitals and occupation numbers.[36,37] Recent advances in the code, some of them inspired by modern machine learning algorithms, have significantly enhanced its performance.[9,38–40] For interested readers, several comprehensive reviews are available,[14,41] and the method has been successfully applied to a variety of chemical problems.[42–44]

PyNOF[15] efficiently implements NOF-based methods in Python, allowing rapid prototyping and algorithm development. One of the key performance improvements was the implementation of orbital rotation techniques. In that context, Adam optimizer[8] proved to be particularly effective in accelerating convergence.[9] This observation inspired the idea of using NOF calculations as a platform to test and compare the performance of classical versus modern optimization algorithms.

In the context of this educational module, NOF theory is also introduced to expose students to the concept of electronic correlation and its numerical treatment through variational optimization.

Optimization Algorithms

In most scientific textbooks on computational techniques, multidimensional optimization algorithms are commonly divided into categories based on the approach they use. These include methods based on changes in the value of the function itself,[45] such as the Simplex method;[46] those based on the gradient, such as gradient descent[47] and the conjugate gradient method;[48] and those that incorporate both the gradient and the Hessian, such as the Newton-Raphson method.[3] While the methods based only on

the change of the function value are computationally fast, the convergence is slow and a lot of steps are required, while in the last class of methods, the convergence may require fewer steps, but the computation of the Hessian may require a lot of computational resources; thus, usually gradient-based methods are the preferred ones in terms of performance and efficiency.

The usual way to introduce gradient-based optimization methods in a theoretical course is to begin with the steepest descent, followed by the conjugate gradient, and then move on to more refined techniques such as those with Momentum.[1] In this module, we adopt a similar pedagogical approach: Steepest descent is presented first, providing a natural foundation for the introduction of momentum-based adaptive strategies like AdaGrad, RMSProp, and Adam. The selection of algorithms included in the module is not meant to be exhaustive. Depending on the time available, the students' background, or the instructor's preferences, additional gradient-based methods may be incorporated or substituted.

To support hands-on learning, we provide students with a PyNOF notebook in which the Steepest Descent algorithm is already implemented. After the instructor explains its theoretical foundations and implementation, students are assigned the task of coding one or more of the remaining algorithms. This structure is flexible: All students may work on the same algorithm, or different groups may implement different ones. Importantly, having Steepest Descent as a reference makes the implementation of the other methods straightforward. This study examines the following optimization techniques:

- **Steepest Descent** (SD)

  Steepest Descent is a simple and intuitive optimization method that updates parameters by moving in the direction of the negative gradient of the objective function. At each step, the algorithm computes the gradient and takes a step proportional to it, typically scaled by a fixed learning rate (α). While easy to implement and conceptually straightforward, Steepest Descent often suffers from slow convergence, especially in ill-conditioned problems, as it tends to zigzag

toward the minimum.

- **Conjugate Gradient** (CG)

  Conjugate Gradient improves upon Steepest Descent by combining the current gradient with information from previous steps to generate search directions that are mutually conjugate. This allows the algorithm to avoid the inefficient zigzag path typical of steepest descent and often leads to significantly faster convergence, particularly in problems with quadratic or nearly quadratic objective functions.

- **AdaGrad**

  The Adaptive Gradient algorithm[6] (AdaGrad) dynamically scales the learning rate for each parameter based on the cumulative sum of squared gradients. This mechanism causes the learning rate to decrease more aggressively for parameters that frequently experience large gradients, while maintaining relatively larger steps for infrequently updated parameters. As a result, AdaGrad is especially effective in scenarios involving sparse gradients, where only a subset of parameters receive significant updates at each iteration. This is common in machine learning applications such as natural language processing, where certain features occur far less frequently than others. Regarding α. The learning rate, AdaGrad is less sensitive than Steepest Descent.

- **RMSProp**

  RMSProp[7] (Root Mean Square Propagation) was developed to address a key limitation of AdaGrad: its tendency to accumulate squared gradients indefinitely, which causes the learning rate to shrink too much over time and eventually halt learning. RMSProp improves this by using a moving average of squared gradients instead of a cumulative sum. In this way, it adapts the learning rate like AdaGrad but avoids the problem of vanishing steps. RMSProp is particularly effective for nonstationary or noisy problems, making it suitable for deep learning and optimization in complex energy landscapes, such as those encountered in quantum chemistry orbital optimizations.

- **Adam** Adam (Adaptive Moment Estimation)[8] combines the strengths of both momentum-based methods and adaptive learning rates. Like AdaGrad and RMSProp, it uses an exponentially decaying average of past squared gradients to scale the learning rate. In addition, it tracks the first moment (the mean) of the gradients, similar to classical momentum methods. This dual accumulation allows Adam to adaptively tune the step sizes for each parameter while also maintaining a sense of direction over time. Originally developed for stochastic optimization in machine learning, Adam has demonstrated rapid and stable convergence in a variety of non-convex problems. These features suggest that it may be particularly well suited for complex quantum chemical optimizations, where convergence stability and efficiency are often critical.

Calculation Consideration

It is important to clarify that the goal of this module is not to obtain highly accurate energies, but rather to provide a realistic and computationally tractable context in which to explore the performance of various optimization algorithms applied to quantum chemistry problems. Specifically, we focus on the iterative procedures used to solve the HF and NOF equations.

In order to ensure that all calculations can be completed interactively and within a reasonable time frame, even on standard personal computers, we have adopted a set of simplifications and constraints. First, the test molecules are small, with modest basis sets. Second, the convergence criteria used in PyNOF are intentionally loose: both the energy and gradient thresholds are set to 10^{-4} (in atomic units). The primary convergence criterion is based on the gradient norm, although energy changes are also monitored.

To further limit the computational cost, the maximum number of iterations is capped at 130. If convergence is not achieved within that limit, the optimization is terminated. Furthermore, if the energy change ΔE becomes smaller than 10^{-8} for 10 consecutive steps, the calculation is stopped under the assumption that the algorithm

has stalled. These choices prioritize pedagogical accessibility and allow students to focus on algorithmic behavior rather than high-precision quantum chemical results.

Application of Optimization Methods to HF and NOF Calculations

HF Results

Learning-Rate dependence

To analyze the effect of the learning rate α, we selected two small molecules, H_2O and CO_2, as test cases. These systems are simple enough to allow fast calculations, yet differ in their symmetry and electronic structure, providing a meaningful contrast for assessing algorithm performance. Each HF energy calculation was performed using different optimization algorithms: SD, AdaGrad, RMSProp, and Adam.

As described in the Calculation Considerations section, the number of iterations was limited to 130, and optimization was stopped early if the energy failed to improve for 10 consecutive steps. Reference values for Hartree-Fock energies can be easily found elsewhere[49] For completeness, we reproduce the values of the molecules used in this work in Table 1. These can be used to evaluate the convergence efficiency and stability between the different optimization schemes tested. In particular, Figure 1 presents the results for H_2O; the execution of the test in a second molecule, such as CO_2 (the results of which can be found in Figure 6 of the SI material), leads the students to confirm they conclusions.

Table 1: HF energies calculated with Psi4,[28] using default options.

	E_{HF} (a.u.)
H_2O	-76.0269680
CO_2	-187.6439441
BF_3	-323.2007977
$CHCl_3$	-1416.938147
$Al(OH)_3$	-468.4248989

From an instructional perspective, this comparative analysis provides students with a hands-on opportunity to observe how algorithmic choices and hyperparameter tuning affect numerical performance in quantum chemical calculations. Among the algorithms tested, Adam generally showed the fastest and most stable convergence across different α values, requiring fewer iterations and demonstrating reliable performance across parameter variations.

RMSProp also performed well, although it generally required more steps to reach a similar level of accuracy. AdaGrad demonstrated good initial convergence, but tended to slow down in later iterations due to its accumulating gradient normalization, which effectively reduces the step size over time. In contrast, SD was highly sensitive to the learning rate, often requiring careful tuning: small values led to slow convergence, while large values risked instability or divergence. The results obtained with the CG method, included in the supporting information, clearly show faster and more stable convergence compared to SD, making CG a valuable upgrade for students to explore beyond basic gradient-based optimization.

Although SD may not be the most efficient algorithm, it remains pedagogically valuable for helping students understand the fundamental role of gradients and how convergence behavior is influenced by step size. The results obtained for H_2O and CO_2 were qualitatively consistent, reinforcing the general applicability of these conclusions. This exercise helps students understand how gradient-based optimization methods behave in realistic quantum chemistry problems and how the choice of algorithm and learning rate affects the outcome. In general, it reinforces key concepts in numerical optimization and fosters critical thinking about algorithmic performance in computational chemistry.

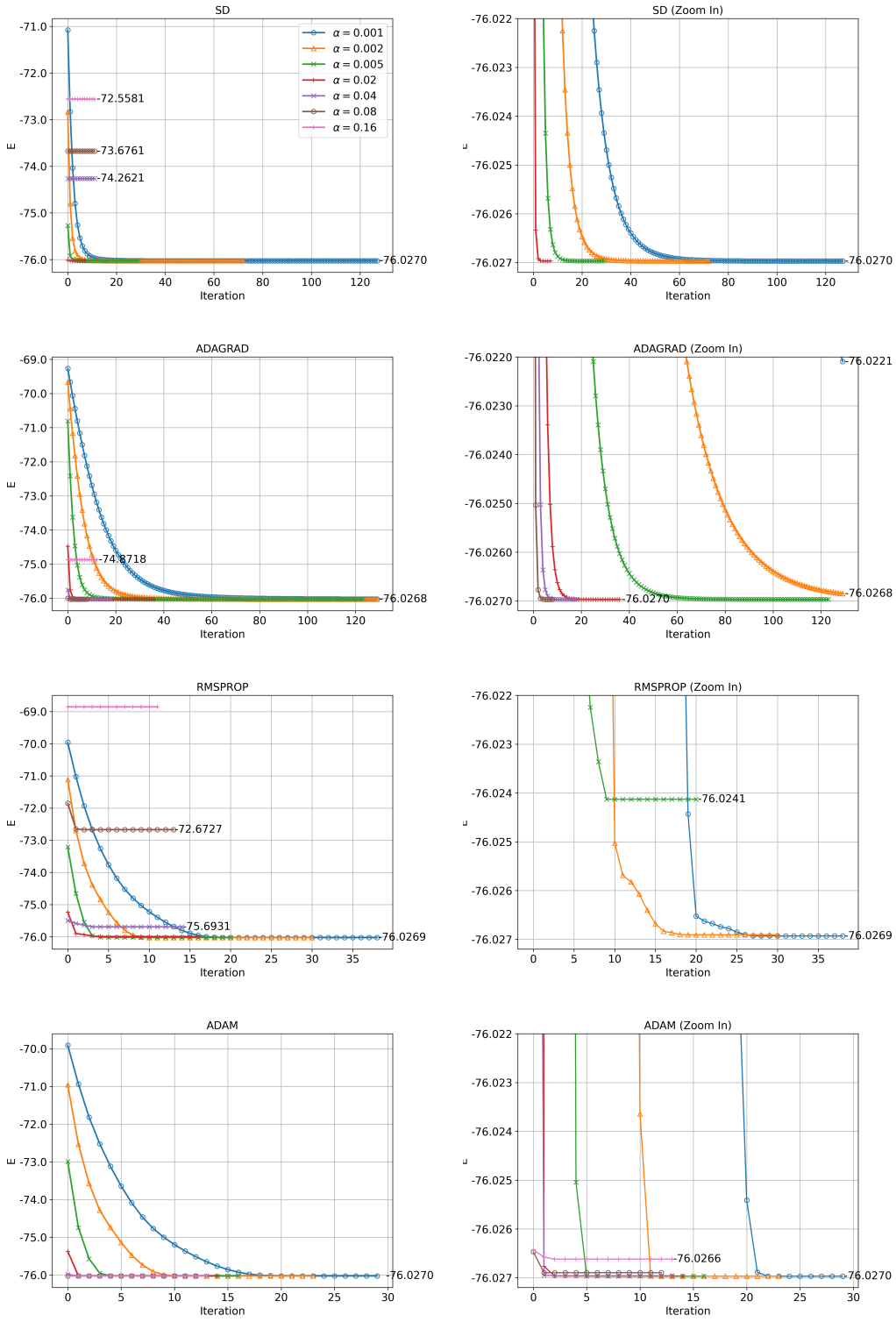

Figure 1: H_2O Hartree-Fock energies convergence (in a.u.) *vs.* the iteration number is plotted for different learning rates (α). Full representations are shown at the left column labeled with the used algorithm. At the right column lowest energy region is zoomed.

Based on the previous analysis of the sensitivity of the learning rate, we selected representative values of α for each algorithm: 0.02 for SD, and Adam; 0.08 for AdaGrad; and 0.002 for RMSProp. These values were chosen because they led to fast and stable convergence in the two test cases discussed earlier, H_2O and CO_2. Using these fixed values, we applied each optimization method to a new set of small molecules (BF_3, $CHCl_3$, and $Al(OH)_3$) to evaluate their HF energies and compare the overall performance of the algorithms. The number of iterations required to achieve energy convergence and the final energy values are summarized in Figure 2.

These results confirm that the primary goal, achieving the lowest possible HF energy, is best accomplished by adaptive methods such as Adam, which consistently reached the lowest energies across systems. While RMSProp showed slightly faster convergence on average, it failed to attain the same level of accuracy, highlighting that speed alone is not a reliable indicator of algorithmic efficiency in energy minimization. In contrast, SD, CG and AdaGrad resulted in significantly higher final energies (except for BF_3 where they performed nicely). Therefore, one must carefully consider not only how quickly an algorithm converges, but also whether it reliably approaches the true minimum energy for the system under study.

Although the learning rate values used in this section were selected based on the convergence behavior observed for H_2O and CO_2, it is clear that several algorithms did not reach convergence for more complex molecules such as BF_3, $CHCl_3$, and $Al(OH)_3$. This finding helps students understand that optimal learning rates are not universally applicable across different systems. While small molecules like H_2O and CO_2 offer a smooth and well-behaved optimization landscape, larger or chemically richer systems may present more irregular gradients or stronger curvature effects, which alter the sensitivity of each algorithm to the learning rate parameter. This reinforces the idea that the tuning of hyperparameters such as α should be approached on a case-by-case basis, and that a learning rate deemed "optimal" in one context may not yield reliable convergence in another. Discussing these limitations with students provides an opportunity to reflect on the importance of adaptability and diagnostic evaluation in

numerical optimization workflows.

For instance, in the case of BF_3, both CG and AdaGrad also reach the correct HF energy, and even require slightly fewer iterations than Adam to do so. However, in more challenging systems like $CHCl_3$ and $Al(OH)_3$, several algorithms fail to converge at all, and only Adam and RMSProp maintain consistent and accurate performance, note though that Adam always achieves a slightly lower energy than RMSProp, which are actually within the convergence criteria, and in a smaller number of steps. These results highlight the importance of evaluating algorithms not just by their speed of convergence, but by whether they reliably reach the correct solution across a range of molecular systems.

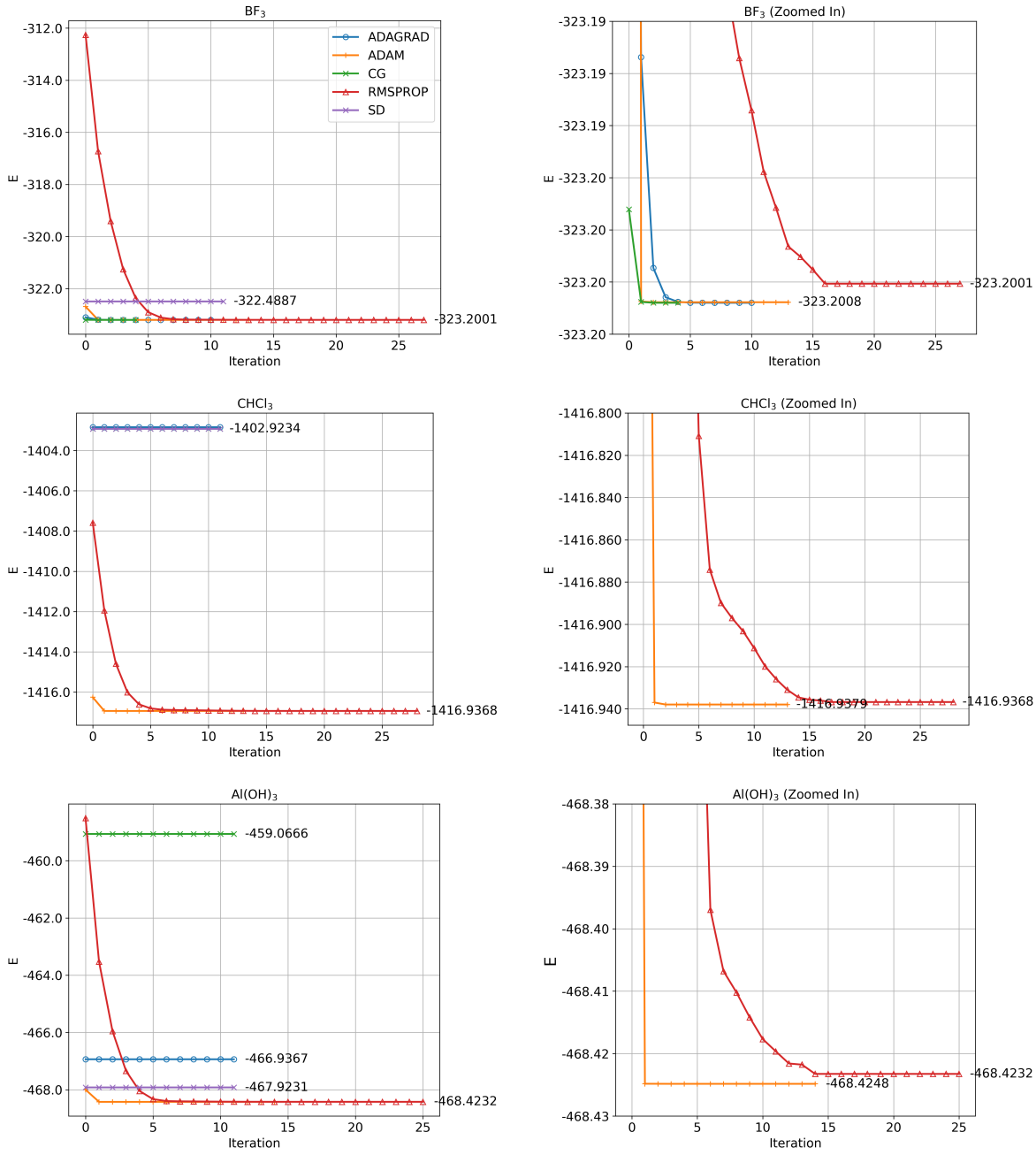

Figure 2: Convergence of HF energies (in a.u.) for different molecules versus iteration number, using various optimization algorithms with selected α values based on previous calculations: 0.02 for SD, CG, and Adam; 0.08 for AdaGrad; and 0.002 for RMSProp.

Having analyzed the performance of the optimization algorithms in the HF framework, we now extend the study to the more demanding context of NOF theory. This shift introduces electron correlation effects, increasing the complexity of the optimiza-

tion landscape and provide a richer context for students to apply and compare the methods.

NOF Results

In NOF theory, the total energy is expressed as a functional of the natural orbitals and their occupation numbers, allowing explicit inclusion of electron correlation effects. As a test set, we have selected three representative molecules: CO, CO_2, and H_2O. For each molecule, calculations are carried out using the five optimization algorithms discussed previously, and for each algorithm, two learning rate values are employed: $\alpha = 0.02$ and $\alpha = 0.002$. The convergence behavior is summarized in Figure 3.

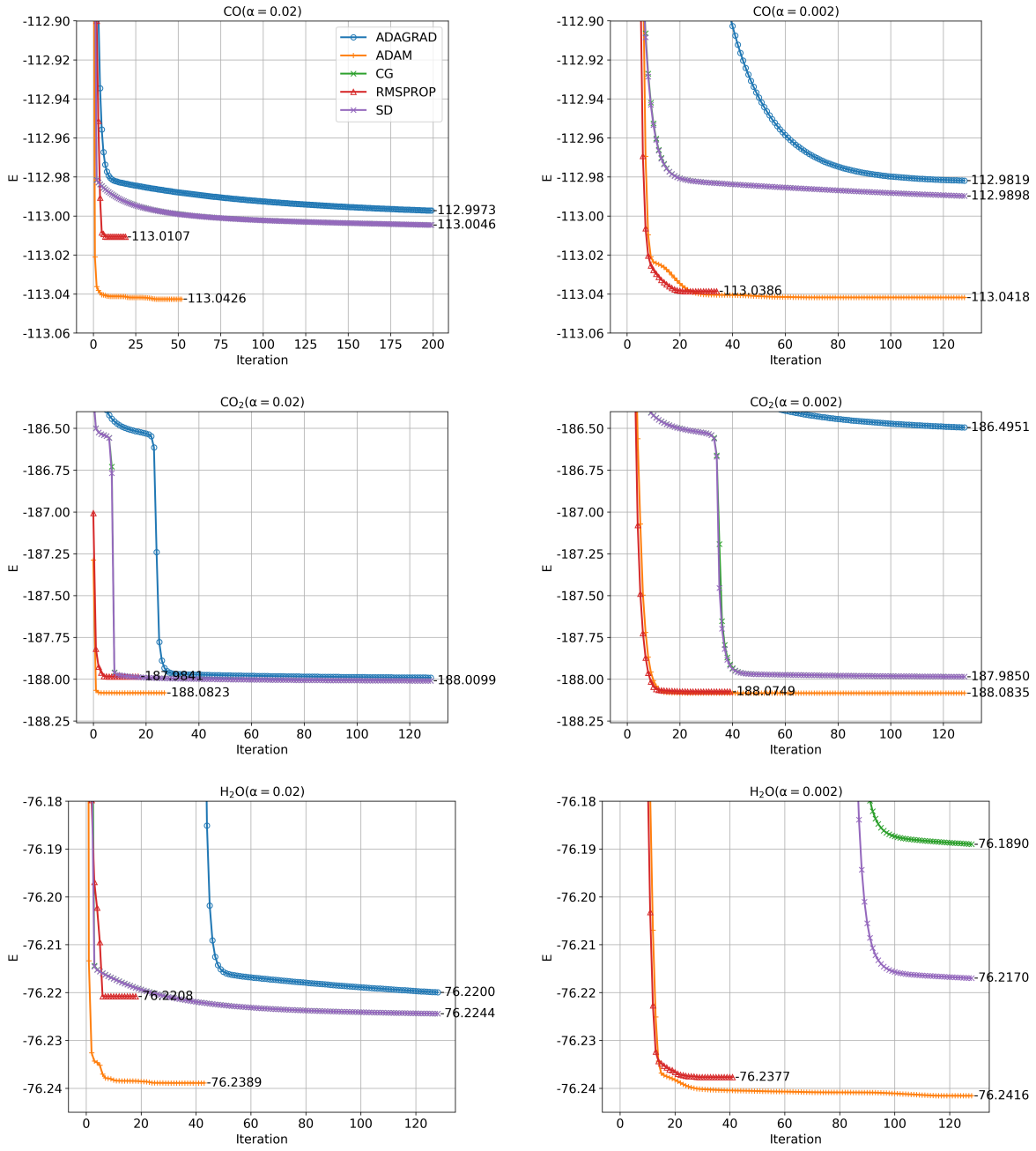

Figure 3: Convergence of NOF energies (in a.u.) for different molecules versus iteration number, using various optimization algorithms with selected α values; 0.02 in the left, and 0.002 in the right.

Adaptive methods, particularly Adam, often shows convergence across the three NOF test cases, (CO, CO_2, and H_2O). It typically reaches low energy values in fewer iterations and is less prone to stalling. RMSProp also shows good stability, although

it sometimes requires more iterations to reach similar energies.

Conjugate Gradient (CG) converges quickly in terms of iteration count, but it occasionally stalls early or converges to higher energies, suggesting that it may be efficient but less reliable under the imposed stopping rules. Steepest Descent (SD) consistently underperforms in both energy and convergence rate, often reaching a plateau far from the minimum and stopping prematurely due to lack of progress.

AdaGrad shows mixed results: while it can perform well for some molecules, it is sensitive to both the initial conditions and the learning rate, and tends to get trapped under the early-stopping condition more often than Adam or RMSProp.

This trend is particularly evident in the case of adaptive algorithms such as Adam and RMSProp. For example, Adam with $\alpha = 0.02$ consistently reaches the lowest final energies in all three molecules while requiring significantly fewer iterations than with $\alpha = 0.002$. RMSProp exhibits a similar pattern, although with slightly higher final energies.

In contrast, algorithms like SD and CG show limited benefit from increasing α. In many cases, the number of iterations remains close to the maximum allowed, and the final energy either improves only marginally or not at all. AdaGrad shows inconsistent sensitivity: in some cases it benefits from the larger step size, while in others it fails to converge efficiently and gets trapped by the stopping criteria.

As seen earlier, choosing an appropriate α depends on both the algorithm and the system. In the case of NOF calculations, adaptive methods show good consistency, yet their performance is still influenced by the choice of α. This motivates a more systematic study of how learning rates influence convergence, and naturally leads to exploring a dynamic learning rate strategy, where α is adjusted during the optimization process.

For this purpose, three additional molecules were selected (CO_2, BF_3, and C_4H_4) to examine how different α values affect both convergence speed and final energy (Figure 4).

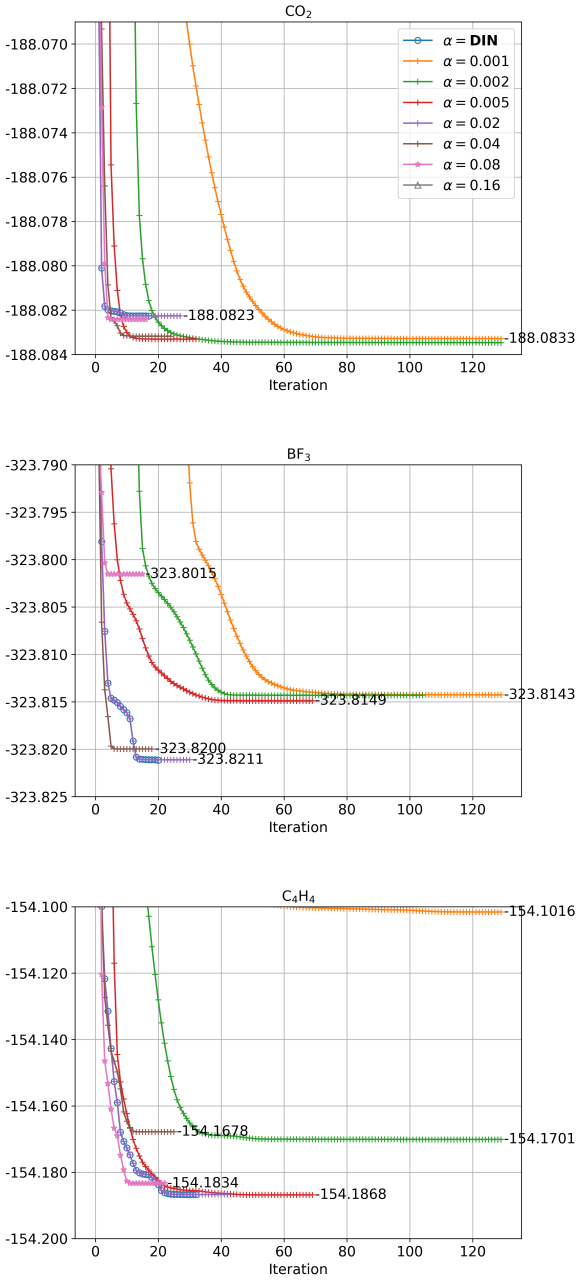

Figure 4: Convergence of NOF energies (in a.u.) with respect to the number of iterations for different molecules, using different values of α. Note that α=**DYN** stands for the Adam modification with dynamic modification of the parameter.

The results confirm that smaller values of α generally lead to lower final energies, but at the cost of significantly more iterations, often reaching the maximum limit of 130. (Note that in the case of C_4H_4, where calculations with $\alpha = 0.002$ or smaller

take noticeably longer (up to 30 minutes), posing a potential challenge in real-time classroom settings). In contrast, larger values of α (e.g., 0.02) converge in fewer steps but may settle prematurely at higher energy plateaus, reflecting a loss of precision and stability. This trade-off between speed and accuracy, already observed in the previous section, is now reinforced with quantitative detail, providing a natural motivation for exploring adaptive strategies. These observations set the stage for introducing a modified version of Adam with an adjustable -scheduled- learning rate, designed to achieve faster convergence while maintaining accuracy in energy minimization.

The results for CO_2, BF_3, and C_4H_4 show that this adaptive strategy leads to both fast and stable convergence. In each case, the number of iterations needed is significantly lower than for small fixed α values, and the final energies are comparable to those achieved with the best static choices. Notably, for C_4H_4, the dynamic learning rate version reaches convergence in just 30 iterations, compared to the 130-step maximum reached when using $\alpha = 0.001$, with nearly identical final energy.

This dynamic α study reinforces the idea that careful parameter tuning can enhance both efficiency and accuracy. As a final integrative step, the students apply the algorithms and α values they have selected, based on previous analyses, to a real physical problem: the dissociation of the hydrogen molecule. In doing so, they also put their Python programming skills into practice, reinforcing the link between computational implementation and theoretical concepts.

H_2 dissociation: Correlation effects

As the internuclear distance increases, the HF method imposes a single-determinant wavefunction that artificially maintains spin pairing, leading to an unphysical dissociation limit in which the electrons remain incorrectly delocalized. This well-known failure is referred to as the homolytic bond dissociation problem in HF theory.

To overcome this limitation, one common strategy in traditional quantum chemistry is to move beyond HF by employing either the unrestricted-HF formalism, which allows spin symmetry breaking at the cost of spin contamination, or, more rigorously,

methods such as NOFs that include static and dynamic electron correlation by directly incorporating electron correlation through occupation numbers and natural orbitals. NOF methods correctly capture the dissociation limit of H_2, as illustrated in Figure 5.

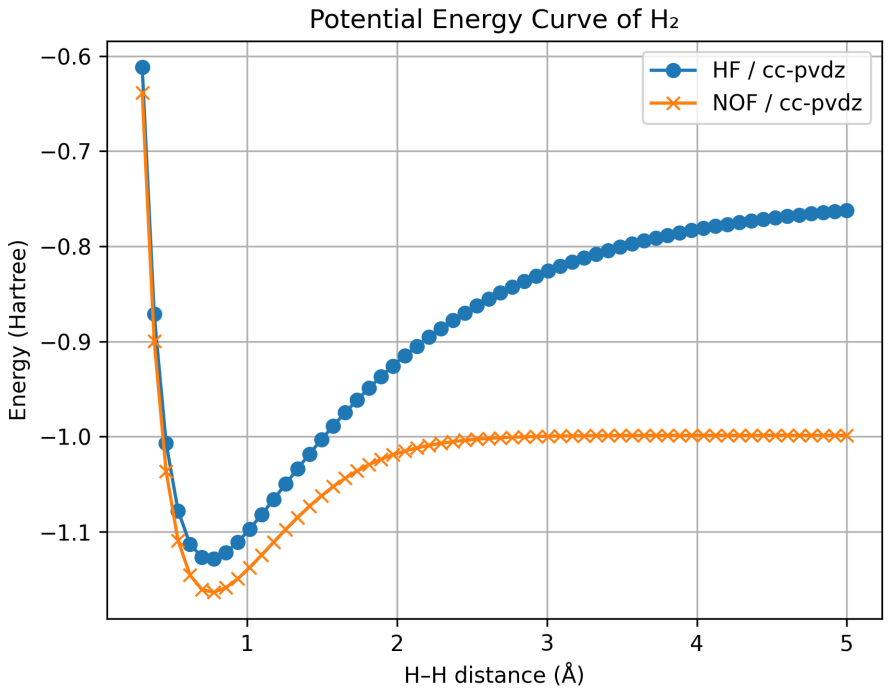

Figure 5: H_2 dissociation curve calculated with NOF and HF.

Conclusions

By incorporating optimization strategies originally developed for deep learning, this module highlights the potential of modern algorithms, such as AdaGrad, RMSProp, and especially Adam, to improve the efficiency of quantum chemical calculations. Although these methods are widely used in machine learning, their application to electronic structure problems remains limited. Exposing students to these techniques not only enriches their computational toolkit but also encourages critical thinking about the transfer of methodologies across scientific domains and the future development of faster, more robust minimization strategies in quantum chemistry.

The structured progression from HF to NOF calculations helps students appre-

ciate how the complexity of the optimization landscape increases with the level of electronic correlation. Through systematic comparisons of classical and modern algorithms, students are able to observe convergence behavior, sensitivity to learning rate, and algorithm consistency across different systems.

The module also emphasizes practical constraints typical of classroom environments, such as time limits and computational resources, and shows how these impact the choice of methods and parameters. The final extension, which explores the benefits of a dynamic learning rate in the Adam optimizer, reinforces the broader lesson that algorithmic adaptation and careful parameter tuning are essential in modern computational chemistry.

Altogether, this module provides an adaptable and practical approach to incorporating optimization theory into quantum chemistry education, and it encourages a more nuanced and research-aware understanding of computational tools among students. In its current implementation within the Master's course "Numerical Analysis and Computational Techniques" (TCCM program), the module includes a one-hour introductory lecture on classical and modern optimization methods, followed by two 2-hour sessions where students implement selected algorithms and perform calculations both independently and with guidance. These sessions provide a context-rich environment for analyzing convergence behavior and algorithmic performance, practicing Python programming skills, and examining the effects of electronic correlation.

With all supporting materials openly available, instructors can readily integrate this module into their courses, adapt it to different levels and formats, and extend it to other molecular systems, thereby fostering active engagement with optimization methods in quantum chemistry.

Acknowledgements

OpenAI's ChatGPT[50] was used as a support tool in the preparation of this manuscript, assisting with writing and with programming guidance in the generation of Python

plots. The authors thank for technical and human support provided by IZO-SGI (SGIker) of UPV/EHU and European funding (ERDF and ESF) and the DIPC. Financial support comes from Eusko Jaurlaritza (Basque Government) through the project IT588–22.

Supporting Information

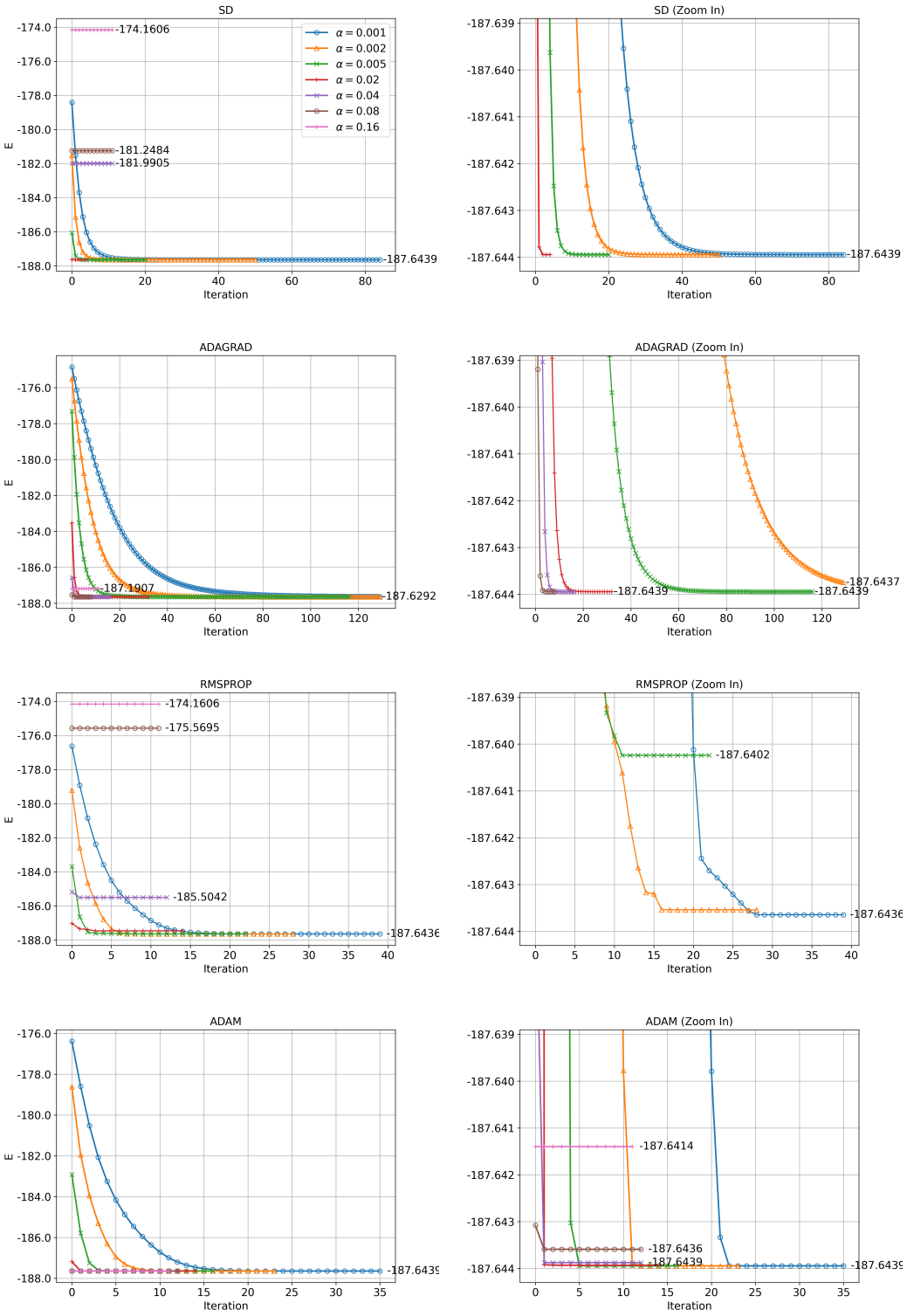

Figure 6: CO_2 Hartree-Fock energies convergence (in a.u.) *vs.* the iteration number is plotted for different learning rates (α). Full representations are shown at the left column labeled with the used algorithm. At the right column lowest energy region is zoomed.

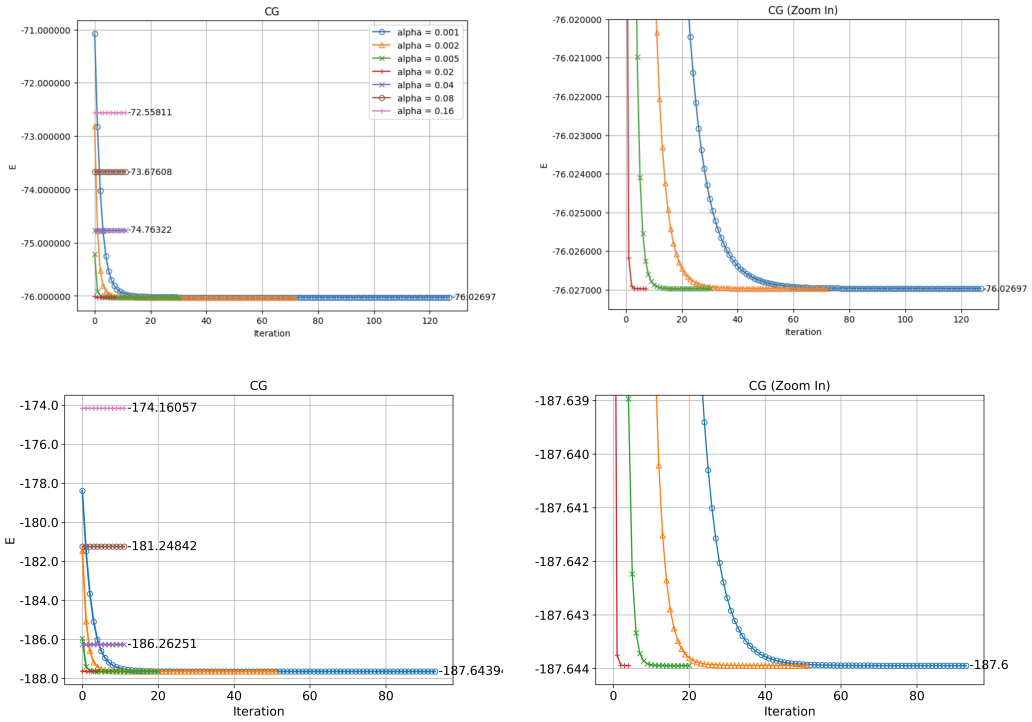

Figure 7: H_2O (top) and CO_2 bottom, HF energies convergence (in a.u.) *vs.* the iteration number is plotted for different learning rates (α). Full representations are shown at the left column labeled with the used algorithm. At the right column lowest energy region is zoomed.

All materials, including the Jupyter Notebook[51] and installation instructions, are available at: `https://github.com/your-repo-link`

References

(1) Kochenderfer, M. J.; Wheeler, T. A. *Algorithms for Optimization*; The MIT Press, 2025.

(2) Chapra, S. C.; Canale, R. P. *Numerical Methods for Engineers*, 6th ed.; McGraw-Hill: New York, 2010.

(3) Nocedal, J.; Wright, S. *Numerical Optimization*, 2nd ed.; Springer: New York, 2006.

(4) Jensen, F. *Introduction to computational chemistry*, third edition ed.; Wiley: Chichester, UK ; Hoboken, NJ, 2017.

(5) Chapra, S. C.; Canale, R. P. *Numerical methods for engineers*, 6th ed.; McGraw-Hill Higher Education: Boston, 2010.

(6) Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* **2011**, *12*, 2121–2159.

(7) Hinton, G. Neural Network for Machine Learning. Unpublished, 2012; `https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf`, Last accessed September 2025.

(8) Kingma, D. P.; Ba, J. ADAM: A Method for Stochastic Optimization. 2017; `https://arxiv.org/abs/1412.6980`, Last accessed September 2025.

(9) Lew-Yee, J. F. H.; Del Campo, J. M.; Piris, M. Advancing Natural Orbital Functional Calculations through Deep Learning-Inspired Techniques for Large-Scale Strongly Correlated Electron Systems. *Physical Review Letters* **2025**, *134*, 206401.

(10) Bergholm, V. et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv [quant-ph]* **2018**,

(11) Szabo, A.; Ostlund, N. S. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*; Courier Corporation, 2012.

(12) Cramer, C. J. *Essentials of Computational Chemistry: Theories and Models*; John Wiley & Sons, 2004.

(13) Pilar, F. L. *Elementary Quantum Chemistry, Second Edition*; Courier Corporation, 2013; Google-Books-ID: CWLCAgAAQBAJ.

(14) Piris, M. Exploring the potential of natural orbital functionals. *Chemical Science* **2024**, *15*, 17284–17291.

(15) Lew-Yee, J. F. H. pyNOF. 2024; `https://doi.org/10.5281/zenodo.10627612`, Last accessed September 2025.

(16) Piris, M.; Mitxelena, I. DoNOF: An open-source implementation of natural-orbital-functional-based methods for quantum chemistry. *Computer Physics Communications* **2021**, *259*, 107651.

(17) Donostia Natural Orbital Functionals. https://github.com/DoNOF/DoNOFsw, Last accessed September 2025.

(18) Szymanski, J. J. Using Hartree–Fock Molecular Orbital Calculations to Illustrate Trends in Bonding. *J. Chem. Educ.* **2008**, *85*, 159.

(19) Halpern, A. M. Student Use of Computational Chemistry for Exploring Molecular Properties. *J. Chem. Educ.* **2006**, *83*, 1243.

(20) Kastner, J.; Welch, B. R. Exploring the Quantum Mechanical Foundations of Molecular Geometry Using Gaussian. *J. Chem. Educ.* **2019**, *96*, 386–389.

(21) Pérez, E.; Romero, N. Teaching Hartree–Fock Theory with Real Molecules. *J. Chem. Educ.* **2024**, *101*, 742–749.

(22) Matito, E.; Duran, M.; Solà, M. A Novel Exploration of the Hartree–Fock Homolytic Bond Dissociation Problem in the Hydrogen Molecule by Means of Electron Localization Measures. *Journal of Chemical Education* **2006**, *83*, 1243.

(23) Smith, D. G. A. et al. Psi4NumPy: An interactive quantum chemistry programming environment for reference implementations and rapid development. *J. Chem. Theory Comput.* **2018**, *14*, 3504–3511.

(24) Fransson, T.; Winter, N.; Ringholm, M.; et al. eChem: A Notebook Exploration of Quantum Chemistry. *Journal of Chemical Education* **2023**, *100*, 319–329.

(25) Cruzeiro, V. W. D.; Rohde, G. K. Implementing New Educational Platforms in the Classroom: An Interactive Approach to the Particle in a Box Problem. *Journal of Chemical Education* **2019**, *96*, 1456–1460.

(26) The Erasmus Mundus Joint Masters (EMJM) in Theoretical Chemistry and Computational Modelling (TCCM). `https://www.emtccm.org/`, Last accessed September 2025.

(27) Frisch, M. J. et al. Gaussian˜16 Revision C.01. 2016; Gaussian Inc. Wallingford CT.

(28) Smith, D. G. A. et al. PSI4 1.4: Open-source software for high-throughput quantum chemistry. *The Journal of Chemical Physics* **2020**, *152*, 184108.

(29) Neese, F. Software Update: The ORCA Program System—Version 6.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2025**, *15*, e70019.

(30) Pulay, P. CONVERGENCE ACCELERATION OF ITERATWE SEQUENCES. THE CASE OF SCF ITERATION. *Chemical Physics Letters* **1980**, *73*, 393–398.

(31) Epifanovsky, E.; others Software for the frontiers of quantum chemistry: An overview of developments in the Q-Chem 5 package. *Journal of Chemical Physics* **2021**, *155*, 084801.

(32) Neese, F. The ORCA quantum chemistry program package. *Journal of Chemical Physics* **2020**, *152*, 224108.

(33) Helmich-Paris, B. A trust-region augmented Hessian implementation for restricted and unrestricted Hartree–Fock and Kohn–Sham methods. *The Journal of Chemical Physics* **2021**, *154*, 164104.

(34) Elayan, I. A.; Gupta, R.; Hollett, J. W. NO and the complexities of electron correlation in simple hydrogen clusters. *The Journal of Chemical Physics* **2022**, *156*, 094102.

(35) Cartier, N. G.; Giesbertz, K. J. H. Exploiting the Hessian for a Better Convergence of the SCF-RDMFT Procedure. *Journal of Chemical Theory and Computation* **2024**, *20*, 3669–3682.

(36) Mitxelena, I.; Piris, M.; Ugalde, J. M. In *Advances in Quantum Chemistry*; Ancarani, L. U., Hoggan, P. E., Eds.; State of The Art of Molecular Electronic Structure Computations: Correlation Methods, Basis Sets and More; Academic Press, 2019; Vol. 79; pp 155–177.

(37) Piris, M. Global Natural Orbital Functional: Towards the Complete Description of the Electron Correlation. *Physical Review Letters* **2021**, *127*, 233001.

(38) Lew-Yee, J. F. H.; Piris, M.; M. Del Campo, J. Resolution of the identity approximation applied to PNOF correlation calculations. *The Journal of Chemical Physics* **2021**, *154*, 064102.

(39) Franco, L.; Bonfil-Rivera, I. A.; Huan Lew-Yee, J. F.; Piris, M.; M. Del Campo, J.; Vargas-Hernández, R. A. Softmax parameterization of the occupation numbers for natural orbital functionals based on electron pairing approaches. *The Journal of Chemical Physics* **2024**, *160*, 244107.

(40) Élodie Boutou; Lew-Yee, J. F. H.; Mercero, J. M.; Piris, M. *Advances in Quantum Chemistry*; Academic Press, 2025.

(41) Piris, M. In *Advances in Quantum Chemistry*; Quintana, R. A. M., Stanton, J. F., Eds.; Novel Treatments of Strong Correlations; Academic Press, 2024; Vol. 90; pp 15–66.

(42) Mercero, J. M.; Ugalde, J. M.; Piris, M. Chemical reactivity studies by the natural orbital functional second-order Møller–Plesset (NOF-MP2) method: water dehydrogenation by the scandium cation. *Theoretical Chemistry Accounts* **2021**, *140*, 74.

(43) Mercero, J. M.; Grande-Aztatzi, R.; Ugalde, J. M.; Piris, M. In *Advances in Quantum Chemistry*; Hoggan, P. E., Ed.; Academic Press, 2023; Vol. 88; pp 229–248.

(44) Piris, M.; Lopez, X.; Ugalde, J. M. Time-Resolved Chemical Bonding Structure Evolution by Direct-Dynamics Chemical Simulations. *The Journal of Physical Chemistry Letters* **2024**, *15*, 12138–12143.

(45) Conn, A. R.; Scheinberg, K.; Vicente, L. N. *Introduction to Derivative-Free Optimization*; Society for Industrial and Applied Mathematics, 2009.

(46) Nelder, J. A.; Mead, R. A Simplex Method for Function Minimization. *The Computer Journal* **1965**, *7*, 308–313.

(47) Curry, H. B. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics* **1944**, *2*, 258–261.

(48) Hestenes, M. R.; Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* **1952**, *49*, 409–436.

(49) Johnson, R. D. Computational chemistry comparison and benchmark database, NIST standard reference database 101. 2002.

(50) OpenAI ChatGPT. `https://chat.openai.com`, 2025; Last accessed September 2025.

(51) Kluyver Thomas; Ragan-Kelley Benjamin; Pérez Fernando; Granger Brian; Bussonnier Matthias; Frederic Jonathan; Kelley Kyle; Hamrick Jessica; Grout Jason; Corlay Sylvain; Ivanov Paul; Avila Damián; Abdalla Safia; Willing Carol; Jupyter Development Team *Positioning and Power in Academic Publishing: Players, Agents and Agendas*; IOS Press, 2016.