Chapter 8

PSI4EDUCATION: Free and Open-Source Programing Activities for Chemical Education with Free and Open-Source Software

D. Brandon Magers,[1,*] Victor H. Chávez,[2] Benjamin G. Peyton,[3] Dominic A. Sirianni,[4] Ryan C. Fortenberry,[5] and Ashley Ringer McDonald[6]

[1]Department of Chemistry & Physics, Belhaven University, Jackson, Mississippi 39202, United States

[2]Department of Chemistry, Purdue University, West Lafayette, Indiana 47907, United States

[3]Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24060, United States

[4]Department of Chemistry, University of Richmond, Richmond, Virginia 23713, United States

[5]Department of Chemistry & Biochemistry, University of Mississippi, University, Mississippi 38677, United States

[6]Department of Chemistry & Biochemistry, California Polytechnic State University, San Luis Obispo, California 93407, United States

[*]Email: bmagers@belhaven.edu.

Computational molecular science, including computational chemistry, programming, data science, and machine learning, has become a cornerstone of chemistry research, but traditional, undergraduate chemistry curricula spend little-to-no time developing these skills in early career chemists. The most common best-case scenario has students using computational chemistry tools with a few exercises in a physical chemistry course. Many degree programs have no exposure whatsoever for students to using computational tools or programming. In either case, a large issue precluding uptake of this medium as both a skill and educational tool is a matter of access to and administration of quality hardware, user-friendly software packages, and correspondingly robust student exercises. Additionally, computation can offer a "dry" alternative for a benefit wet laboratory provides in chemical education: a means by which to reinforce conceptual learning through tangible experience with the phenomena being studied. To this end, the PSI4EDUCATION community utilizes the free and open-source PSI4 quantum chemistry package, Python and Jupyter Notebooks, and the WEBMO graphical user interface to facilitate the growth and utilization of computational molecular science in the classroom. Additionally, the community has pioneered the use of cloud-based computing for student access, software hosting, and quantum chemistry program execution. This promises to create a plug-and-play environment with little overhead work for the instructor and costs below the level of most consumables in a traditional chemistry laboratory exercise. PSI4EDUCATION is continuously growing our resource database through contributions from the larger chemistry education community, and dozens of exercises have thus far been developed. In this narrative, the utilization of programming via Jupyter Notebooks within chemistry education takes focus. Examples of current activities discussed herein include analysis of bond breaking, exploration of the Hartree-Fock procedure, and initial forays into machine learning applications to chemistry.

Ringer McDonald and Nash; Teaching Programming across the Chemistry Curriculum
ACS Symposium Series; American Chemical Society: Washington, DC, 2021.

Introduction

Computing, and the technical skills needed to fully utilize it, are driving nearly every aspect of modern commerce including research and development. However, few chemistry undergraduate degree programs have been able to meet this need in their curricula due to the already heavy knowledge load expected of competent B.S. chemistry majors ranging from proficiency in the four (five) areas of chemistry to instrumentation utilization to generic laboratory safety (*1*). On top of this, the standard undergraduate chemistry course sequence is largely built around historical traditions and professional school demands (*2*–*5*). Hence, the current education of chemistry students is providing them with 20th century skills for a 21st century workplace. Tools to shift this are emerging, but educators must invent creative ways or utilize existing means developed by our professional colleagues to fill this knowledge gap in computer skills, a gap that is growing exponentially with each successive graduating class of chemistry majors.

Previous efforts to include exposure to computational tools within the chemistry curriculum have largely originated within the physical chemistry community. This is partly due to the nature of the subdiscipline itself as highly mathematical in nature and partly due to the bent of those who call themselves physical chemists. MATLAB, Mathematica, and Microsoft Excel are far superior tools for doing difficult computations involving integration by parts or considerations of a particular partition function than paper and pencil work alone. Computational tools can also aid in bridging conceptual connections across subdisciplines as students try to become proficient in the four (five) areas of chemistry (*6*–*9*), as cross-discipline connections are important to overall undergraduate curricula (*5*, *10*–*14*). Quantum chemical computations built on compiled imperative programming languages have also become nearly required companions for experimental research publications. While the implementation of both types of computer skills (mathematical analysis and computational chemistry) are now fairly routine within upper-level physical chemistry courses (*15*, *16*), even these skills within chemistry majors are slowly receding from the forefront of needs within the modern chemical workforce (*17*).

The emerging need is for chemists of all disciplines to be able to construct their own computational tools in order to push research forward (*1*, *18*). This requires programming. The physics community has required programming competency of its practitioners, even those who consider themselves experimental physicists, for decades. Chemists cannot neglect this set of skills any longer (*19*).

The analytical and experimental physical communities of chemists have been requiring graphical programming (often through software such as LabVIEW) and Unix-based operating system competency for some time now (*20*) in addition to the eternal need for programming within theoretical chemistry. However, the experimental organic chemist must now be able to construct numerical analyses in many cases in order for their data to be interpreted in novel ways (*21*). Anecdotally, recent work by one of these authors provided a relatively simple tangent line analysis program (*22*) to the organic solar cell community that promises to reduce analysis time and error by as much as 5%, a major amount in a field with big dollar applications on the line. Had an experimental, synthetic organic chemistry graduate student had these skills, such a need would not have been left unmet for so long.

Such examples are myriad, but the largest barrier to providing generic chemistry students with programming skills does not lie with desire on our part as educators. Time to cover such material in class or laboratory, time to learn programming skills ourselves, and time to construct and evaluate

meaningful exercises is often in short supply. However, the PSI4EDUCATION team has been developing these types of tools to produce 21st century skills in 21st century chemistry students. These tools and exercises for the students do not require a significant overhaul of the existing chemistry curriculum, but they should be implemented throughout the curriculum. Naturally, they often start in physical chemistry, but they should not be limited to that subdiscipline.

PSI4EDUCATION

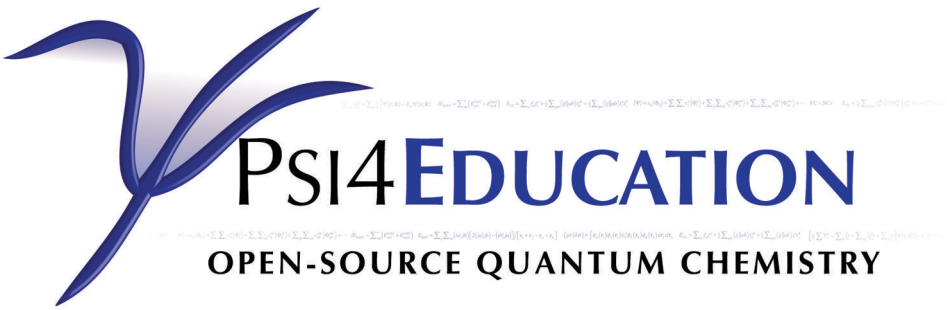

Figure 1. PSI4EDUCATION Logo.

PSI4EDUCATION (Figure 1) is the education and outreach arm of PSI4, a free, open-source, quantum chemistry software package (*23*, *24*). PSI4EDUCATION provides a suite of free, open-source lab and classroom activities for use in courses across the undergraduate chemistry curriculum (https://psicode.org/posts/psi4education/). These activities utilize PSI4 as a quantum chemistry engine, PSI4NUMPY (an interactive quantum chemistry programming environment) (*25*), PSIAPI Jupyter Notebooks (*26*), and WEBMO (*27*). A major goal of PSI4EDUCATION is to increase student exposure to scientific programming and help students learn chemistry concepts through computational resources. While the PSI4EDUCATION consortium has been previously discussed in the literature (*16*), this particular chapter will focus on the developments made in the past six years, specifically in the realm of including programming and computational problem solving for chemical applications.

Some base knowledge Python scripting skills are required as a prerequisite for many of the PSI4EDUCATION activities. Many B.S. chemistry majors do not take a programming course as a requirement of their degree; instead, this base knowledge must also be incorporated into the current curriculum. The Molecular Sciences Software Institute (MolSSI) (*28*) funded by the NSF provides a series of free lessons on "Python Scripting for Computational Molecular Science (*29*)" which can aid students in building the required prerequisite skills. Of note, lessons 1, 4, and 5 together cover all the prerequisite skills needed for all the PSI4EDUCATION activities including assigning variables, importing libraries, loops, logical expressions, etc. Additionally, the use of Jupyter Notebooks lowers the barrier to getting the students comfortable in a coding environment. A Jupyter notebook is a web application that allows the creation of documents with executable code, rich text formatting, and widgets. Use of a Jupyter notebook removes the need for the student to worry about remote connections and job submissions, and it lets the student focus more on learning chemistry through the use of programming.

Another major goal of PSI4EDUCATION is to decrease barriers of chemistry educators implementing computational educational resources in classrooms. Often, a chemistry department does not have a faculty or staff member that can dedicate the time and energy needed to manage a

server to host many traditional computational chemistry resources. This is especially true at primarily undergraduate institutions (PUIs) and community colleges. PSI4EDUCATION utilizes many partnerships and options to make implementing the activities as plug-and-play as possible without the need of a dedicated, local server or workstation. Additionally, WEBMO and PSI4 can be installed on Google Cloud, removing the need for any local maintenance at a minimal cost. For the PSIAPI Jupyter Notebook activities, PSI4EDUCATION has partnered with the Chem Compute Science Gateway (ChemCompute; https://chemcompute.org/) (*30*) to offer free computational resources without the need to install or configure any software or hardware.

WebMO Labs (Graphical Interface) Prerequisites:
Build molecules
Run energy calculation
Run geometry optimization

WebMO

Activity	**Area of Knowledge**	**Time (Hours)**
Is C_3H^+ present in the Horsehead nebula?	Astro, Analytical, Physical	2.5
What is the radius of an atom?	Physical	1.5
Can we visually predict binding energies?	Organic	1.5
What happens to the orbitals as one-dimensional box gets longer?	Physical	1.5
What makes a molecule polar or non-polar?	General	0.75
What is symmetry in chemistry?	Physical, Inorganic	2.5
When does water become OH + H?	Physical	1.5

Programming in Python Prerequisites:
Import libraries
Data types
Slicing of Lists and Arrays
Logic statements and loops

PsiAPI Labs (Python Interface) Prerequisites:
Build Molecules
Run energy calculation
Run optimization calculation

Psi4

Activity	**Area of Knowledge**	**Time (Hours)**
Basis sets	Physical	3
Bond breaking of H_2	Physical, Organic, Inorganic	3
Hartree-Fock procedure for approximate quantum chemistry	Physical	3
Linear Algebra	Physical	3
Machine-learning applications in computational chemistry	Physical	3
Determining Structure from Microwave Spectroscopy	Physical, Analytical	3
Symmetry-Adapted Perturbation Theory (SAPT)	Physical	3
Spectroscopic Constants	Physical, Analytical	2

Figure 2. List of the currently available exercises organized by the framework used (WebMO and PSIAPI). More details can be found on the PSI4 website (https://psicode.org/posts/psi4education/).

Through the PSI4EDUCATION PSIAPI activities utilizing Jupyter Notebooks, a breadth of scientific programming experience can be exposed to students. As would also be expected in a foundational programming course, students gain experience with program data types, logical expressions, and data manipulation. However, in addition to these experiences, these activities also include many computational science packages such as NUMPY, SciPy, Matplotlib (*31*), and scikit-learn. All of this is couched within standard chemical applications giving students a more familiar environment for developing their programming skills while working through these chemical problems with new skills. A list of the currently available exercises is given in Figure 2; this delineates which exercises use WEBMO and which exercises use PSIAPI. A complete list of the exercises

along with learning objectives and implementation details are available on the PSI4 website (https://psicode.org/posts/psi4education/).

Example Activities

Three specific PSI4EDUCATION activities are highlighted which provide a means of teaching programming within the chemistry curriculum. Each has been developed by different groups and have been implemented in at least one instance. On the PSI4 website where the PSI4EDUCATION materials are hosted, learning outcomes, time to complete the assignments, and other educational metadata are available for any and all of the exercises provided. We begin this sampling of developed exercises with bond breaking, one of the most fundamental processes in chemistry.

Bond Breaking

This activity introduces students to concepts intrinsic within diatomic molecular dissociation. Namely, many theoretical models fail to predict the correct behavior of diatomics at long separation distances, and thus, this PSI4EDUCATION activity investigates the bond-breaking reaction in the H_2 molecule. The exercise could be used in an advanced general chemistry course, a physical-organic course, or physical chemistry course.

The exercise begins by introducing the PSIAPI via a set of single-point energy calculations. The student then uses Python functions to generate multiple calculations and parse their results. For example, running jobs for various bond distances simply requires using a loop in which a new line replaces the separation between fragments. Beyond providing quick access to data, one of the most appealing attributes of this activity is its focus on visualization despite being a programming-based exercise. The exercise requires critical thinking about the orbitals. Immediately plotting the orbitals is a critical element in understanding the differences in the calculations. The orbitals can be plotted in various plotting libraries such as Matplotlib or Plotly (*32*) mentioned above. The addition of plots turns a dry mathematical demonstration into an interactive and visually striking proof (Figure 3). The MolSSI has recently developed a tutorial describing how to generate impactful and aesthetically pleasing plots (*33*).

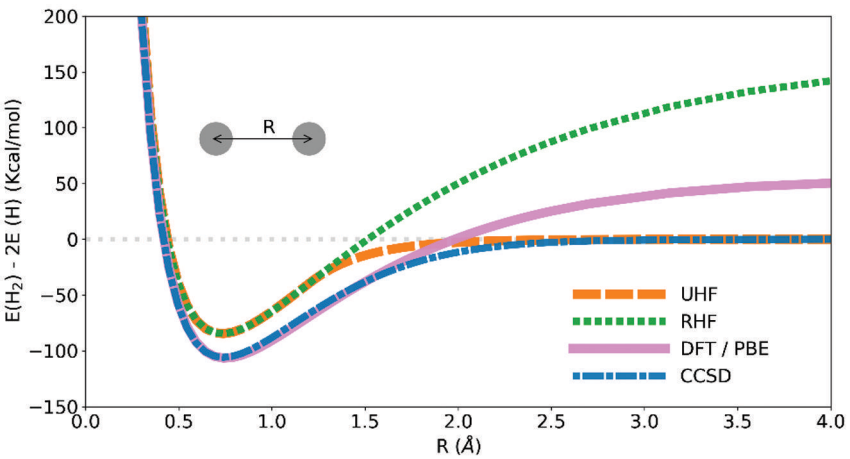

*Figure 3. 6-31G** Potential Energy curves for H_2 generated using Matplotlib.*

The student analyzes the plots and energies in order to determine differences in how bonds may be represented. For example, Hartree-Fock and density functional theories model bonds in different ways that are appropriate at the equilibrium geometry, but will produce spurious results for longer internuclear distances. The student uses coupled-cluster theory to generate an exact result to compare other methods against it. Restrictred and unrescrited references are introduced in order to address the previously described unphysical behavior of the Hartree-Fock solution. A way to visualize the dissociation problem is to plot the orbitals at the stretched geometry. This asymmetry is only captured by the unrestricted model (*34*) (Figure 4). Instead of dealing with separate calculation and visualization software, this activity lets the student do all the work in the same environment. The tasks of generating atomic orbitals, adding them to generate particular solutions, and visualizing them can be done in successive cells with ease. Furthermore, every parameter including the method of calculation, basis set, and atoms used can be easily modified allowing the student to try different settings and see the effects immediately. Tools like this are present in all of PSI4EDUCATION's activities, and they give the students firsthand exposure to every part of a calculation.

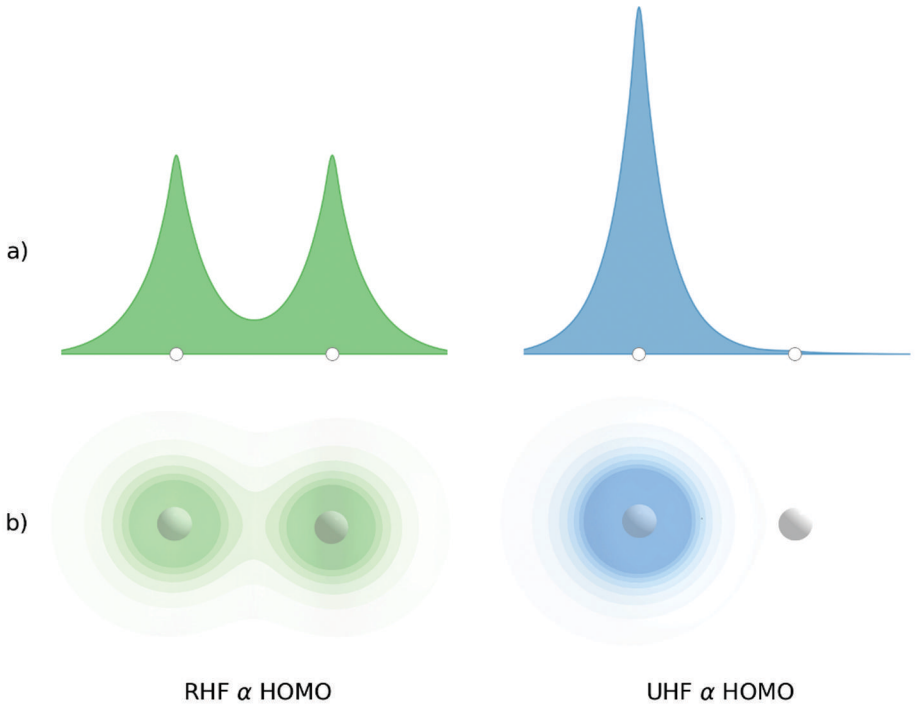

Figure 4. Alpha molecular orbital of the RHF and UHF solutions for the stretched H_2 visualized using a) Matplotlib and b) Moly (35), a molecular library built on top of Plotly.

Student Learning Outcomes

The instructor version of the exercise provides information about the activity scope, required student prerequisite knowledge, an expected schedule, and student learning objectives (SLOs). Upon completing this activity, students are expected to be able to:

1. Valuate different computational methods for bond breaking.
2. Compare restricted and unrescrited references.
3. Benchmark these results with respect to results from coupled-cluster theory.
4. Visualize molecular orbitals using plotting libraries.

Hartree-Fock Self-Consistent-Field Theory

Hartree-Fock theory and its self-consistent-field (SCF) formulation are the cornerstone of wavefunction-based electronic structure theory, enabling understanding of both chemical bonding and molecular structure in terms of chemically meaningful molecular orbitals constructed from a linear combination of atomic orbitals (LCAO-MO). Despite the discussion of LCAO-MO theory in an inorganic chemistry course, both the mathematical and technical details necessary to develop Hartree-Fock theory and implement it into working computer code are generally considered to be outside the scope of an undergraduate chemistry curriculum. However, by leveraging the interactive Jupyter notebook programming environment, a dry laboratory exercise has been developed to introduce the basics of HF theory and its implementation in the Python programming language, which is accessible at the undergraduate level using PSIAPI and the PSI4NUMPY quantum chemistry programming framework.

This activity, entitled "Hartree-Fock Self-Consistent-Field Theory," is broken into three main sections, comprised of both conceptual discussion and interactive exercises, which, taken together, guide students through the theoretical and technical considerations necessary to implement a HF-SCF code. This approach allows the activity to stand alone, with no need for students to complete other activities as a prerequisite (even though PSI4EDUCATION does offer, e.g., a detailed activity in linear algebra), as well as allowing for only certain sections to be assigned at the instructor's discretion. Below, each of these sections will be briefly discussed, with regards to their motivation and content, before finally discussing the SLO for the activity as a whole.

Primer: The Hydrogen Atom & Hartree-Fock Basics

Aside from the technical details of the Hartree-Fock procedure itself, the need for approximate computational methodologies is itself a conceptual hurdle that students must overcome. In order to familiarize students with performing such computations, this exercise begins by referencing the exact, quantum mechanical description of the hydrogen atom, compared with the energy computed by the Hartree-Fock procedure as provided by PSI4. In this manner, the concept of an approximate quantum chemistry approach is connected to the act of performing a computation with PSI4, allowing students to reinforce dynamically their learning while introducing the HF-SCF approach. Next, in order to implement a Hartree-Fock code, students must be familiar with the form of the Hartree-Fock equations. While the derivation of these equations has been treated in significant detail elsewhere (*34*), this is usually presented at the graduate level, and, as such, is beyond the scope of this activity. Instead of focusing on the derivation, therefore, the SCF equations are directly presented before spending time discussing the quantities which comprise them (e.g., the Fock matrix, F). Finally, this section provides a "blueprint" for the rest of the exercise, which prompts students to anticipate their learning, a key step in metacognition that has been shown to increase learning retention and understanding (*36*, *37*).

The wavefunction, $\psi(x)$ can be represented as a *column vector*, $|\psi\rangle$. The complex conjugate of the wavefunction, $\psi^*(x)$ is also represented by a vector which is the complex conjugate transpose of $|\psi\rangle$.

$$\psi(x) \rightarrow |\psi\rangle \quad \text{column vector}$$
$$\psi^*(x) \rightarrow \langle\psi| \quad \text{row vector}$$

The normalization of a wavefunction $\psi(x)$ is an integral

$$\int \psi^*(x)\psi(x)\,dx = 1.$$

In Dirac notation, it is replaced with a vector equation

$$\langle\psi|\psi\rangle = 1.$$

The orthogonality of two wavefunctions, $\psi_i(x)$ and $\psi_j(x)$, which, in integral notation, is

$$\int \psi_i^*(x)\psi_j(x)\,dx = 0,$$

becomes, in Dirac notation,

$$\langle\psi_i|\psi_j\rangle = 0.$$

Figure 5. Brief overview of linear algebra concepts and Dirac notation in the Hartree-Fock activity.

Review of Mathematical Concepts

The largest challenge for students when learning Hartree-Fock theory, and indeed in the study of quantum mechanics in general, is its mathematical complexity. In particular, Hartree-Fock theory relies both on variational calculus for its derivation and linear algebra concepts for its self-consistent-field formulation, with which most undergraduate chemistry majors will be unfamiliar. In order to meet the need for students to have a sufficient working understanding of these mathematical tools to fully engage with this activity, a review of relevant mathematical concepts is included. This includes a brief discussion of normalization, orthogonality, and basis sets, as well as Dirac notation (Figure 5). Once these concepts and notation are introduced, students complete a few short exercises which reinforce this conceptual knowledge with the skills of performing vector operations in Python using the NUMPY library.

Even though Hartree-Fock is a two-particle theory, where all relevant quantities can be represented as rank-2 arrays (i.e., matrices), the resulting matrix equations can be challenging to conceptualize directly, and, in turn, implement in computer code, even for students who have had formal training in linear algebra. This is especially true for the formulation of the Fock matrix,

$$\mathbf{F} = \mathbf{H} + 2\mathbf{J} - \mathbf{K}$$

where **H** is the one-electron core Hamiltonian matrix, and **J**, **K** are the Coulomb and exchange matrices, respectively. To form the Coulomb and exchange matrices, tensor contractions (generalized matrix-matrix products between two or more multidimensional arrays) must be performed between the rank-4 electron repulsion integrals and the density matrix, **D**.

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma)\, D_{\lambda\sigma}$$

$$K_{\mu\nu} = \sum_{\lambda\sigma} (\mu\lambda|\nu\sigma)\, D_{\lambda\sigma}$$

In order to simplify these equations for students with whom linear algebra may be unfamiliar, the addition of another layer of abstraction is useful to the discussion of linear algebra: introducing the Einstein summation convention for denoting tensor contractions. In the Einstein convention, repeated indices are assumed to be contracted over, simplifying the notation for defining, e.g., **J** and **K**.

$$J_{\mu\nu} = (\mu\nu|\lambda\sigma)\, D_{\lambda\sigma}$$

$$K_{\mu\nu} = (\mu\lambda|\nu\sigma)\, D_{\lambda\sigma}$$

On the surface, introducing this convention may seem to be a trivial simplification; however, thanks to the NumPy library function, numpy.einsum(), tensor contractions written in the Einstein summation convention can be directly translated from symbolic notation to executable computer code with a single command.

```
J = numpy.einsum('pqrs,rs->pq', I_pqrs, D_rs)
K = numpy.einsum('prqs,rs->pq', I_pqrs, D_rs)
```

This ease of translation between symbolic mathematical expression and computer code is the heart of the PSI4NUMPY quantum chemistry programming framework, which is used heavily in this exercise. Therefore, a brief primer on Einstein summation convention is included in this section so that students are familiar with it in advance of its use below.

Hartree-Fock Self-Consistent-Field Procedure

After students' understanding of mathematical and notational details has been reinforced in the sections above, they are finally ready to move on to implementation of their own Hartree-Fock program. This begins by using PSI4 through the PSIAPI application programming interface to read in a molecule and relevant basic quantities, such as the atomic orbital basis set. Next, students are guided through the use of PSIAPI and the examination of the atomic orbital (AO) basis set through the real AO overlap matrix, **S**, in analogy to the analysis carried out above in the review of linear algebra. Students are then prompted to verify that this "real" basis set is not orthonormal. Students are guided to construct a transformation matrix which is used to orthogonalize the AO basis set and simplify the Hartree-Fock equations into a genuine eigenvalue equation rather than a pseudoeigenvalue equation. Once the basis set has been orthogonalized, the next step in the SCF procedure is to build an initial guess for the Fock matrix. While many such guesses exist, the simplest such quantity, the core Hamiltonian matrix, is utilized herein before diagonalizing this guess Fock matrix to obtain guess orbitals and electron density. Next, the Coulomb and exchange matrices are constructed using these guess orbitals before students compute the Hartree-Fock energy for their guess wavefunction. Finally, students implement the SCF iterations using each of the previous steps as a template, reinforcing their learning while simultaneously challenging them to think about the iterative nature of solving the coupled Hartree-Fock equations.

Student Learning Outcomes

The instructor version of the exercise provides information about the activity scope, required student prerequisite knowledge, an expected schedule, and SLOs. Upon completing this activity, students are expected to be able to:

1. Recognize that the AO basis is not orthonormal and must be transformed.
2. Transform from one basis to another using a transformation matrix.
3. Recognize the iterative nature of the HF procedure.
4. Define convergence and use convergence criteria in a self-consistent procedure.
5. Compute MO energies and coefficients by diagonalizing the Fock matrix.

Machine Learning

In the six years since PSI4EDUCATION was first introduced, the use of machine-learning (ML) applications in the physical sciences has increased exponentially (*38*–*40*). Naturally, chemists have begun utilizing these techniques in increasingly complex and successful ways. While the underpinnings of ML are fundamental concepts with which B.S. chemistry majors are undoubtedly familiar, such as regressions and statistics, ML techniques are often seen as "magic" by the broader community. This is usually due to a combination of esoteric vocabulary and a liberal application of pre-built scientific software packages. As with many mathematical concepts, the understanding afforded by live, hands-on experience cannot be overstated. The complicated workflows involved in state-of-the-art ML research preclude the possibility of reproducing these models by hand in an undergraduate setting. With guidance, however, a programming package such as scikit-learn (*41*) offers a quick way to both rationalize and experiment with basic ML models.

The ML activity offered in PSI4EDUCATION, titled "Machine-learning applications in computational chemistry," does not aim to teach a student all of the aspects of ML. Instead, the focus is on teaching the student how to build, evaluate, and refine a ML model. One case study employed in the exercise is in predicting the potential energy surface (PES) of the symmetric stretch in a water molecule. Through this, the concepts of training sets, linear regression, and molecular representations are introduced, while scikit-learn (a Python module which implements a wide range of ML techniques) handles the optimization of the model parameters. A particular focus is given to the generation of molecular representations, which encode the pertinent molecular information and serve as the input to the ML model, which is an often overlooked detail (*42*, *43*). The student is presented with questions regarding the form and function of the Coulomb matrix representation (*44*), show below. This simple two-particle representation requires only the atomic numbers (Z) and bond distances (r_{ij}) between every pair of atoms.

$$C_{ij} = \begin{cases} \dfrac{Z_i Z_j}{r_{ij}} \; for \; i \neq j \\ 0.5 \; Z_i^{2.4} \; for \; i = j \end{cases}$$

By training a simple linear regression model (which is not expected to perform admirably), the student is given an example of one of the key ideas in ML research: testing and improving upon a model. Additional flexibility and non-linearity are achieved through the popular kernel ridge regression (KRR) model, where the representations are used to generate a covariance matrix called a kernel (specifically, the radial basis function kernel) (*45*, *46*). This model reproduces the surface to within milliHartree accuracy (Figure 6).

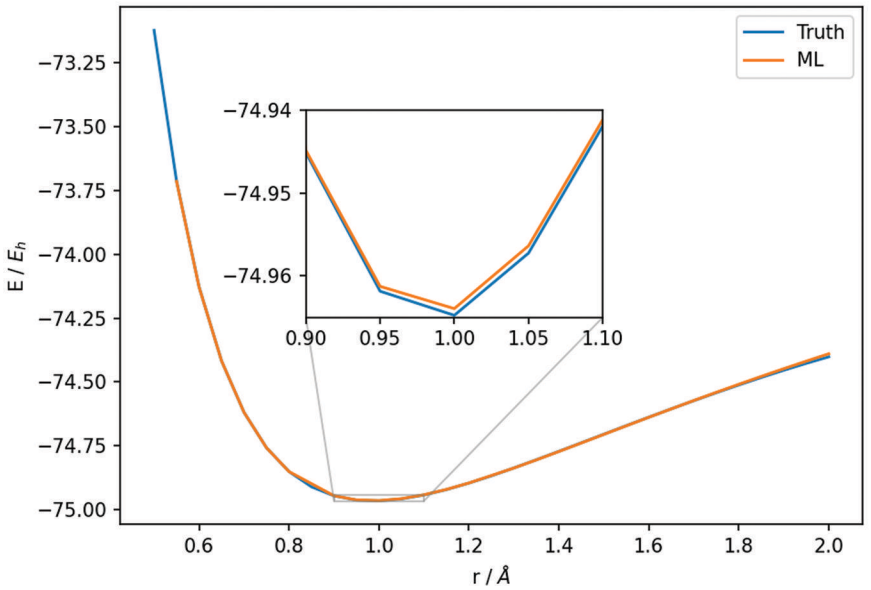

Figure 6. H_2O symmetric stretch PES, calculated using ab initio simulation ("Truth") and a KRR model ("ML").

Once again, the use of scikit-learn allows the student to bypass the linear algebra and implementation of the method and optimization routines needed to switch to this model, requiring only the understanding that a non-linear model with more flexibility is expected to perform better in an ML application. Armed with an effective routine for learning the symmetric stretch, the student is tasked with repeating the process for predicting the hypersurface of simultaneous O–H stretching. The student is encouraged to decide their own method for choosing training points and evaluate their performance relative to the canonical Hartree-Fock surface, which is computed for comparison.

Finally, a larger-scale application of the KRR model is demonstrated. Atomization energies are taken from a subset of the ANI-1 dataset (*47*, *48*), a collection of calculated thermochemical data spanning more than 50,000 small organic molecules in over 20 million conformations. The students are asked to apply the KRR model to predict the atomization energies of some of these molecules from a set of pre-computed Coulomb matrices. This exercise is adapted from the ML tutorial provided by the MolSSI for the QCArchive project (*49*) and constitutes an example of a "real-world" scenario. In the PSI4EDUCATION exercise, the student can see the organization of a large database, as well as probe the chemical diversity of the dataset through visualization inside the Jupyter notebook. Performance is plotted in a violin plot, and the student is asked to judge the efficacy of the model based on the reported error statistics.

Student Learning Outcomes

This exercise should serve to lower the barrier for introducing students to different ML applications in chemistry. The activity is intended for students familiar with the concept of atomization energies and the general PSI4EDUCATION requirements, but a deep understanding of the underlying theoretical chemistry is not required by design. This makes the activity appropriate for any undergraduate students beyond the general chemistry level but may include advanced sections of general chemistry. The instructor version of the exercise provides information about the activity

scope, required student prerequisite knowledge, an expected schedule, and SLOs. Upon completing this activity, students are expected to be able to:

1. Summarize basic ML concepts, such as training sets and parameter optimization.
2. Design geometry-based molecular representations.
3. Apply basic ML techniques across two- and three-dimensional potential energy surfaces.
4. Utilize large databases for the prediction of chemical properties.
5. Relate how the application of ML can facilitate chemical research and development.

Conclusions

PSI4EDUCATION promises to develop students' programming skills as a tool for reinforcing chemical concepts and giving them new skills to solve chemical problems. Additionally, the exercises developed by this group lower the barrier to implementation since the educator does not have to develop these activities in isolation. Additionally, contributions from the education community are welcome in order to expand the footprint of PSI4EDUCATION. The described activities in this narrative highlight principle chemical modeling (bond breaking and Hartree-Fock) and emerging trends in chemical research (machine learning). However, these can be easily applied to organic chemistry (*21*), analytical chemistry (*6*), and even general chemistry (*16*). While the core team is continually adding to the database of exercises, the culture of the PSI4 community at large is one of inclusion, external contribution & development, and provision of open-source materials. Hence, PSI4EDUCATION is also a platform for development and distribution of educators' novel ideas for cutting-edge chemistry education. PSI4EDUCATION has evolved since its inception nearly a decade ago, and further developments are welcome with new ideas from educators with expertise across the chemical sciences.

Acknowledgments

The authors wish to acknowledge the other contributors to the PSI4EDUCATION effort. These include (but are not limited to): Dr. Tricia D. Shephard of the POGIL Project, Dr. Lori Burns of the Georgia Institute of Technology, Dr. Matthew Kennedy, Dr. Rollin King of Bethel University (MN), Dr. Beulah Narendrapurapu of Georgia Southern University, Dr. Rafael Quirino of Georgia Southern University, and Dr. Konrad Patkowski of Auburn University.

RCF wishes to acknowledge funding from the National Science Foundation through NSF grant OIA-1757220, from NASA through NASA grant NNX17AH15G, and start-up funds provided by the University of Mississippi. He would also like to acknowledge Jax D. Dallas of the University of Mississippi for useful insights provided in critiquing many assignments.

VHC was supported by a fellowship from The Molecular Sciences Software Institute under NSF grant OAC-1547580. He would also like to acknowledge the helpful comments provided by Dr. Lyudmila Slipchenko of Purdue University about the bond-breaking activity.

BGP acknowledges support from the US Department of Energy (DOE) Office of Science, Office of Basic Energy Sciences, Computational Chemical Sciences (CCS) Research Program under work proposal number AL-18-380-057, as well as the Graduate School Doctoral Assistantship Program from the Virginia Tech College of Science.

DAS wishes to gratefully acknowledge financial support from Bristol Myers Squibb, the U.S. National Science Foundation RUI Program (Grant No. CHE-1800014), the Donors of the American Chemical Society Petroleum Research Fund, the Floyd D. and Elisabeth S. Gottwald Endowment, and the University of Richmond.

References

1. Neiles, K. Y.; Mertz, P. S. In *Integrating Professional Skills into Undergraduate Chemistry Curricula*; 2020; pp 3–15.
2. Clark, R. W. The Structure of Chemistry. *J. Chem. Educ.* **1999**, *76*, 1612.
3. Cooper, M. The Case for Reform of the Undergraduate General Chemistry Curriculum. *J. Chem. Educ.* **2010**, *87*, 231–232.
4. Cooper, M.; Klymkowsky, M. Chemistry, Life, the Universe, and Everything: A New Approach to General Chemistry, and a Model for Curriculum Reform. *J. Chem. Educ.* **2013**, *90*, 1116–1122.
5. Schaller, C. P.; Graham, K. J.; Johnson, B. J.; Fazal, M. A.; Jones, T. N.; McIntee, E. J.; Jakubowski, H. V. Developing and Implementing a Reorganized Undergraduate Chemistry Curriculum Based on the Foundational Chemistry Topics of Structure, Reactivity, and Quantitation. *J. Chem. Educ.* **2014**, *91*, 321–328.
6. Pemberton, A. T.; Magers, D. B.; King, D. A. Integrated TGA, FTIR, and Computational Laboratory Experiment. *J. Chem. Educ.* **2019**, *96*, 132–136.
7. Rodríguez Ortega, P. G.; Montejo, M.; Valera, M. S.; López González, J. J. Studying the Effect of Temperature on the Formation of Hydrogen Bond Dimers: A FTIR and Computational Chemistry Lab for Undergraduate Students. *J. Chem. Educ.* **2019**, *96*, 1760–1766.
8. Perri, M. J. Online Data Generation in Quantitative Analysis: Excel Spreadsheets and an Online HPLC Simulator Using a Jupyter Notebook on the Chem Compute Web Site. *J. Chem. Educ.* **2020**, *97*, 2950–2954.
9. Arrabal-Campos, F. M.; Cortés-Villena, A.; Fernández, I. Building "My First NMRviewer": A Project Incorporating Coding and Programming Tasks in the Undergraduate Chemistry Curricula. *J. Chem. Educ.* **2017**, *94*, 1372–1376.
10. Bruck, L. B.; Towns, M.; Bretz, S. L. Faculty Perspectives of Undergraduate Chemistry Laboratory: Goals and Obstacles to Success. *J. Chem. Educ.* **2010**, *87*, 1416–1424.
11. Bretz, S. L.; Fay, M.; Bruck, L. B.; Towns, M. H. What Faculty Interviews Reveal about Meaningful Learning in the Undergraduate Chemistry Laboratory. *J. Chem. Educ.* **2013**, *90*, 281–288.
12. Talanquer, V. Chemistry Education: Ten Facets To Shape Us. *J. Chem. Educ.* **2013**, *90*, 832–838.
13. Levy, S. T.; Wilensky, U. Crossing Levels and Representations: The Connected Chemistry (CC1) Curriculum. *J. Sci. Educ. Technol.* **2009**, *18*, 224–242.
14. Levy, S. T.; Wilensky, U. Students' Learning with the Connected Chemistry (CC1) Curriculum: Navigating the Complexities of the Particulate World. *J. Sci. Educ. Technol.* **2009**, *18*, 243–254.
15. Jones, M. B. Molecular Modeling in the Undergraduate Chemistry Curriculum. *J. Chem. Educ.* **2001**, *78*, 867.
16. Fortenberry, R. C.; McDonald, A. R.; Shepherd, T. D.; Kennedy, M.; Sherrill, C. D. PSI4Education: Computational Chemistry Labs Using Free Software. In *The Promise of Chemical Education: Addressing our Students' Needs*; Daus, K., Rigsby, R., Eds.; American Chemical Society, Oxford University Press: Washington, DC, 2015; pp 85–98.

17. Teles dos Santos, M.; Vianna, A. S., Jr.; Le Roux, G. A. C. Programming Skills in the Industry 4.0: Are Chemical Engineering Students Able to Face New Problems? *Educ. Chem. Eng.* **2018**, *22*, 69–76.
18. McDonald, A. R.; Hagen, J. P. Beyond the Analytical Solution: Using Mathematical Software To Enhance Understanding of Physical Chemistry. In *Using Computational Methods to Teach Chemical Principles*; Grushow, A., Reeves, M. S., Eds.; American Chemical Society: Washington, DC, 2019.
19. Weiss, C. J. Scientific Computing for Chemists: An Undergraduate Course in Simulations, Data Processing, and Visualization. *J. Chem. Educ.* **2017**, *94*, 592–597.
20. Tan, S. W. B.; Naraharisetti, P. K.; Chin, S. K.; Lee, L. Y. Simple Visual-Aided Automated Titration Using the Python Programming Language. *J. Chem. Educ.* **2020**, *97*, 850–854.
21. Esselman, B. J.; Hill, N. J. Integrating Computational Chemistry into an Organic Chemistry Laboratory Curriculum Using WebMO. In *Using Computational Methods to Teach Chemical Principles*; Grushow, A., Reeves, M. S., Eds.; American Chemical Society: Washington, DC, 2019.
22. Wallace, A. M.; Curiac, C.; Delcamp, J. H.; Fortenberry, R. C. Accurate Determination of the Onset Wavelength (Λonset) in Optical Spectroscopy. *J. Quant. Spectrosc. Radiat. Transf.* **2021**, *265*, 107544.
23. Turney, J. M.; Simmonett, A. C.; Parrish, R. M.; Hohenstein, E. G.; Evangelista, F. A.; Fermann, J. T.; Mintz, B. J.; Burns, L. A.; Wilke, J. J.; Abrams, M. L.; Russ, N. J.; Leininger, M. L.; Janssen, C. L.; Seidl, E. T.; Allen, W. D.; Schaefer, H. F., III; King, R. A.; Valeev, E. F.; Sherrill, C. D.; Crawford, T. D. PSI4: An Open-Source $Ab\ Initio$ Electronic Structure Program. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2012**, *2*, 556–565.
24. Parrish, R. M.; Burns, L. A.; Smith, D. G. A.; Simmonett, A. C.; Deprince III, A. E.; Hohenstein, E. G.; Bozkaya, U.; Sokolov, A. Y.; Remigio, R. Di; Richard, R. M.; Gonthier, J. F.; James, A. M.; McAlexander, H. R.; Kumar, A.; Saitow, M.; Wang, X.; Pritchard, B. P.; Verma, P.; Schaefer III, H. F.; Patkowski, K.; King, R. A.; Valeev, E. F.; Evangelista, F. A.; Turney, J. M.; Crawford, T. D.; Sherrill, C. D. Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability. *J. Chem. Theory Comput.* **2017**, *13*, 3185–3197.
25. Smith, D. G. A.; Burns, L. A.; Sirianni, D. A.; Nascimento, D. R.; Kumar, A.; James, A. M.; Schriber, J. B.; Zhang, T.; Zhang, B.; Abbott, A. S.; Berquist, E. J.; Lechner, M. H.; Cunha, L. A.; Heide, A. G.; Waldrop, J. M.; Takeshita, T. Y.; Alenaizan, A.; Neuhauser, D.; King, R. A.; Simmonett, A. C.; Turney, J. M.; Schaefer, H. F.; Evangelista, F. A.; DePrince, A. E.; Crawford, T. D.; Patkowski, K.; Sherrill, C. D. Psi4NumPy: An Interactive Quantum Chemistry Programming Environment for Reference Implementations and Rapid Development. *J. Chem. Theory Comput.* **2018**, *14*, 3504–3511.
26. Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S.; Ivanov, P.; Avila, D.; Abdalla, S.; Willing, C.; development team, J. Jupyter Notebooks - a Publishing Format for Reproducible Computational Workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*; Loizides, F., Scmidt, B., Eds.; IOS Press, 2016; pp 87–90.
27. Schmidt, J. R.; Polik, W. F. *No Title*. 2020.

28. Krylov, A.; Windus, T. L.; Barnes, T.; Marin-Rimoldi, E.; Nash, J. A.; Pritchard, B.; Smith, D. G. A.; Altarawy, D.; Saxe, P.; Clementi, C.; et al. Perspective: Computational Chemistry Software and Its Advancement as Illustrated through Three Grand Challenge Cases for Molecular Science. *J. Chem. Phys.* **2018**, *149*, 180901.
29. Ringer McDonald, A.; Nash, J. Python Data and Scripting Workshop for Computational Molecular Scientists. *Github* 2020, https://github.com/MolSSI-Education/python_scripting_cms.
30. Perri, M. J.; Akinmurele, M.; Haynie, M. Chem Compute Science Gateway: An Online Computational Chemistry Tool. In *Using Computational Methods to Teach Chemical Principles*; Grushow, A., Reeves, M. S., Eds.; American Chemical Society: Washington, DC, 2019.
31. Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. & Eng.* **2007**, *9*, 90–95.
32. *Collaborative Data Science*. Plotly Technologies Inc.: Montreal, QC 2015.
33. Nash, J. Scientific Visualization Using Python. *Github* 2021, https://github.com/MolSSI-Education/python-visualization.
34. Szabo, A.; Ostlund, N. S. *Modern Quantum Chemistry*; Introduction to Advanced Electronic Structure Theory; Courier Corporation, 1996.
35. Chávez, V. H. Moly. Molecular Visualization in Jupyter. *Github* 2021, https://github.com/VHchavez/moly.
36. Holton, D.; Clarke, D. Scaffolding and Metacognition. *Int. J. Math. Educ. Sci. Technol.* **2006**, *37*, 127–143.
37. Tanner, K. D. Promoting Student Metacognition. *CBE Life Sci. Educ.* **2012**, *11*, 113–120.
38. Haghighatlari, M.; Hachmann, J. Advances of Machine Learning in Molecular Modeling and Simulation. *Curr. Opin. Chem. Eng.* **2019**, *23*, 51–57.
39. Elton, D. C.; Boukouvalas, Z.; Fuge, M. D.; Chung, P. W. Deep Learning for Molecular Design—a Review of the State of the Art. *Mol. Syst. Des. Eng.* **2019**, *4*, 828–849.
40. Noé, F.; Tkatchenko, A.; Müller, K.; Clementi, C. Machine Learning for Molecular Simulation. *Annu. Rev. Phys. Chem.* **2020**, *71*, 361–390.
41. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
42. Kammeraad, J. A.; Goetz, J.; Walker, E. A.; Tewari, A.; Zimmerman, P. M. What Does the Machine Learn? Knowledge Representations of Chemical Reactivity. *J. Chem. Inf. Model.* **2020**, *60*, 1290–1301.
43. Peyton, B. G.; Briggs, C.; D'Cunha, R.; Margraf, J. T.; Crawford, T. D. Machine-Learning Coupled Cluster Properties through a Density Tensor Representation. *J. Phys. Chem. A* **2020**, *124*, 4861–4871.
44. Rupp, M.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. A. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.* **2012**, *108*, 58301.
45. Rasmussen, C. E.; Williams, C. K. I. *Gaussian Processes for Machine Learning*; The MIT Press, 2006.
46. Murphy, K. P. *Machine Learning: A Probabilistic Perspective*; The MIT Press, 2012.

47. Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1: An Extensible Neural Network Potential with DFT Accuracy at Force Field Computational Cost. *Chem. Sci.* **2017**, *8*, 3192–3203.
48. Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1, A Data Set of 20 Million Calculated off-Equilibrium Conformations for Organic Molecules. *Sci. Data* **2017**, *4*, 170193.
49. Smith, D. G. A.; Altarawy, D.; Burns, L. A.; Welborn, M.; Naden, L. N.; Ward, L.; Ellis, S.; Crawford, T. D. The MolSSI QCArchive Project: An Open-Source Platform to Compute, Organize, and Share Quantum Chemistry Data. *ChemRxiv* 2020.