Check for updates

Received: 29 March 2020 | Revised: 18 May 2020 | Accepted: 29 May 2020

DOI: 10.1002/qua.26359

REVIEW

International Journal of Quantum Chemistry WILEY

Building capacity for undergraduate education and training in computational molecular science: A collaboration between the MERCURY consortium and the Molecular Sciences Software Institute

Ashley Ringer McDonald[1] | Jessica A. Nash[2] | Paul S. Nerenberg[3] | K. Aurelia Ball[4] | Olaseni Sode[5] | Jonathon J. Foley IV[6] | Theresa L. Windus[7] | T. Daniel Crawford[2,8]

[1]Department of Chemistry and Biochemistry, California Polytechnic State University, San Luis Obispo, California

[2]The Molecular Sciences Software Institute, Blacksburg, Virginia

[3]Departments of Physics and Astronomy and Biological Sciences, California State University, Los Angeles, Los Angeles, California

[4]Department of Chemistry, Skidmore College, Saratoga Springs, New York

[5]Department of Chemistry and Biochemistry, California State University, Los Angeles, Los Angeles, California

[6]Department of Chemistry, William Paterson University, Wayne, New Jersey

[7]Department of Chemistry, Iowa State University, Ames, Iowa

[8]Department of Chemistry, Virginia Tech, Blacksburg, Virginia

Correspondence
Ashley Ringer McDonald, Department of Chemistry and Biochemistry, California Polytechnic State University, 1 Grand Avenue, San Luis Obispo, CA 93407.
Email: armcdona@calpoly.edu

Funding information
ART Program at William Paterson University; National Science Foundation, Grant/Award Number: ACI-1547580; Research Corporation for Scientific Advancement, Grant/Award Number: Cottrell Scholar Award; University of California

Abstract

The Molecular Sciences Software Institute (MolSSI) is an National Science Foundation (NSF) funded institute that focuses on improving software, education, and training in the computational molecular sciences. Through a collaboration with the Molecular Education and Research Consortium in Undergraduate computational chemistRY (MERCURY), the MolSSI has developed resources for undergraduate and other early career students to lay an educational foundation for the next generation of computational molecular scientists. The resources focus on introducing best practices in software engineering to students from the very start to make their software more useable, maintainable, and reproducible.

KEYWORDS

Jupyter notebook, molecular mechanics education, programming education, python, quantum mechanics education

1 | INTRODUCTION

Computational molecular scientists worldwide have made innumerable, vital contributions to scientific discovery. Modern molecular dynamics (MD) and quantum chemical methods and models have achieved such a level of robustness and accuracy that they are often considered "computational experiments," in some cases with greater reliability than laboratory measurements, with direct impact on the design of more efficient

 © 2020 Wiley Periodicals LLC

combustion systems, the development of new chiral drugs, the elucidation of the detailed functionalities of biological macromolecules, the development of more advanced materials for energy storage, and more. These breakthroughs have been made possible by the evolution of hundreds of community software packages, some with lifetimes dating back to the earliest days of computing. These programs, which include both open-source and commercial packages, involve hundreds of thousands to millions of lines of hand-written code in a broad spectrum of languages, with underlying algorithms spanning the entire domain of computing motifs. The complexity of these software packages naturally reflects the intricacy of the scientific problems they were designed to solve, as well as the sophisticated computing hardware for which they were constructed. Their diverse code stacks thus require vast amounts of human time and thousands of programmers to develop and maintain—a daunting prospect for any new student wishing to join this growing field of science.

With the continuing advances in both computing hardware and software, the need for improved training of noncomputer science students in modern software engineering tools and best practices has become an imperative for our research domain. Tools such as programming languages that encourage high-level expressions and more readable code, standardized application programming interfaces (APIs), unit and regression testing, continuous integration, version control methods, automated generation of documentation, and more are the standards for advanced software development now required of our community. Unfortunately, such tools and techniques are seldom included in the curricula for chemistry and physics students who will ultimately be the ones responsible for building and maintaining these community codes.

Since its founding in 2001, the Molecular Education and Research Consortium in Undergraduate Computational chemistRY (the MERCURY consortium) has been dedicated to establishing research collaborations among faculty at primarily undergraduate institutions (PUIs) to provide high-quality undergraduate research experiences to students. The Molecular Sciences Software Institute was established in 2016 with support from the US National Science Foundation to serve and enhance the software development efforts of the broad field of computational molecular sciences (CMS).[1] In recent years, the MERCURY consortium and the Molecular Sciences Software Institute (MolSSI) have joined forces to provide numerous new educational opportunities for training the next generation of computational molecular scientists. The result has been a wealth of new resources for undergraduate students to enhance their ability to participate in undergraduate research by developing software skills.

The MolSSI was established as a collaborative project between multiple institutions with three major goals. The first is to provide software expertise and infrastructure support in collaboration with the CMS community to design, develop, test, deploy, and maintain key code infrastructure and frameworks. Part of this goal is accomplished through interactions with partners in academia, industry, supercomputing centers, and national laboratories to identify best practices, mitigate current software bottlenecks, set software priorities, and identify future workforce career paths. In addition, the MolSSI is developing new open-source software infrastructure to meet a variety of community needs. Examples here include the new and improved Basis Set Exchange[1],[2] the Community Code Database[2], the CookieCutter-CMS[3], the MolSSI Driver Interface[4], the MolSSI Integral Reference Project[5], QCArchive[6], and the Simulation Environment for Atomistic and Molecular Modeling[7].

The MolSSI's second goal is to provide engagement and leadership by helping the broad CMS community to establish its own standards for interoperability, best practices, and curation tools. In addition to holding many community workshops, the MolSSI is working with the community to establish standards for quantum chemistry data and for software best practices. In addition, the MolSSI is making larger efforts within the broad computational science community to develop best practices, such as the Better Scientific Software effort, to the benefit of other communities beyond CMS.

Finally, the MolSSI's third goal is to provide education and training to the CMS community on modern programming practices. In this area, the MolSSI organizes summer schools, workshops, and online resources[8] to encourage excellent programming skills for the next generation of CMS students. Software engineering classes are seldom required for scientists, and the MolSSI's education resources seek to fill that gap by teaching practical programming concepts and best practices that allow students to be more productive in their research.

The MolSSI's education program consists of cohorts of graduate student and postdoctoral Software Fellows, online training materials, and multiple in-person workshops at various locations each year around the United States. Education workshops are typically in-person events that teach programming concepts with examples that are relevant to CMS. The instruction style of in-person workshops is modeled after Software Carpentry's approach to teaching software best practices,[3] utilizing interactive, live coding to engage students with the concepts and material.

Since mid-2017, over 650 attendees have participated in the MolSSI's education workshops. While the MolSSI's initial educational program was primarily geared toward graduate and postgraduate students, in 2017, the MolSSI held its first undergraduate education workshop at the MERCURY Conference on Computational Chemistry. The MolSSI partnered, in the proposal stage, with MERCURY due to the concentration of undergraduates who were pursuing CMS research, as well as MERCURY's excellent record in attracting students from underrepresented groups. In addition, the connection with faculty from PUIs has been instructive in understanding the needs of the undergraduate CMS community. The MERCURY workshop has continued to be offered each year, along with expansion in the number and scope of workshops supported by the Institute.

Building on this success, in 2019, the MolSSI made a significant new investment in their undergraduate education and training program by hosting a MERCURY consortium faculty member on a sabbatical to work with the MolSSI education lead and focus on this type of programming. In this article, we will outline many of the resources the MolSSI has developed that benefit not only the faculty and students of the MERCURY consortium but the CMS community as a whole.

2 | UNDERGRADUATE EDUCATION AND TRAINING RESOURCES

The MolSSI education program is built around teaching students best practices in programming and software development to make CMSsoftware more useful, maintainable, and reproducible. In the materials for undergraduate or other early career students, we focus on teaching students to use programming best practices from the beginning—introducing concepts such as writing documentation, version control, and testing the very first time students are learning to program.

The resources outlined here begin with an introduction to programming concepts in the Python Data and Scripting Workshop and the Data Analysis using Python modules, followed by applications of Python programming to perform computational chemistry calculations in resources such as the Quantum Mechanics (QM) Tools workshop. All of MolSSI's educational materials are available online through websites hosted on GitHub. Both students and instructors can access the workshops to work at their own pace. Each of the resources can be taught as a stand-alone workshop; the prerequisite skills students need to learn the material (and, in some cases, resources that might help them develop those prerequisite skills) are delineated below. Some particularly important best practices, such as version control, are included in more than one workshop.

For all activities in all modules presented, complete code solutions are provided in the online materials. For many workshops, there is supplemental information available, such as lecture slides or Jupyter notebooks, which can be found in the *files* folder of the GitHub repository for the activity. The GitHub repositories are all linked from the MolSSI Education Resources page[9].

While the workshops were all designed with the goal to enhance undergraduate students' ability to participate in undergraduate research experiences, they may also have utility in the chemistry curriculum. In such cases, the Python Data and Scripting workshop would be foundational and could help students develop the skills they need for the subsequent material.

2.1 | Python data and scripting workshop

The Python Data and Scripting workshop provides an overview of using the Jupyter notebook, basic Python syntax, file parsing, data analysis, and graphing. It also introduces some best practices in software development, including using version control and writing documentation. The intended audience for this workshop is undergraduate or other early career students who are involved in, or are about to begin, research in CMS. The workshop assumes no prerequisite knowledge other than using the bash terminal and understanding filepath structure. This workshop was primarily developed by Ashley Ringer McDonald and Jessica A. Nash.

The workshop is divided into nine modules, each of which includes short formative assessment questions and longer summative assessments through programming projects. Module 1 introduces basic Python programming syntax, lists, control structures such as for loops and logic, data types, and a few base Python functions. Chemistry-relevant examples, such as converting from joules to calories, are used throughout. Module 2 applies these skills to a real situation students could encounter in CMS research—parsing the output file of a quantum chemistry calculation. Through this module, students learn to work with filepaths, read in a file, search its contents line by line for a particular string, parse this information, manipulate strings and change data types, and save information to new files. Module 3 introduces Python imports. Students import the *glob* library[10] to analyze multiple files at once. The use of control structures such as for loops and logic are reinforced throughout these lessons.

Module 4 introduces the *NumPy* library[4,5] and how students can use it to analyze tabular data that they might have previously analyzed through spreadsheet programs. In Module 4, a new dataset, results from a MD simulation, is introduced. This module concludes with a summative assessment programming project that requires students to combine skills learned in the file-parsing modules and the tabular data module to calculate the distances between atoms in an xyz file. The project offers several optional extensions to challenge students if they master the initial project. When taught as an in-person workshop, this task is given as homework after the first day of instruction. As with all assessments in the modules, complete code solutions are provided in the online materials. Module 5 introduces the *Matplotlib* library and has students plot the data from the MD dataset. This module also focuses on making high-quality graphics that could be used in research presentations and publications.

In Module 6, the focus of the workshop turns to best practices for software development. Module 6 teaches students to write functions and immediately introduces writing documentation for functions using Python docstrings and assigning default values in functions. Students refactor the code they wrote in the geometry analysis project (the summative assessment project of Module 4) and write documentation for each function. In Module 7, students learn to move code from the Jupyter notebook to command line-executable scripts. Module 8 focuses on the importance of code testing. The *pytest* framework[11] is introduced to write and run unit tests. Students write unit tests for their geometry analysis code and think about the relevant edge and corner cases for testing. The final module focuses on sharing code and code collaboration. Students learn to use the *git* version of control software, establish a new repository controlled by *git*, and ultimately share their code on GitHub. An optional extension to Module 9 teaches a GitHub collaborative workflow, discussing forks, branches, pull requests, and merges and merge conflicts.

The Python Data and Scripting workshop is the flagship training of MolSSI's undergraduate education program; it is taught through in-person workshops and webinars and is available online as self-paced learning modules. The workshop can be accessed online at //education.molssi.org/python_scripting_cms/. In 2019, the workshop was offered in a variety of settings, including regional ACS meetings, regional theoretical chemistry

conferences, a workshop program with the Tapia Center for Equity and Excellence at Rice University, and through existing summer research programs.

2.2 | Data analysis using Python

The Data Analysis using Python workshop provides additional modules that can be used to supplement the Python Data and Scripting workshop to help students develop more advanced skills. This workshop was primarily developed by Jessica A. Nash. The workshop includes a module to provide a more in-depth look at *NumPy* arrays and highlights particular advantages that they have over lists. Element-wise operations, broadcasting, and NumPy array axes are introduced. The students are challenged to return to their geometry analysis project from the Python Data and Scripting workshop and rewrite their code using the new concepts they have learned. A second module introduces the students to the popular Python data science library, *pandas*. This module introduces the concept of a data frame and discusses different forms of indexing and lookup that are available in pandas. The module uses a free periodic table dataset for its examples and activities. These lessons are available online at //education.molssi.org/python-data-analysis/.

2.3 | QM tools

The QM tools workshop introduces the basics of running and analyzing electronic structure calculations using the freely available electronic structure package Psi4[6] in concert with Python. It also introduces version control and using GitHub as a repository for software. The intended audience for this workshop is undergraduate or other early career students who are currently engaged in CMSresearch or may begin working in a CMS research lab. This workshop assumes that students have taken introductory-level physics and chemistry courses or have equivalent knowledge. It also has a prerequisite that students have some familiarity with Python programming, equivalent to what would be learned in the MolSSI Python Data and Scripting workshop. The workshop content was primarily developed by Paul Nerenberg, Aurelia Ball, Olaseni Sode, Lee-Ping Wang, Theresa Windus, and Ashley Ringer McDonald.

The workshop begins with a 1-hour lecture that provides a high-level overview of QM, the purpose of electronic structure calculations, and an introduction to common electronic structure methods and concepts. This lecture helps set the stage for all the modules that come after it during the workshop by providing students who may never have been exposed to these ideas with a common intellectual framework.

In Module 1, students are introduced to the concept of geometry optimization. This is typically the first step in any computational analysis of a molecular system and is thus a natural first activity. The objectives for this module are to generate the geometry of a small molecule, optimize this geometry using the PsiAPI of Psi4[6] within the Jupyter notebook workflow, and plot the bond length and energy changes during the optimization. The benzene molecule is used as the first example for the students. More advanced students are encouraged to use the Avogadro[7] software package to generate the initial geometry structure, while less advanced students had access to a previously generated benzene structure. The module walks the students through the setup of the calculation, including how to specify the number of iterations for the self-consistent field (SCF) procedure and number of geometry optimization steps to perform. Once the students have optimized the molecule's geometry, the module details how to read bond distances and energies from each of the optimization steps and plots these in the notebook to showcase the convergence of the procedure. In the summative assessment project for this module, students perform a geometry optimization and analysis for nitrobenzene.

Module 2 uses the optimized nitrobenzene structure from Module 1 and is aimed at calculating the potential energy surface along an intramolecular coordinate of the molecule. The objectives for this module are to identify dihedral angles for particular intramolecular coordinates, systemically increase the values of these coordinates during a series of geometry optimizations, and finally plot the energy changes as a function of the dihedral angles. In the module, students are first introduced to the internal coordinate system with the Avogadro software package by identifying bond distances, bond angles, and dihedral angles. Next, the students are instructed on how to set up the nitrobenzene molecule for subsequent calculations with Psi4 and how to generate a list of dihedral angles using the Python programming language. With this set of angles, the module instructs students to set up a succession of constrained optimizations at the Hartree-Fock/3-21G level of theory and basis set. Once all the optimizations have been completed, the minimum energies at each of the constrained geometries are plotted to reveal the rotational barrier of the nitro group around the benzene plane.

Module 3 progresses from thinking about intramolecular coordinates to intermolecular coordinates and interactions between molecules. This module focuses on scanning the intermolecular potential energy surface of a water dimer. The purpose of this exercise is to be able to use the z-matrix format to specify and adjust the geometry parameter and visualize the dimer interaction energy using one-dimensional and two-dimensional plots. The students are instructed to specify the water dimer geometry using the internal coordinate system. Next, students create a list of variables for the intermolecular oxygen-oxygen distance starting from 1.5 to 3.0 Å. The energy of the dimer system is calculated at each distance. The computed energies are then plotted to show the one-dimensional potential energy surface (PES). Next, students

perform a scan of the two-dimensional (2D) water dimer surface along the intermolecular oxygen-oxygen distance and a rotation around one of the water molecules. The students then generate different 2D and three-dimensional (3D) plots to study the interaction energy as a function of the angle.

In Module 4, students learn how the computed properties of molecules converge as the basis set size increases. The students perform geometry optimizations and frequency analysis on carbon dioxide, an important greenhouse gas. The students use the Hartree-Fock level of theory with three different basis set sizes (cc-pVDZ, cc-pVTZ, and cc-pVQZ) and plot the vibrational frequencies vs increasing basis set size to observe the convergence of these frequencies. See representative plot in Figure 1. The summative assessment of this module is to perform the same analysis of the carbon dioxide vibrational frequencies but using a post-Hartree-Fock method.

The goal of Module 5 is to compute the standard redox potential for a chemical reaction. This module begins with a short lecture about the importance of redox reactions in chemistry and the environment, as well as an overview of how redox potentials are determined theoretically. The students are guided through the procedure of computing the redox potential for the neutral nitrobenzene and the nitrobenzene anion. For each compound, the students compute the Gibbs energy using implicit solvent by first optimizing each molecular geometry and then performing a vibrational frequency calculation to determine the enthalpic and entropic contributions to the free energy. Using the Gibbs energy difference between the two compounds, students are then asked to calculate the redox potential using the Nernst equation.

Module 6 provides an overview of version control using git and GitHub. The students learn to fork a repository on GitHub and how to clone that repository to their local machine. Next, students learn how to use git status, git add, and git commit to make changes to a local repository and push their changes back up to their GitHub repository. Finally, students create a pull request to an upstream GitHub repository and learn how to address merge conflicts that arise when multiple people merge changes.

Module 7 is a summative assessment activity of the entire QM tools workshop in which students are asked to synthesize concepts from throughout the workshop. This module involves several steps: forking an existing GitHub repo that contains a "skeleton" Psi4 Python script for performing a vibrational frequency analysis of methane, cloning this forked repo to one's local computer, adding code to implement "new" features into the Python script such as extracting the nonzero vibrational frequencies and calculating their degeneracies, committing the changes to one's personal GitHub repo, and finally creating a pull request for the new code to be added back to the original repo. The "adding code" portion of this module is inquiry-based and requires that students learn about Python dictionaries and explore the *NumPy* documentation on their own in order to complete the necessary code.

The QM Tools workshop was taught at the MERCURY Conference on Computational Chemistry at Furman University in 2019 and will be offered again in this conference in July 2021. The workshop materials can be accessed online at: //education.molssi.org/qm-tools/

2.4 | Ab initio MD

The ab initio MD workshop provides an introduction to the theory and applications of MD simulations and also introduces students to the computation of interatomic potentials derived from ab initio calculations. This workshop emphasizes the fundamental principles of these tools through hands-on coding exercises that culminate in the creation of a simple Python MD program that simulates the vibrational motion of a diatomic molecule by solving classical equations of motion with forces that derive from an ab initio potential energy surface. Foundational practices in CMS, including dimensional analysis and validation against exact/analytical models, are integrated into the development of the MD code. The workshop is primarily intended for undergraduates who are interested in CMS but may also be useful to early career graduate students in CMS. The workshop assumes prerequisite knowledge of single-variable calculus and that students have taken introductory-level physics and chemistry courses

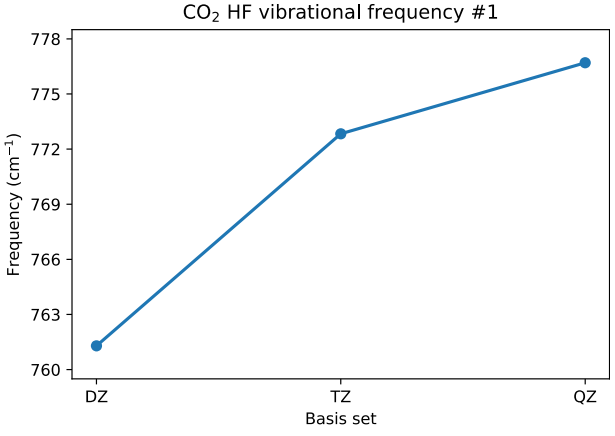


FIGURE 1 Convergence of calculated vibrational frequency for the bending mode of CO_2 with increasing basis set size (cc-pVDZ to cc-pVQZ) using the Hartree-Fock method

or have equivalent knowledge. Students need familiarity with Python and the Jupyter notebook, equivalent to what would be learned in the Python Data and Scripting workshop. More advanced knowledge of numerical methods, physical chemistry, classical mechanics, and computer programming may be helpful but are not required. The workshop content was primarily developed by Jonathan J. Foley.

The workshop begins by teaching students to compute and validate an ab initio potential for the diatomic molecule HF. To facilitate computation of the ab initio potential energy surface, the Psi4NumPy package[8] is introduced. The Psi4NumPy package permits the use of the Psi4 quantum chemistry package with simple Python syntax for input specification, execution, and data analysis. In particular, Psi4NumPy is utilized to compute the energy as a function of bond length at three different levels of theory (RHF/cc-pVTZ, MP2/cc-pVTZ, and CCSD/cc-pVTZ).[9] It is possible to compute the energy for ~25 different bond lengths at all three levels of theory in a matter of a few minutes. Students use *Matplotlib* to visualize and compare the three different potential energy curves.

Students are then introduced to the concept of spline interpolation, as well as the spline-fitting tools of the Python package *SciPy*.[10] Students utilize SciPy's interpolation capabilities to fit a cubic spline to their ab initio data and locate the minimum of this spline in order to estimate the equilibrium bond length of each level of theory. Computing the equilibrium bond length provides an opportunity for validation through comparison with experimentally determined values through, for example, the National Institute of Standards and Technology (NIST) database.[11]

SciPy's built-in spline differentiation capabilities are then utilized to compute a force spline (from the negative of the first derivative of the potential spline), as well as a curvature spline (from the second derivative of the potential spline). The importance of this force spline to the MD program is highlighted. The curvature splines and the computed equilibrium bond lengths are used to estimate the harmonic force constant and vibrational frequency of HF at each level of theory, which are also compared with experimental values available in the NIST database. The relationship between the slope of the potential energy curve and the force is used to introduce the concept of dimensional analysis. This module concludes by using the force constant to compute a harmonic potential and plotting this against the ab initio potential energy curve, showing that the two agree close to the equilibrium bond length (see representative plot in Figure 2).

In the next module, students implement and validate a velocity Verlet algorithm[12] and use it to simulate the vibrational motion of their diatomic molecule. The working equations of the velocity Verlet algorithm are provided within the Jupyter notebook, but students are asked to apply their skills to write a custom function that takes as input arguments the force spline, the particle mass, the current position, the current velocity, and a time step and that returns as outputs the updated position and velocity. The equations of this algorithm present an additional opportunity to apply dimensional analysis to verify if all terms in the position update equation have the dimensions of length and that all terms in the velocity update equation have dimensions of length over time. Once the velocity Verlet algorithm is implemented, the approximate nature of this numerical method for solving Newton's equations of motion are discussed, and the importance of validating against an analytical solution is emphasized. Students are then shown the exact solution to Newton's equation of motion for a harmonic potential and asked to implement a function that takes as input arguments regarding the harmonic frequency, equilibrium position, amplitude, phase, and the value of the time variable and that outputs the position variable. The amplitude and phase values are provided by the instructor and are chosen to be consistent with the initial conditions that will be used for velocity Verlet integration with both harmonic and ab initio potentials. These two custom functions are then used to generate trajectories of the position vs time for vibrational motion subject to a harmonic potential, which are plotted using *Matplotlib*. The students see that the plots from the velocity Verlet algorithm and the analytical solution are indiscernible. Once validated, students rerun trajectories using force splines derived from their ab initio potentials, create plots of position vs time (and position vs momentum, optionally), and identify at least two key ways in which these trajectories differ from the harmonic case (see Figure 3).

For the summative assessment for the ab initio MD workshop, the students are divided into small groups and choose another diatomic molecule to extend their ab initio *MD* code; each group gives a short presentation on their code. This provides a great opportunity for students to examine the limitations of the current implementation and try to find simple ways to overcome those limitations. Course instructors may wish to provide a few concrete suggestions for these extensions; suggestions might include simulating an ionized or radical diatomic molecule, improving

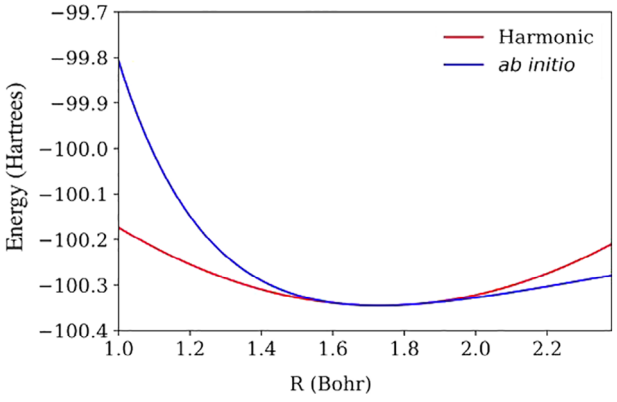


FIGURE 2 Plot of the ab initio potential computed at the CCSD/cc-pVTZ level of theory (blue curve) against a harmonic potential (red curve) fit to force constant and equilibrium bond length from the CCSD/cc-pVTZ potential

the level of theory used to generate the PES, considering the level of theory appropriate for an open-shell molecule, and modeling vibrational motion in the condensed phase by augmenting the equations of motion with fluctuating forces and dissipation (which also requires modification to the velocity Verlet algorithm).[13]

The ab initio MD workshop was presented as part of a week-long workshop on CMS cosponsored by the MolSSI and the Richard Tapia Center for Excellence and Equity at Rice University in June 2019, with plans to hold the workshop annually. The workshop can be accessed online at http://education.molssi.org/ab-initio-md/.

2.5 | Molecular mechanics tools

The Molecular Mechanics Tools workshop introduces the basics of running and analyzing MD simulations using the freely available OpenMM[14] and MDTraj[15] libraries for Python. It also introduces version control and using GitHub as a repository for software. As with the QM Tools workshop, the intended audience for this workshop is undergraduate or other early career students who are currently engaged in CMS research or may begin working in a CMS research lab. It likewise assumes students have taken introductory-level physics and chemistry and have some familiarity with Python programming, equivalent to what would be learned in the Python Data and Scripting workshop. The workshop was primarily developed by Paul Nerenberg, Lee-Ping Wang, and Theresa Windus.

The workshop consists of five modules interspersed with short lectures on molecular mechanics concepts. The first three modules involve guided exercises where students must make slight modifications based on a detailed example, the fourth module covers using git and GitHub, and the fifth module is a challenging summative assessment project where students bring together many of the ideas from the workshop.

The beginning of the workshop focuses on understanding what molecular mechanics force fields are and how one can create or modify force field parameters for the purpose of MD simulations. First, a lecture introduces the fundamentals of molecular mechanics, focusing on force fields: what they include, how they are developed, and their limitations. Then, Module 1 focuses on understanding an OpenMM (XML format) parameter and topology file for ethane and asks students to complete a similar file for butane using a text editor. Students must figure out what parts of the parameter file they have to modify and why.

Next, students begin using Jupyter notebooks to learn how to run and interpret the results of basic MD simulations. A brief lecture introduces how MD simulations work, what you need to run them, and some of the challenges of MD. In Module 2, the students perform MD simulations of alkanes in a Jupyter notebook using *OpenMM*. Module 2 builds off of Module 1 by using the force field files for ethane and butane created in that earlier module. Students are provided with a complete notebook that performs an energy minimization and MD simulation of ethane. Then, they are asked to create a copy of this notebook and modify it to run a simulation of butane, changing some of the parameters of the simulation.

In the second part of Module 2, students analyze the alkane simulation trajectories using *MDTraj* and visualizing them with *NGLView*.[16] Again, students are provided with a complete notebook for ethane and are asked to create their own notebook for the butane trajectory analysis. Students examine torsion angles, bond angles, and bond lengths, making histograms and potential of mean force plots for each observable.

The next section of the workshop focuses on how to run and analyze a simulation of a protein. Module 3 begins with running a simulation of the protein BPTI based on a provided Jupyter notebook, reproducing the simulation published in a 1977 *Nature* article by Andy McCammon, Bruce Gelin, and Martin Karplus.[17] The student contribution is to measure particular bond lengths, bond angles, and dihedral angles to reproduce a figure from that research article (see a representative plot in Figure 4). In this module, students also have to figure out how to convert units and reformat their plots of the dihedral angles in order to make them easier to read.

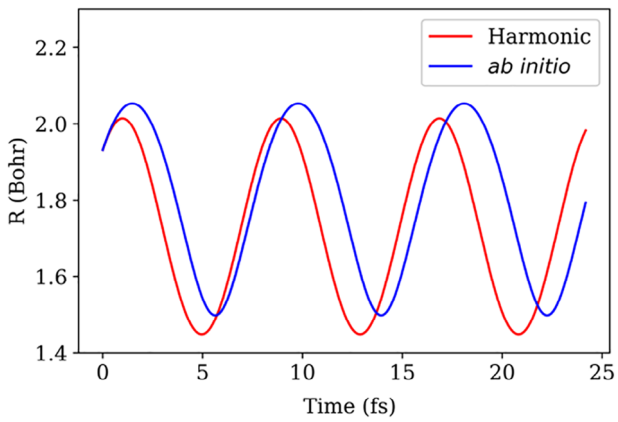


FIGURE 3 Trajectory of the HF bond length vs time subject to the ab initio potential (blue curve) compared to the bond length trajectory subject to a harmonic potential (red curve), where the ab initio and harmonic potentials that drive these dynamics are shown in Figure 2

Module 4 once again covers version control using git and GitHub, which is identical to the analogous module in the QM Tools workshop mentioned above. The workshop concludes with a summative assessment module, in which students are asked to synthesize concepts from throughout the workshop, again following the same basic format as the QM Tools workshop. The final module is once again a summative assessment of the entire workshop. Module 5 involves several steps: forking an existing GitHub repo that contains an OpenMM Python script for performing an MD simulation of the Trp-cage mini protein, cloning this forked repo to one's local computer, implementing one of two possible "new" features into the Python script, committing the changes to one's personal GitHub repo, and finally creating a pull request for the new feature to be added back to the original repo. Together, these steps comprise a challenging integrative exercise that touches on all the elements of the workshop and helps students solidify their knowledge.

The Molecular Mechanics Tools workshop has been taught twice at workshops associated with the MERCURY Conference on Computational Chemistry, during the summers of 2017 and 2018. The workshop materials can be accessed online at //education.molssi.org/mm-tools/.

3 | SUMMARY AND PROSPECTUS

The MolSSI offers many additional educational resources, all of which are continually updated and maintained at http://education.molssi.org/resources.html. Of particular interest to those that do research with undergraduate students in CMS, the Getting Started in Computational Chemistry page has a curated list of tutorials for common computational skills that students need to start research in computational chemistry, such as use of the terminal, text editors, and remote computing resources. For more advanced students, the Best Practices workshop teaches them to build a Python package using the MolSSI CookieCutter, more advanced testing and documentation concepts, continuous integration, and hosting the project on GitHub. Other more advanced resources include workshops on parallel computing, object-oriented programming, and design patterns.

The MolSSI also offers education opportunities for graduate students, postdocs, and faculty. Graduate students and postdocs who work at institutions in the United States are eligible to apply for the MolSSI Software Fellowship program, offering financial support, mentoring, and training to individuals working on software development projects of broad importance to our field. Furthermore, through its community-led workshop program, the MolSSI also exposes academic faculty, staff members of national laboratories, and computational scientists working in industry to training in the best practices for software development, thus reaching a broader swatch of practicing computational molecular scientists.

While all of the resources presented in this article were developed with the goal of enhancing students' ability to participate in undergraduate research in CMS, they may have additional applicability in other curricular contexts. As more institutions emphasize programming throughout the chemistry curriculum, many chemistry educators are developing domain- and discipline-specific lessons that rely on foundational knowledge of Python and the use of Jupyter notebooks. The resources offered by the MolSSI could be used to provide fundamental skills such that the instructor does not have to develop such lessons on their own. Furthermore, while these resources use the context of CMS—electronic structure, MD, and molecular materials—they could easily be adapted to other scientific domains, including, for example, high-energy physics, biology, and even climate science.

All of the MolSSI's educational resources focus on introducing accepted best practices in software engineering right from the beginning such that students make continuous use thereof. The best practices for software engineering that the MolSSI teaches are the same ones the MolSSI Software Scientist team uses in its own software infrastructure development. These practices are vital skills for students who go on to graduate school in CMS but are also sought after for many other science careers and technical industries.

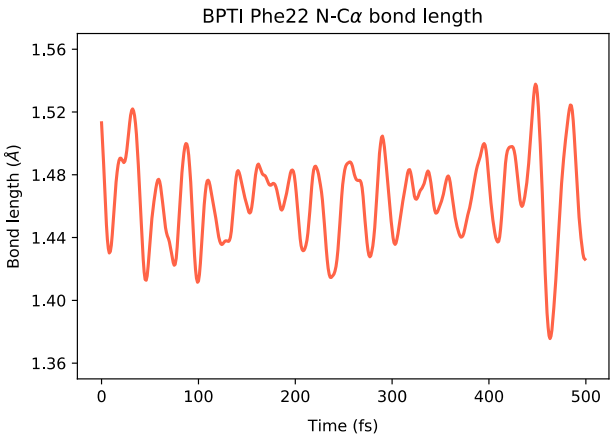


FIGURE 4 Time series of the N-Cα bond length in Phe22 during an MD simulation of BPTI, bearing close resemblance to the top panel of Figure 3A in Ref.[15]

The MolSSI education program is critical for the outreach of the Institute and for CMS in general. By providing education and training resources, the MolSSI aims to develop a consensus within the CMS community around standards for best practices for code, data sharing, and software development. These resources can help lay an educational foundation for the next generation of computational molecular scientists who will be developing and maintaining CMS software for many years to come. Internal surveying, conducted at each MolSSI workshop, suggests that these early career resources are very important to reach a wide range of students and actually attract a more diverse audience than programming aimed at graduate students and postdocs. By offering online and early career resources, and through collaborations with other organizations such as the MERCURY consortium, we are able to reach a broader and more diverse coalition of students to be the future of our field.

ACKNOWLEDGMENTS

The authors thank Lee-Ping Wang (University of California Davis) for his contributions to the Molecular Mechanics tools workshop and for serving as a workshop instructor at the MERCURY Conference on Computational Chemistry in 2017 and 2019. The Molecular Sciences Software Institute is funded by the National Science Foundation under grant ACI-1547580. Jonathon J. Foley acknowledges the Research Corporation for Scientific Advancement Cottrell Scholar Award and the ART program at William Paterson University for partial support of this work.

AUTHOR CONTRIBUTIONS

Ashley Ringer McDonald: Conceptualization; investigation; writing-original draft; writing-review and editing. **Jessica A. Nash:** Conceptualization; investigation; writing-original draft. **Paul S. Nerenberg:** Investigation; writing-original draft. **K. Aurelia Ball:** Investigation; writing-original draft. **Olaseni Sode:** Investigation; writing-original draft. **Jonathon J. Foley IV:** Investigation; writing-original draft. **Theresa L. Windus:** Funding acquisition; investigation; writing-original draft. **T. Daniel Crawford:** Funding acquisition; writing-original draft; writing-review and editing.

ORCID

Ashley Ringer McDonald https://orcid.org/0000-0002-4381-1239
Jessica A. Nash https://orcid.org/0000-0003-1967-5094

END NOTES

[1] http://www.basissetexchange.org/
[2] http://molssi.org/software-search/
[3] http://github.com/molssi/cookiecutter-cms
[4] http://github.com/MolSSI-MDI/MDI_Library
[5] http://github.com/MolSSI/MIRP
[6] http://qcarchive.molssi.org/
[7] http://github.com/molssi-seamm
[8] http://education.molssi.org/resources.html
[9] http://education.molssi.org/resources.html
[10] http://docs.python.org/3/library/glob.html
[11] http://github.com/pytest-dev/pytest

REFERENCES

[1] A. Krylov, T. L. Windus, T. Barnes, E. Marin-Rimoldi, J. A. Nash, B. Pritchard, D. G. A. Smith, D. Altarawy, P. Saxe, C. Clementi, T. D. Crawford, R. J. Harrison, S. Jha, V. S. Pande, T. Head-Gordon, *J. Chem. Phys.* **2018**, *149*, 180901.
[2] B. P. Pritchard, D. Altarawy, B. Didier, T. D. Gibson, T. L. Windus, *J. Chem. Inf. Model.* **2019**, *59*, 4814.
[3] http://software-carpentry.org.
[4] T. E. Oliphant, *A Guide to NumPy*, Trelgol Publishing, Austin, TX **2006**.
[5] S. van der Walt, S. C. Colbert, G. Varoquaux, *Comput. Sci. Eng.* **2011**, *13*, 22.
[6] R. M. Parrish, L. A. Burns, D. G. A. Smith, A. C. Simmonett, A. E. DePrince 3rd., E. G. Hohenstein, U. Bozkaya, A. Y. Sokolov, R. Di Remigio, R. M. Richard, et al., *J. Chem. Theory Comput.* **2017**, *13*, 3185.
[7] M. D. Hanwell, D. E. Curtis, D. C. Lonie, T. Vandermeerschd, E. Zurek, G. R. Hutchison, *J. Cheminform.* **2012**, *4*, 17.
[8] D. G. A. Smith, L. A. Burns, D. A. Sirianni, D. R. Nascimento, A. Kumar, A. M. James, J. B. Schriber, T. Zhang, B. Zhang, A. S. Abbott, E. J. Berquist, M. H. Lechner, L. A. Cunha, A. G. Heide, J. M. Waldrop, T. Y. Takeshita, A. Alenaizan, D. Neuhauser, R. A. King, A. C. Simmonett, J. M. Turney, H. F. Schaefer, F. A. Evangelista, A. E. DePrince III., T. D. Crawford, K. Patkowski, C. D. Sherrill, *J. Chem. Theory Comput.* **2018**, *14*, 3504.
[9] C. J. Cramer, *Essentials of Computational Chemistry: Theories and Models*, 2nd ed., Wiley, West Sussex, England **2004**.
[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., *Nat. Methods* **2020**, *17*, 261.
[11] Lovas, F. J.; Tiemann, E. Diatomic Spectra Database. //www.nist.gov/pml/diatomic-spectral-database (accessed March 25, 2020).
[12] W. C. Swope, H. C. Andersen, P. H. Berens, K. R. Wilson, *J. Chem. Phys.* **1982**, *76*, 637.

[13] A. Brünger, C. L. Brooks, M. Karplus, *Chem. Phys. Lett*. **1984**, *105*, 495.
[14] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L. P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, V. S. Pande, *PLoS Comput. Biol.* **2017**, *13*, e1005659.
[15] R. T. McGibbon, K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L. P. Wang, T. J. Lane, V. S. Pande, *Biophys. J.* **2015**, *109*, 1528.
[16] H. Nguyen, D. A. Case, A. S. Rose, *Bioinformatics* **2018**, *34*, (7), 1241–1242. http://dx.doi.org/10.1093/bioinformatics/btx789.
[17] J. A. McCammon, B. R. Gelin, M. Karplus, *Nature* **1977**, *267*, 585.

AUTHOR BIOGRAPHIES

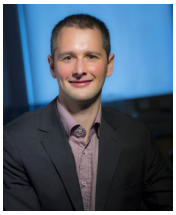
Ashley Ringer McDonald is an associate professor of chemistry at Cal Poly San Luis Obispo. She received her bachelor's degree from Mississippi College and her Ph.D. from the Georgia Institute of Technology. Her research focuses on using multiscale modeling to study molecular interactions in complex chemical contexts. She is an Associate of the Molecular Sciences Software Institute (MolSSI) and a member of the MERCURY Consortium. She also leads Psi4Education, the education and outreach program of the quantum chemistry software package Psi4.

Jessica Nash is a Software Scientist and the Education Lead at The Molecular Sciences Software Institute (MolSSI). Her love of coding and simulations began when she worked in the group of Prof. Michael as an undergraduate chemistry major at UNC Chapel Hill. She completed her PhD at North Carolina State University in Materials Science and Engineering under the direction of Prof. Yaroslava G. Yingling and Prof. Thomas H. LaBean, where she studied DNA nanomaterials using molecular dynamics simulations. Since joining MolSSI in 2017, her work has focused on the development and improvement of software in the computational molecular sciences. As Education Lead for the Institute, she develops educational materials for researchers which enhance their capabilities to write code and use computational molecular science software.

Paul Nerenberg is an assistant professor in the Departments of Physics & Astronomy and Biological Sciences at Cal State LA. In addition to his research interests involving the application and development of molecular simulation methods in biophysical contexts, he is an avid proponent of interdisciplinary educational efforts in scientific computing and data analysis.

K. Aurelia Ball earned a PhD in Biophysics from the University of California at Berkeley in 2013. She then started a postdoctoral research position at University of California at San Francisco and became an NIH National Research Service Award Postdoctoral Fellow in 2015. Prof. Ball started her position as an Assistant Professor of Chemistry at Skidmore College in 2016. She has been MolSSI Associate since 2019.

Olaseni Sode is an assistant professor of chemistry at Cal State LA. He received his bachelor's degree from Morehouse College and his Ph.D. from the University of Illinois at Urbana-Champaign. Prof. Sode's research focuses on using computational approaches, including electronic structure theory, molecular dynamics simulations, and machine learning to study multiscale chemical phenomena.



Jonathan J. Foley IV is an a Assistant Professor of Chemistry at William Paterson University. He received his PhD in Physical Chemistry in 2012 from the University of Chicago working on reduced density matrix mechanics. He was a postdoctoral fellow from 2012-2015 at the Center for Nanoscale Materials, Argonne National Laboratory, where he worked on theory and modeling of nanophotonics and plasmonics. Prof. Foley's research group develops and deploys multi-scale theoretical methods to study light-matter interactions.

Theresa L. Windus is a Distinguished Professor of Chemistry at Iowa State University (ISU), an Associate with Ames Laboratory, an ISU Liberal Arts and Sciences Dean's Professor, and Deputy Director for the Molecular Sciences Software Institute. Theresa received her B.A. degrees in chemistry, mathematics and computer science from Minot State University. She then completed her Ph.D. in physical chemistry at Iowa State University where she focused on developing high performance algorithms. Theresa examined relativistic effects and developed novel tensor methods as a postdoctoral researcher. Theresa develops new methods and algorithms for high performance computational chemistry as well as applying those techniques to both basic and applied research.

T. Daniel Crawford is University Distinguished Professor of Chemistry at Virginia Tech and the Director of the Molecular Sciences Software Institute in Blacksburg, Virginia. He received his bachelor's degree in 1992 from Duke University and his PhD in 1996 from the University of Georgia. His research focuses on quantum chemical models of molecular properties. He is a Fellow of the American Chemical Society and the winner of 2010 Dirac Medal of the World Association of Theoretical and Computational Chemists.

How to cite this article: McDonald AR, Nash JA, Nerenberg PS, et al. Building capacity for undergraduate education and training in computational molecular science: A collaboration between the MERCURY consortium and the Molecular Sciences Software Institute. *Int J Quantum Chem*. 2020;120:e26359. //doi.org/10.1002/qua.26359