JOURNAL OF CHEMICAL EDUCATION

Technology Report

pubs.acs.org/jchemeduc

A Python Program for Solving Schrödinger's Equation in Undergraduate Physical Chemistry

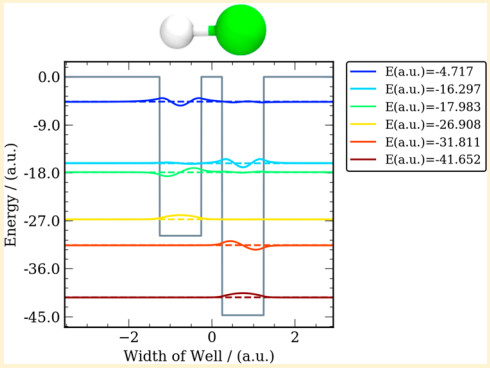
Matthew N. Srnec,*[,†] Shiv Upadhyay,[‡] and Jeffry D. Madura[‡,§]

[†]Department of Chemistry, Physics, and Engineering; Franciscan University, Steubenville, Ohio 43952 United States

[‡]Department of Chemistry and Biochemistry, Center for Computational Sciences; Duquesne University, Pittsburgh, Pennsylvania 15282 United States

Supporting Information

ABSTRACT: In undergraduate physical chemistry, Schrödinger's equation is solved for a variety of cases. In doing so, the energies and wave functions of the system can be interpreted to provide connections with the physical system being studied. Solving this equation by hand for a one-dimensional system is a manageable task, but it becomes time-consuming once students aim to make various changes and investigate the impact of those changes on the results. To address this challenge, numerical methods, such as the shooting and linear finite-difference methods, have been utilized to quickly solve Schrödinger's equation. In this technology report, we use the Python programming environment and the three-point finite-difference numerical method to find the solutions and plot the results (wave functions or probability densities) for a particle in an infinite, finite, double finite, harmonic, Morse, or Kronig–Penney finite potential energy well. We believe that this technology report will educate undergraduates on the basic tools of computer programming, data analysis, and making connections between mathematical models and the physical systems with which they are associated.



KEYWORDS: *Upper-Division Undergraduate, Physical Chemistry, Computer-Based Learning, Computational Chemistry*

BACKGROUND

In quantum chemistry, Schrödinger's equation is written as

$$\hat{H}\psi = E\psi \quad (1)$$

where $\hat{H}$ is the Hamiltonian operator, ψ is the eigenvector or wave function, and E is the eigenvalue or energy of the system.[1] The Hamiltonian operator comprises kinetic and potential energy terms, where the kinetic energy component contains the Laplacian operator. For many of the cases addressed in undergraduate physical chemistry and in this report, the form of the Laplacian operator is one-dimensional in the x direction. By solution of Schrödinger's equation, the energies and wave functions of the system can be interpreted by students to make connections with the physical system being studied. For example, the particle in a finite potential energy well can be used to represent an atom and its ionization potential.

In this *Journal*, various tools (BASIC, Mathematica, Mathcad, Microsoft Excel spreadsheets, and Fortran with Java) have been utilized to find numerical solutions to Schrödinger's equation.[2−10] These have provided valuable contributions to chemical education, but all of them are limited in the number of cases and variables that students can investigate. For instance, Lang and Towns reported the use of Mathematica to visualize wave functions focused on the two-dimensional particle in a box, harmonic oscillator, and particle on a sphere models. However, they did not address the double finite well, Morse potential, and Kronig–Penney cases that we have included in our work. The Python program provided in this technology report provides a single resource for students to utilize and investigate many cases (infinite, finite, double finite of equal depth, double finite of unequal depth, harmonic, Morse, and Kronig–Penney potentials) that they encounter in their physical chemistry curriculum.

The Schrödinger program is coded in Python,[11] a modern scientific coding language that is free to users, in order to provide a tool with which science students will be familiar if they are taking introductory programming courses as part of their undergraduate curriculum. High-resolution graphical output is also provided for data analysis and inclusion in written reports. Moreover, our goal is that others will incorporate this tool in their physical chemistry classrooms, and therefore, we have provided several tools to aid in accomplishing this. Most notably, we have utilized the IPython notebook feature of the Python programming environment, providing a novel approach to teaching Schrödinger's equation in the physical chemistry classroom. Utilizing the IPython notebooks, instructors have the ability to teach theory and code components from a single resource rather than juggling lecture notes and a separate window for running their code. In the subsequent sections, we outline a pedagogical framework for

Received: January 3, 2017
Revised: March 31, 2017
Published: April 17, 2017

DOI: 10.1021/acs.jchemed.7b00003
J. Chem. Educ. 2017, 94, 813−815

utilizing the IPython notebooks in the classroom. In addition, we have also provided an example assignment and pre/post-assignment survey questions to study the project's impact on student learning. Together, instructors' use of the IPython notebooks coupled with the students' use of the full Schrodinger.py program will allow undergraduates to numerically solve Schrödinger's equation and graphically visualize the wave functions and their energies.

METHODS

The program presented herein is divided into three components: the main Python code (Schrodinger.py), a utilities program written in version 2.7 of the Python programming language (utils2.py), and a utilities program written in version 3.5 of Python (utils3.py). The utilities programs detect which version of Python a user is running and also contain various functions and subroutines that are called by the main Schrodinger.py code. The Schrodinger.py program runs on Windows, OSX, and Linux operating systems. We advise using Python with Anaconda. Instructors and students can find explicit installation instructions in the Python_Files folder of the Supporting Information. Comments are also provided to explain various aspects within the code.

In order to run the Schrodinger.py program, Windows users can open an Anaconda prompt, navigate to the directory where they have saved the program, and run the program using the command **python Schrodinger.py**. Likewise, OSX users would do the same, but they would run the program from the terminal window. In both the Anaconda prompt and terminal window, the program provides interactive feedback to the user, prompting various inputs. Users also have the option to save a high-resolution graphic (in PNG format) of their results from each case for inclusion in write-ups/reports.

After prompting the user for several inputs (length of the potential energy well, depth of the potential energy well, and distance between potential energy wells), the program then utilizes a three-point-finite-difference method,[12]

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad (2)$$

to construct a matrix representation of the Laplacian differential operator. In eq 2, h is some small number and represents a change in x. It is also important to note that the mass of the particle and $\hbar$ are set equal to 1 in all cases for the sake of calculating energies in atomic units (au). With this in mind, users are also prompted to enter the length, depth, and distance between potential energy wells in atomic units, recalling that 1 Å = 1.89 au. Next, the Hamiltonian is calculated, and SciPy's built in "eigh" function is utilized to determine the eigenvectors (wave functions) and eigenvalues (energies). SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.[13] The program then plots the results (e.g., Figure 1) for students to interpret and answer several fundamental questions in each case. It is also important to note that students and instructors have the option to plot wave functions or the probability density at the conclusion of each calculation.

PEDAGOGICAL FRAMEWORK

To use these tools in the classroom, we first advise that instructors take advantage of the IPython notebooks provided for the seven cases addressed in the Schrodinger.py program.

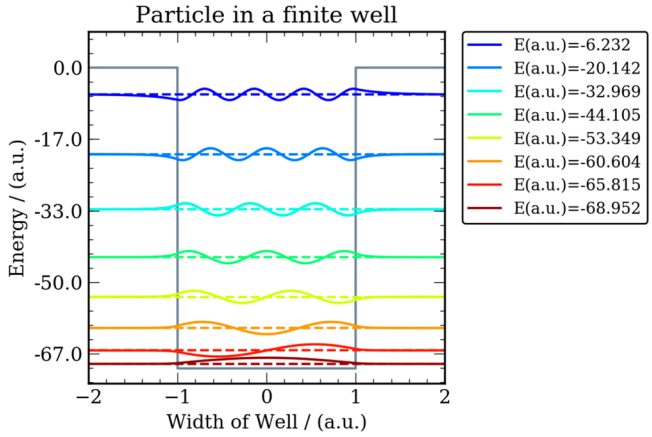


Figure 1. Eight bound states for a particle in a finite potential energy well of width = 2 atomic units (au) and depth = 70 au.

These notebooks are located within the Python_Files folder in the Supporting Information. The notebooks are named Case1–7.ipynb, where Case1 = infinite well, Case 2 = finite well, Case 3 = double finite well of equal depth, Case4 = double finite well of unequal depth, Case5 = harmonic well, Case6 = Morse well, and Case7 = Kronig–Penney. Following the installation instructions document previously mentioned in Methods, instructors will open the IPython notebooks via Jupyter[14] on their Windows or OSX computer. Once opened, instructors will find that the notebooks are divided into "Code" sections that run the corresponding Schrodinger.py program and "Markdown" sections where the instructor is capable of typing up their lecture notes for classroom presentation. Here we have left the theory/lecture components of the IPython notebooks blank so that instructors can customize these sections to their desired format. In order to do so, instructors would simply click on a cell in their IPython notebook, then find the "Insert" tab at the top of their Jupyter window, and last, select insert cell above or below. It is important that instructors designate the new cell as "Markdown" format instead of "Code" before they begin populating their lecture content. For a brief guide on getting started with "Markdown" notation, users are encouraged to utilize the links provided at the end of the Installation_Instructions.docx document in the Supporting Information. We feel that it is best that individual instructors populate their own lecture content because of the significant variability in notation.

Once the IPython notebooks are complete, instructors can utilize them in their classroom as they would their normal lecture slides. This provides a novel approach to teaching Schrödinger's equation in the physical chemistry classroom because it utilizes a single, clear resource for presenting theory and code content to students. Rather than alternating between lecture notes and another window for running their code, the IPython notebooks allow the instructor to present the material in an organized fashion, which ultimately results in the students being able to better focus on the content.

Accompanying the IPython notebooks are a preassignment survey (to be administered to students prior to beginning the assignment), an example assignment corresponding to the various cases included in the Schrodinger.py program, and a postassignment survey (to be administered to students following completion of the assignment). These documents should be utilized to evaluate the program's impact on student learning and can be found in the Supporting Information.

DOI: 10.1021/acs.jchemed.7b00003
J. Chem. Educ. 2017, 94, 813–815

Instructors are encouraged to modify these three documents as they see fit.

CONCLUSION

This technology report presents a Python program for solving Schrödinger's equation for a particle in an infinite, finite, double finite, harmonic, Morse, or Kronig–Penney finite potential energy well. The Schrodinger.py program provides students experience with the Python programming language and numerical approximations for solving differential equations. In addition, this technology report also introduces a novel approach to teaching Schrödinger's equation in undergraduate physical chemistry courses through the use of IPython notebooks. Using the provided assignment, students then communicate their findings through a written report that addresses several fundamental quantum mechanics topics/questions.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available on the ACS Publications website at DOI: 10.1021/acs.jchemed.7b00003.

Full Python code (Schrodinger.py) for OSX terminal or Anaconda prompt in Windows; IPython notebooks for each of the seven cases outlined above (Case*N*.ipynb), to be used by instructors in their classrooms; Python installation instructions (Installation_Instructions.docx); Jupyter configuration file (jupyter_notebook_config.py); pre- and postassignment surveys; and an example assignment for students (ZIP)

AUTHOR INFORMATION

Corresponding Author

*E-mail: msrnec@franciscan.edu.

ORCID

Matthew N. Srnec: 0000-0001-5386-1603

Notes

The authors declare no competing financial interest.

[§]Deceased March 14, 2017.

Users are encouraged to check back for updates to the code along with additional plotting capabilities. Code updates will be available at https://github.com/maduraresearchgroup.

ACKNOWLEDGMENTS

The authors acknowledge NSF Award 1259941, providing support for M.N.S. In addition, M.N.S. and S.U. dedicate this work to the deceased Dr. Jeffry D. Madura.

REFERENCES

(1) Levi, A. F. J. *Applied Quantum Mechanics*; Cambridge University Press: Cambridge, U.K., 2003.

(2) Hansen, J. C. Schrödinger.m: A Mathematica package for solving the time-independent Schrödinger equation. *J. Chem. Educ.* **1996**, *73* (10), 924.

(3) Tellinghuisen, J. Accurate numerical solutions of the one-dimensional Schrödinger equation. *J. Chem. Educ.* **1989**, *66* (1), 51.

(4) Beddard, G. S. Solution of the Schrödinger equation for one-dimensional anharmonic potentials: an undergraduate computational experiment. *J. Chem. Educ.* **2011**, *88* (7), 929–931.

(5) Ge, Y.; Rittenhouse, R. C.; Buchanan, J. C.; Livingston, B. Using a spreadsheet to solve the Schrödinger equations for the energies of the ground electronic state and two lowest states of H_2. *J. Chem. Educ.* **2014**, *91* (6), 853–859.

(6) Johnson, J. L. Visualization of wavefunctions of the ionized hydrogen molecule. *J. Chem. Educ.* **2004**, *81* (10), 1535.

(7) Lang, P. L.; Towns, M. H. Visualization of wavefunctions using Mathematica. *J. Chem. Educ.* **1998**, *75* (4), 506.

(8) Rioux, F. Quantum mechanics using Mathcad 3.0. *J. Chem. Educ.* **1992**, *69* (9), A240.

(9) Poiares, J. P. M.; Rodrigues, S. P. J.; Marques, J. M. C. A quantum mechanics toolkit: useful internet toolkit to teach fundamental concepts of quantum mechanics. *J. Chem. Educ.* **2008**, *85* (4), 591.

(10) Francis, T. A.; Miles, D. G., Jr. A graphical approach to the angular momentum Schrödinger equation. *J. Chem. Educ.* **2001**, *78* (3), 405.

(11) Python Software Foundation. Python Language Reference, version 2.7. http://www.python.org (accessed March 2017).

(12) Burden, R. L.; Faires, J. D. *Numerical Analysis*; Brooks/Cole, Cengage Learning: Boston, MA, 2011.

(13) Jones, E.; Oliphant, E.; Peterson, P.; et al. SciPy: Open Source Scientific Tools for Python. http://www.scipy.org/ (accessed March 2017).

(14) Pérez, F.; Granger, B. E. IPython: A System for Interactive Scientific Computing. *Comput. Sci. Eng.* **2007**, *9* (3), 21–29. Available at http://ipython.org/ (accessed March 2017).

DOI: 10.1021/acs.jchemed.7b00003
J. Chem. Educ. 2017, 94, 813–815