A Student-Friendly Guide to Molecular Integrals

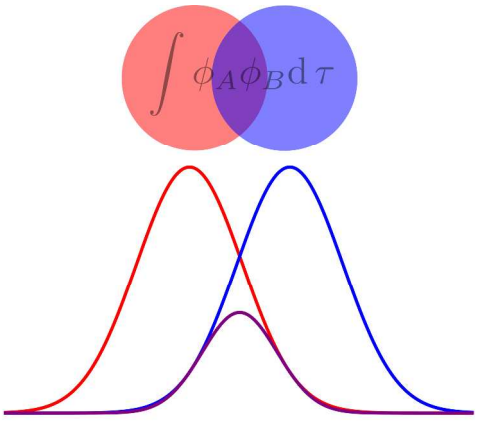
Kevin V. Murphy,* Justin M. Turney, and Henry F. Schaefer III

Center for Computational Quantum Chemistry, University of Georgia, Athens, GA, 30602, USA

ABSTRACT

Preceding even the Hartree–Fock method, molecular integrals are the very foundation upon which quantum chemical molecular modelling depends. Discussions of molecular integrals are normally found only in advanced and technical texts or articles. The objective of the present article is to provide less experienced readers, or students in a physical/computational chemistry course, a thorough understanding of molecular integrals. Through a series of detailed Handouts, the student/reader can participate in the derivation of molecular integrals, and in turn implement them in computer code. Hartree–Fock theory is discussed in enough detail to motivate the molecular integrals and address such topics as the atomic orbital basis. An introduction to the programming language of choice, Python3, is provided, tailored towards developing the essential skills necessary for implementing molecular integrals. The article is intended to be useful not only to instructors of physical/computational chemistry, but also to any reader who has independently sought a primer on this elusive subject.

GRAPHICAL ABSTRACT



KEYWORDS

Physical Chemistry; Quantum Chemistry; Computational Chemistry; Theoretical Chemistry; MO Theory; Graduate Education / Research; Upper-Division Undergraduate; Computer-Based Learning

INTRODUCTION

The fledgling computational quantum chemist is often directed to implement (program) the Hartree—Fock (HF) method in computer code. This is done as an exercise to help her or him better understand this powerful method of constructing molecular orbitals from atomic orbitals. Being able to implement HF requires having access to "molecular integrals," quantities on which the HF method performs its iterative operations. To this end, the student may be given a set of text files, each filled with a long list of cryptic numbers for reading in by their program. Alternatively, the student is advised to use an extant quantum chemistry program to compute these integrals, so that they may call them into their code. Too often the origin and form of these fundamental quantities remains a mystery.

Many students and professional computational quantum chemists go about their work with only a vague idea of what molecular integrals are and how they are generated. A perusal of the literature does little to enlighten. To redress this unfortunate state of affairs, we offer the present article. The objective is to elucidate precisely what molecular integrals are, how their mathematical form is derived, and how to implement them in code. It is written to be accessible to the undergraduate who has completed, or is taking part in, the physical chemistry sequence. We hope it will be a useful resource for instructors of undergraduate or graduate-level physical and computational chemistry courses. Among other features, the present article distinguishes itself from the few other introductory treatments of molecular integrals by presenting methods that are applicable to orbitals of arbitrary angular momentum, for all atoms, of any molecule, with any geometry.

The molecular integrals that are derived and presented here are not the most computationally efficient – that is, they are not the fastest means of generating these numbers. These molecular integrals are the explicit, "block" equations proposed by Taketa, Huzinaga, and O-ohata in 1966.[1] There is a rich literature dedicated to the purpose of efficiently computing molecular integrals, but unfortunately it is incomprehensible to all but the initiated. The point of this article is to provide basic understanding. When the reader has completed the exercises, they will be armed with the knowledge necessary to interpret the advanced literature. Additionally, in a future article we will address some of the more modern methods of molecular integral evaluation, such as the recursion schemes of the Obara—Saika method[2] and Head-Gordon—Pople method,[3] the quadrature of Dupuis and King,[4] and

others. The present article establishes the fundamental knowledge and skills upon which a discussion of those methods will depend.

In this main article, we provide an overview and preview of the structure and contents of the pedagogical documents (derivations, programming projects, and more), which are available as downloadable Supporting Information documents (see Associated Content). Following the preview, we conclude this main article with a brief discussion of the authors' experience using these materials with two undergraduate and two graduate students. Scaffolding techniques for instructors who seek to tailor the project to the level of their students are provided.

Quantum chemistry is a vast, deep subject. In the interest of clarity and space, we skim over fundamental but tangential details of quantum mechanics such as anti-symmetry. Fortunately, most of these will have been addressed in a physical chemistry course. There are many excellent textbooks[5—7] and *J. Chem. Ed.* articles,[8—20] if review is needed.

PEDAGOGICAL CONTENT OVERVIEW

Molecular modelling based upon quantum mechanics is a complex topic. In this article we strive to provide a *comprehensible* and *comprehensive* introduction to the foundational quantities upon which all such modelling depends: the molecular integrals. Because of the detail necessary to explicate the topic in this way, the bulk of the article appears in separate documents of the Supporting Information (SI), as outlined in Associated Content. We term five of these documents "Handouts," because they are designed to be ready for use by instructors as handouts to their students. The sixth SI is a compilation of computer code, input files, and output files.

While it is recommended that the Handouts be read in order, it is at the instructor's (or general reader's) discretion to take advantage of their modular structure, selecting those pieces useful to their purposes; the Handouts have been written to be as independent of one another as possible, including only a few cross-references. For example, instructors of graduate courses in quantum chemistry that assume some programming experience on the part of the student may choose to bypass Handout 3, which is an introduction to programming. By contrast, instructors of an undergraduate course in computational or physical chemistry may assign only Handouts 1 (background), part of 2 (derivation

through the overlap integrals), and 3 (a detailed introduction to programming, guiding through the implementation of the overlap integrals).

We now provide a brief preview of the contents of each Handout.

In Handout 1, molecular integrals are motivated particularly with respect to their role in Hartree—Fock theory. We begin with a discussion of the Schrödinger equation,

$$\hat{H}\Psi = E\Psi \tag{1}$$

which is solved on a computer for a molecule by recasting it into the Hartree—Fock—Roothan—Hall equation, which in matrix form is

$$\mathbf{FC} = \epsilon\mathbf{SC} \tag{2}$$

The matrix **S** is the overlap integral matrix, and each matrix element is made up of an integral (rather, the integral's solution) of the following general form

$$S_{AB} = \int \phi_A \phi_B \, \mathrm{d}\tau \tag{3}$$

where ϕ_A and ϕ_B are atomic orbitals. Each atomic orbital ϕ has the general form

$$\phi = \sum_p \mathrm{d}_p \underbrace{x^l y^m z^n}_{angular} \underbrace{\mathrm{e}^{-\alpha_p r^2}}_{radial} \tag{4}$$

The angular component of the atomic orbital ϕ has powers of angular momentum for each Cartesian coordinate (e.g. l = 0, m = 1, n = 0 describes a p_y orbital). The radial component is expressed as a Gaussian function instead of the more accurate exponential function ($e^{-\alpha r}$), because Gaussian functions make evaluation of the integrals easier. They do so by allowing two atom-centered orbitals, ϕ_A and ϕ_B, to be integrated over with respect to a single, intermediate center, as Figures 1 and 2 reveal. By summing several such Gaussian functions, accuracy approaching that of the exponential function can be achieved. The parameters d_p and α_p are values found in basis set tables.

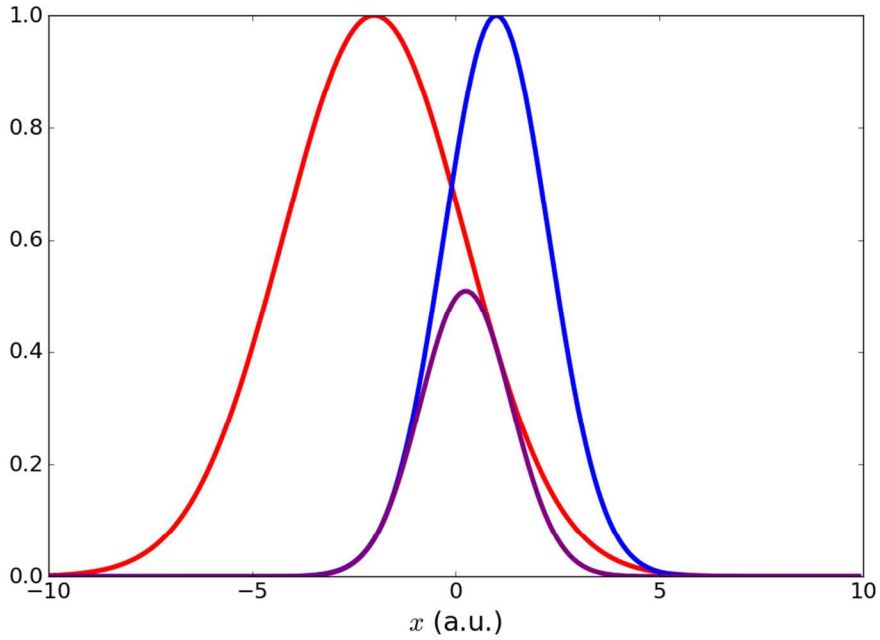


Figure 1. The product of two gaussian functions (red and blue) yields a third, intermediate Gaussian function (purple). Details are provided in Handout 1 (see Associated Content).

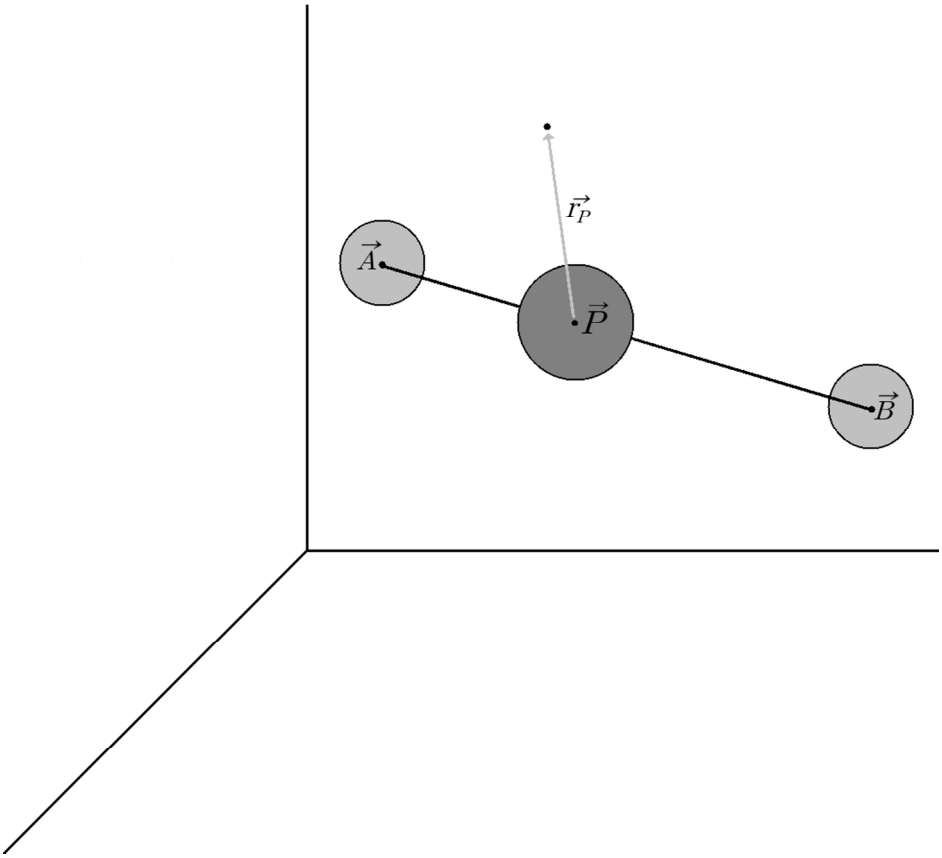


Figure 2. In taking the product of orbitals ϕ_A and ϕ_B, each represented by Cartesian Gaussian functions, a third center $\vec{P}$ located between $\vec{A}$ and $\vec{B}$ is formed. $\overrightarrow{r_P}$ is the distance between the electron and the third center $\vec{P}$.

The Fock matrix **F** is effectively the sum of three matrices (**T**, **V**, **G**), the matrix elements of which constitute the other three molecular integrals that we address. The elements of kinetic energy integral matrix **T** have the general form

$$T_{AB} = \int \phi_A \left(-\frac{1}{2}\nabla^2 \right) \phi_B \, \mathrm{d}\tau \tag{5}$$

The elements of electron-nuclear attraction integral matrix **V** have the general form

$$V_{AB} = \int \phi_A \left(\frac{-Z_C}{r_{iC}} \right) \phi_B \, \mathrm{d}\tau \tag{6}$$

The elements of electron-electron repulsion integral matrix (or tensor) **G** have the general form

$$G_{ABCD} = \int \phi_A \phi_B \left(\frac{1}{r_{ij}} \right) \phi_C \phi_D \, \mathrm{d}\tau \tag{7}$$

the solution of which requires an intermediate center over not just orbitals ϕ_A and ϕ_B, but an additional intermediate center over orbitals ϕ_C and ϕ_D.

To address these solutions, as well as the structure of the matrices **S**, **T**, **V**, and **G** that they make up, we discuss the meaning of an atomic orbital basis. This is particularly important because molecular orbitals are formed from the atomic orbital basis, by iteratively solving the Hartree—Fock—Roothan—Hall equation (Equation 2). We discuss matrix and tensor structures in Handouts 1 and 2.

The full, explicit derivation of the solutions to Equations 3, 5, 6, and 7 – for the solutions are what we seek to implement in computer code – is provided in Handout 2. The equations are based on the original 1966 paper by Taketa, Huzinaga, and O-ohata,[1] and use notation developed by Cook.[21] The derivation entails Fourier transforms, which mitigate the challenge of the inverse operators present in Equations 6 and 7. The student (reader) is encouraged to actively participate in the derivation through guided exercises. For example, as the student (reader) derives the solution to the angular component

of an overlap integral, they are given the questions that appears in Figure 3. They are encouraged to cover up solutions, which follow a solid black line.

One of the polynomials in Eq. 13 can be expanded as

$$(x - \vec{A}_x)^{l_A} = \{(x - \vec{P}_x) + (\vec{P}_x - \vec{A}_x)\}^{l_A} = (x_P - \vec{PA}_x)^{l_A} \,. \tag{15}$$

We have employed the x-component of the vector quantity $\vec{P}$, which is the third Gaussian center located between centers $\vec{A}$ and $\vec{B}$. Using the standard binomial expansion,

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k \,, \tag{16}$$

where $\binom{n}{k}$ is the binomial coefficient meaning "n choose k" or

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \,, \tag{17}$$

we can recast Eq. 15.

Exercise.
Express the right-most side of Eq. 15 as a binomial expansion, with a binomial coefficient "l_A choose i."

Solution.

$$(x_P - \vec{PA}_x)^{l_A} = \sum_{i=0}^{l_A} \binom{l_A}{i} (\vec{PA}_x)^{l_A - i} (x_P)^i \tag{18}$$

[end of solution]

Figure 3. An example of an exercise appearing in Handout 2, which serves to further the derivation of the solution to the overlap integral S_{AB} (Equation 3).

Some of the questions are quite simple, as suggested in Figure 3. Others are more involved, such as one directed to deriving the equation for the normalization constant *N*, an equation which is ubiquitous throughout the solutions to the molecular integrals. Regardless of an exercise's complexity, students are given detailed guidance and adequate information to solve the problem. Completion of the exercise serves to move the derivation forward. Most exercises are designed to take no more than about 15 minutes, though this will depend on the student's proficiency with what is largely algebraic mathematics.

Handout 3 is a gentle introduction to the Python programming language for those with little or no prior programming experience. It is tailored towards programming molecular integrals. Python was chosen over other languages for both its relative ease of use as well as its power. Once again, a series

of exercises are included for the student (reader) to practice using their newfound knowledge. For example, Figure 4 shows a coding exercise wherein students are asked to practice their for loop skills, as well as hone their ability to translate mathematical equations into code. They must type in the provided block of code and replace the placeholder text that appears in dark blue with functioning code. The skills developed in Handout 3 prepare the reader for Handout 4.

14. Define another function for computing the the constant c_k. Referring to Eq. 21 of Handout 2, and calling the `binomial` function of the `special` module (part of the SciPy library), type in the following code:

```
def compute_ck(arguments):
  for i in range(l+1): # i from 0 up to and including l, hence +1
    #for loop over j
      if i + k == ?:
        ck += special.binom(l,k) * ? * a**(l-i) * ?
  return ?
```

Complete the code.

Figure 4. An example of a programming exercise that appears in Handout 3. Students are asked to complete the function that calculates the coefficient c_k, by typing in the provided code, and replacing the dark blue text with functioning code.

The coding introduction of Handout 3 is followed by the full implementation of the molecular integrals in code in Handout 4. As with the other documents (except for Handout 3 which occasionally references Handout 2), this can be utilized independently of the other Handouts. The construction of each molecular integral matrix (**S**, **T**, **V**, **G**) and its elements is treated in turn. While the instructions in Handout 4 are highly detailed, they are less overtly "guided" than those in Handout 3, requiring the application of deeper problem-solving strategies. Hints and tips are provided throughout, including directions for using special functions that perform certain mathematical operations with simple keywords, such as the calculation of factorials and double factorials.

Having calculated the molecular integrals, Handout 5 describes their deployment in the implementation of the HF method. This Handout is quite brief by comparison to the other handouts, highlighting the complexity of molecular integrals as compared to the HF "algorithm." Completion of Handouts 4 and 5 marks the development of a complete, independent quantum chemistry program.

The sixth Supporting Information is a compilation of the authors' implementation of the programming projects. As with answers in the back of a mathematics textbook, the reader should use these with care. It is strongly advised to reference these codes (`.py` files) only as a last resort and only as needed. Otherwise the learning value is forfeit.

Additionally, SI 6 includes molecule coordinate (`.xyz`) input files and output integral matrix files (in `.txt` format). For example, `out.S.h2o.txt` contains the "printed" calculated overlap integral matrix corresponding to the `h2o.xyz` input file. These input files are included as "test cases," so that reference to the respective output integral matrices enables one to ensure that her or his code generates the same, correct integral matrix.

RESULTS AND SCAFFOLDING TECHNIQUES FOR INSTRUCTORS

A sample of four students completed the central programming project, corresponding to Handout 4. This was preceded by a lecture addressing most of the content present in Handout 1 and parts of the derivations in Handout 2. Two of the four students were undergraduates who had just completed physical chemistry. The other two were recent graduates about to begin graduate studies. Every student completed the entire set of programming projects. The completion times for each student were approximately 8 hours, 20 hours, 35 hours, and 40 hours.

The two students who completed the project in the least amount of time worked independently and required no assistance. The first of these (8 hours) was an undergraduate studying computer science and chemistry. The second student (20 hours) had graduated with majors in chemistry, mathematics, and a minor in computer science. The student who completed the project in 35 hours was a graduate and had a few, clarifying questions, but was otherwise independent; this student did consult online resources for coding guidance. The student's background was in chemistry and mathematics, and had acquired some previous coding experience. The student who completed the project in 40 hours was an undergraduate studying chemistry, and had no experience in programming or formal background in mathematics. Several hours of close assistance from one of the authors was required to complete the project, though it should be noted that Handout 3 (introduction to programming) was not available to the student at this time. Other than the student who used online resources for coding guidance, no students indicated the need to consult outside references about molecular integrals, other than for personal interest in the subject.

The students indicated that the most difficult part of the programming projects was the complexity of the equations needing to be implemented, increasing the possibility of introducing errors that would later require debugging. One student suggested that maximally functionalizing code (i.e. writing

separate functions for frequently used tasks), and having reference integral matrices to compare against (as we have provided in SI 6), helped reduce this difficulty. The students reported finding the lecture and programming project instructions clear and helpful. The predominant reported benefit was the clarity the exercises brought to their previously vague and abstract conceptions of molecular integrals. One student also wrote, "It made the sheer number of equations computational quantum chemists use more viscerally real, and demonstrated the importance of efficiency in code." The students reflected that the experience helped them develop a more intuitive understanding of each molecular integral type, especially two-electron integrals, and their relation to atomic and molecular features like orbital position, orientation, and type (e.g. *s, p, d*). They described having a greater appreciation for the use of Gaussian functions to approximate Slater-type orbitals. The HF method itself, which some of them had previously implemented, "made more sense" as a result of these exercises.

While one must be careful of making generalizations from a very small sample, it is reasonable to infer that students with more computer programming experience, especially those having formally studied computer science, will complete the projects significantly faster than those without it. More formal experience in mathematics probably helps as well, though the magnitude of this effect is difficult to parse.

The authors offer to instructors four possible strategies for scaffolding the material to students of different levels:

1. Students should have access to reference integral matrices for several molecules. That is, for a few specific `.xyz` files ("test cases") students should also be given the correct integrals. Students can use these test cases to directly compare the output of their code to the correct molecular integral values, which is especially valuable when debugging. During debugging, patterns often appear in the integral matrices that make it possible to deduce, or at least narrow down, the location of offending code. To this end, `.xyz` files and the correct integrals for H_2, HeH^+, HF (hydrogen fluoride), and H_2O are packaged as a `.zip` file in SI 6.
2. The overlap integrals are the easiest to implement. It may be appropriate to have some students code just the S matrix. (Handout 3 guides the reader with no programming

experience through the implementation of this integral matrix via an especially detailed set of exercises.) If they go on to implement the HF code, which is a relatively straightforward exercise, the instructor can provide the code or values for **T**, **V**, and **G**. It is worth noting that the kinetic energy integrals, **T**, are a fairly direct extension of the overlap integrals in **S**, and do not typically require much more work to implement. There is a relatively large difficulty gap between coding these integrals (**S** and **T**) and coding the two potential energy integrals (electron-nuclear attraction **V** and electron-electron repulsion **G**) due to the complexity of the matrix element equations.

3. Some students may benefit from working in pairs, though this has not been assessed. A common practice in software development is pair programming, in which two programmers work together at one computer.[22] For the current project, one student would write the code while the other reviews each line of code as it is typed in. The two students should switch roles frequently.
4. Particularly for undergraduate physical/computational chemistry labs, the instructor could provide incomplete code to be filled in by the student. (This is demonstrated in Handout 3.) There are at least two ways of utilizing this strategy:
    a. Within a function, provide the beginning of a block of code that reveals a pattern – for example, the code for the *x*-component of the overlap integral. Students can observe the pattern and replicate it for the *y*- and *z*-components.
    b. Provide largely completed code where there are just a few empty functions that need to be written. This might entail the student implementing a handful of equations (such as the Boys Function, or the coefficient c_k; see Handout 3 or Handout 4), where each equation requires only a few lines of computer code.

By any of these methods, students are exposed to the logic of molecular integrals and how they are built in the atomic orbital basis, and are given the opportunity to practice basic coding skills without the steeper demand of generating the code entirely from scratch. For those students that are members of a computational chemistry summer research undergraduate program, or are beginning graduate

students of computational quantum chemistry, they should be able to complete all of the projects independently, perhaps using limited assistance from fellow graduate students or an advisor.

ASSOCIATED CONTENT

Supporting Information

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

Handout 1: Background to molecular integrals and HF theory (PDF)

Handout 2: Derivation of the molecular integrals, with exercises (PDF)

Handout 3: Coding in Python: The essential skills, with exercises (PDF)

Handout 4: Implementation of the molecular integrals: Programming project (PDF)

Handout 5: Using the molecular integrals: HF programming project (PDF)

Computer files: Test case input and output files, and authors' Python implementation (ZIP)

AUTHOR INFORMATION

Corresponding Author

*E-mail: kevinmurphy@uga.edu

ACKNOWLEDGMENTS

We acknowledge support from the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-SC0018412, and the U.S. National Science Foundation, Grant No. CHE-1661604.

REFERENCES

1. Taketa, H.; Huzinaga, S.; O-ohata, K. Gaussian-Expansion Methods for Molecular Integrals. *J. Phys. Soc. Jap.* **1966**, *21* (11), 2313–2314.
2. Obara, S.; Saika, A. Efficient Recursive Computation of Molecular Integrals Over Cartesian Gaussian Functions. *J. Chem. Phys.* **1986**, *84* (7), 3963–3974.
3. Head-Gordon, M.; Pople, J. A. A Method for Two-Electron Gaussian Integral and Integral Derivative Evaluation Using Recurrence Relations. *J. Chem. Phys.* **1988**, *89* (9), 5777–5786.
4. King, H. F.; Dupuis, M. Numerical Integration Using Rys Polynomials. *J. Comp. Phys.* **1976**, *21* (2), 144–165.

5. McQuarrie, D. A.; Simon, J. D. *Physical Chemistry: A Molecular Approach*, 1st ed.; University Science Books: California, 1997; pp 70–243.
6. McQuarrie, D. A. *Quantum Chemistry*, 2nd ed.; University Science Books: California, 2008; pp 97–432.
7. Levine, I. *Quantum Chemistry*, 7th ed.; Prentice Hall: New York, 2014; pp 142–187.
8. Parson, R. Visualizing the Variation Principle: An Intuitive Approach to Interpreting the Theorem in Geometric Terms. *J. Chem. Ed.* **1993**, *70* (2), 115–119.
9. Zúñiga, J.; Bastida, A.; Requena, A. Using the Screened Coulomb Potential to Illustrate the Variational Method. *J. Chem. Ed.* **2012**, *89* (2), 1152–1158.
10. Ge, Y. Let Students Derive, by Themselves, Two-Dimensional Atomic and Molecular Quantum Chemistry from Scratch. *J. Chem. Ed.* **2016**, *93* (12), 2033–2039.
11. Goodfriend, P. L. Simple Perturbation Example for Quantum Chemistry. *J. Chem. Ed.* **1985**, *62* (3), 202–204.
12. Aebersold, D. Integral Equations in Quantum Chemistry. *J. Chem. Ed.* **1975**, *52* (7), 434.
13. Lucas, J. M.; Mota, F.; Novoa, J. J. Theoretical Study of the Vibrational-Rotational Spectra of Diatomic Molecules: A Quantum Chemistry Experiment. *J. Chem. Ed.* **1986**, *63* (10), 919–920.
14. Weninger, S. J. The Molecular Structure Conundrum: Can Classical Chemistry be Reduced to Quantum Chemistry? *J. Chem. Ed.* **1984**, *61* (11), 939–944.
15. Li, W.-K.; Blinder, S. M. Introducing Relativity into Quantum Chemistry. *J. Chem. Ed.* **2011**, *88* (1), 71–73.
16. El-Issa, H. D. The Particle in a Box Revisited. *J. Chem. Ed.* **1986**, *63* (9), 761–764.
17. Pettit, B. A. The Morse Oscillator and Second-Order Perturbation Theory. *J. Chem. Ed.* **1998**, *75* (9), 1170.
18. Besalú, E.; Martí, J. Exploring the Rayleigh-Ritz Variational Principle. *J. Chem. Ed.* **1998**, *75* (1), 105–107.
19. Ma, N. L. Quantum Analogies on Campus. *J. Chem. Ed.* **1996**, *73* (11), 1016–1017.
20. Hall, P. G. An Introduction to Quantum-Mechanical Operators. *J. Chem. Ed.* **1966**, *43* (1), 38–39.

21. Cook, D. B. *Handbook of Computational Quantum Chemistry*, 2nd ed.; Dover: New York, 2005; pp 217–251.

22. Williams, L.; Kessler, R. R.; Cunningham, W.; Jeffries, R. Strengthening the Case for Pair Programming. *IEEE Software* **2000**, *17* (4), 19–25.